

Anvar Narzullayev

Pythonda

dasturlash asoslari

+50 dan
ortiq video
darslar



Veb dasturlar, mobil ilovalar, sun'iy intellekt,
robototexnikaning eng ilg'or tilini oson va
mukammal o'rgatuvchi qo'llanma



Anvar Narzullayev

Pythonda

dasturlash asoslari

Veb dasturlar, mobil
ilovalar, sun'iy intellekt,
robototexnikaning eng
ilg'or tilini oson va
mukammal o'rgatuvchi
qo'llanma



Toshkent
«Akademnashr»
2021

UO'K: 004.42
KBK: 32.973-018
N 25

N 25 Narzullayev Anvar

PYTHONda dasturlash asoslari [Matn]: Ilmiy-omnabop /
A.Narzullayev. – Toshkent: "Akademnashr", 2021. – 336 b.

ISBN 978-9943-6964-5-7

UO'K: 004.42
KBK: 32.973-018

Ushbu kitob bugungi kunda dasturlash asoslariga oid o'zbek tilidagi mukammal tuzilgan qo'llanmalardan biridir. Qo'lingizdagi kitobning o'ziga xos jihati shundaki, uning har bir bo'limi uchun tayyorlangan qo'shimcha onlayn materiallar, jumladan, 50 dan ortiq video darslar, amaliy mashg'ulotlar va vazifalarning kodlari e'tiboringizga havola etilgan. O'quvchilar bu materiallarni maxsus QR kod yordamida o'z kompyuterlariga yuklab olib, ulardan umumiy foydalanishi mumkin.

ISBN 978-9943-6964-5-7

© A.Narzullayev
«PYTHONda dasturlash asoslari»
© «Akademnashr», 2021

MUQADDIMA

Bugungi kunda kompyuter savodxonligi o'qish va yozishni bilishdek dolzarb bo'lib qoldi. Kundalik hayotimizni elektron qurilmalarsiz tasavvur ham qilib bo'lmaydi. Kompyuterlar u yoki bu ko'rinishda (telefon, noutbuk, aqlli soatlar, televizor, avtomobillar) hayotimizning ajralmas qismiga aylandi. Bu kompyuterlar uchun yozilgan dasturlar esa turmush tarzimizni butunlay o'zgartirib yubordi.

Kun davomida o'nlab, balki yuzlab dasturlar og'irimizni yengil qiladi. Ertalab uyqudan turishga alohida dastur, kun davomida yaqinlarimiz bilan muloqot qilishga alohida dastur, taksi chaqirish uchun, ovqat buyurtma berish uchun, yangiliklardan xabardor bo'lish uchun, namoz vaqtlarini eslatib turish uchun, sevimli kino va seriallarni ko'rish uchun, ijtimoiy tarmoqlarda masofadan fikr almashish (talashib-tortishish 😊) uchun, onlayn ta'lim olish uchun va boshqa minglab ehtiyojlarimiz uchun alohida dasturlar mavjud.

Dasturlarga talab bor ekan, tabiiyki, ularni yaratuvchi dasturchilarga ham kundan kunga ehtiyoj ortib bormoqda. AQSh statistika byurosi hisobotiga ko'ra, 2020-yilda faqatgina AQShning o'zida dasturchilar uchun 500 ming ish o'rinlari bo'sh qolgan bo'lsa, 2021-yilda bu ko'rsatkich 1,4 millionga yetgan. 2020-yilda dunyo bo'ylab malakali mutaxassislarining tangisligi 40 millionni tashkil qilgan va eng ko'p talab aynan dasturchilarga to'g'ri kelgan.

O'zbekistonda ham bu soha mutaxassislariga bo'lgan ehtiyojning balandligini ularga taklif etilayotgan maoshlardan ko'rishimiz mumkin. Bugungi kunda eng yuqori maoshlar aynan axborot texnologiyalari yo'nalishi, xususan, dasturiy ta'minot yaratuvchilari orasida ekanligi hech kimga sir bo'lmasa kerak. Shunday ekan, siz ham imkoniyatni qo'ldan boy bermasdan, eng dolzarb mutaxassislar qatoridan joy

olish uchun harakatni boshlayvering. Ushbu kitob yetuk va malakali dasturchi bo'lish yo'lidagi safaringizning debochasi bo'ladi deb umid qilamiz.

Maqсадimiz

Darsligimizning asl maqsadi – sizda dastur va dasturlash haqida tushuncha hosil qilish, sizni tez va samarali yo'llar bilan dasturlash olamiga olib kirish, dasturlash asoslarini, eng muhimi, turli muammolarga yechim bo'luvchi dasturlar yozishni o'rgatish. Buning uchun Python tilidan foydalansak-da, kitobimizda yoritilgan mavzular va asosiy tushunchalar barcha dasturlash tillari uchun umumiydir.

Darslarimiz kim uchun?

Kitobimiz umrida biror marta Python yoki umuman boshqa tillarda dastur yozmagan barcha yoshdagi insonlarga mo'ljallangan.

Darslarimiz davomida siz barcha tillar uchun umumiy bo'lgan dasturlash asoslarini puxta o'zlashtirib olasiz, bu esa, o'z navbatida, kelajakdagi faoliyatingiz uchun muhim poydevor bo'ladi.

Umid qilamizki, kitobimiz boshqa o'qituvchilar va dasturchilar uchun ham foydali manba bo'lib xizmat qiladi.

Darslarimiz sizga nima beradi?

Darslarimiz yakunida siz nafaqat Python tilini, balki barcha dasturlash tillari uchun umumiy bo'lgan tushunchalar va asoslarni puxta o'zlashtirib olasiz.

Kitobni muvaffaqiyatli tamomlagan tinglovchilar kelajakda dasturlashning tor va murakkab yo'nalishlarini ham, mutlaqo yangi dasturlash tilini ham osongina o'zlashtira oladilar.

Kitobimiz ikki qismdan iborat. Birinchi qism dasturlash asoslarini o'rgatgan holda sizni murakkab dasturlar yozishga tayyorlaydi. Ikkinchi qism esa to'liqicha amaliy bo'lib, 1-qismda o'rgangan bilimlaringizni qo'llagan holda bir nechta loyihalar ustida ishlaysiz.

Kitobimizning avvalida kompyuter va kompyuter dasturlari haqida gaplashamiz, dasturlash tillari haqida umumiy tushunchalarga ega bo'lamiz.

Navbatdagi bo'limga kitobni o'zlashtirish uchun kerak bo'ladigan ish qurollari bilan tanishamiz, kompyuterimizni sozlaymiz va barcha dasturchilar uchun birinchi qo'yiladigan qadam – "Hello World" dasturini yozamiz.

Keyin esa sekin-astalik bilan dasturlash olamiga qadam tashlaymiz. Nafaqat Python, balki har qanday dasturlash tillari uchun muhim tushunchalar bilan tanisha boshlaymiz, jumladan, darslarimiz davomida quyidagi mavzular bilan tanishamiz:

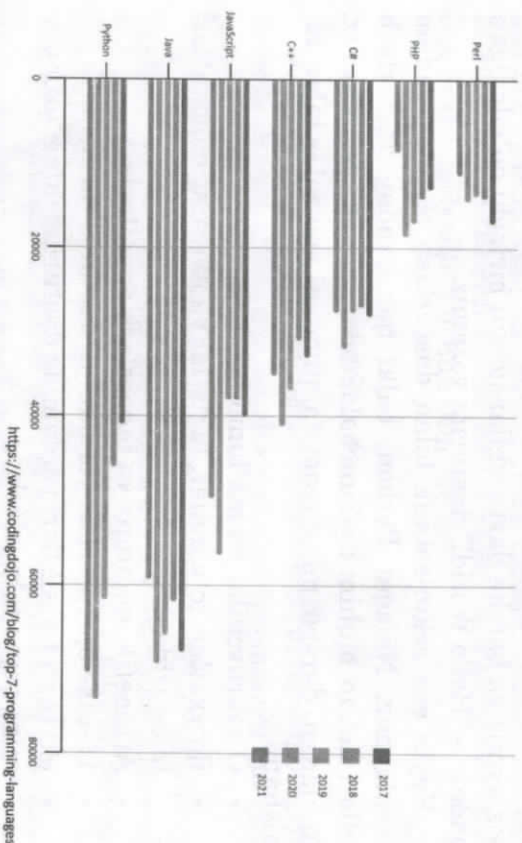
- O'zgaruvchilar va ma'lumotlar turlari
- Ro'yxatlar, to'plamlar, lug'atlar va ularning elementlari bilan ishlash
- Arifmetik, mantiqiy va taqqoslash operatorlari
- if-else operatorlari yordamida dasturni tarmoqlash
- while va for sikllari yordamida dasturimizning biror qismini qayta takrorlash
- Interaktiv dasturlar yaratish orqali foydalanuvchilar bilan ikki tomonlama "mulogot" o'rnatish, ulardan ma'lumot qabul qilish
- Kodning ma'lum qismlarini qayta-qayta ishlatish uchun funksiyalar yozish
- Obyektga yo'naltirilgan dasturlash tamoyillari yordamida murakkab dasturlar yozish
- Yozgan dasturingizni avtomatik ravishda tekshiruvchi testlar yozish orqali kelajakdagi xatolarning oldini olish

Nima uchun aynan python?

Python o'rganish oson, foydalanish uchun qulay, ko'p qirrali dasturlash tili bo'lib, dasturlashga yangi kirganlar uchun ham, soha mutaxassislari uchun ham zo'r tanlovdir.

Python tilini o'rganish uchun 5 sabab:

1. Python dasturlash tiliga bo'lgan ehtiyoj yildan yilga oshib bormoqda. CodingDojo portalining tadqiqotlariga ko'ra, 2020 – 2021-yillarda aynan Python tilida dasturlovchi mutaxassislariga eng ko'p talab bo'lgan.



1-RASM. Dasturlash tillariga bo'lgan talab

2. Python Artificial Intelligence (Sun'iy intellekt) va Data Science (Ma'lumotlar muhandisligi) sohalarining tili hisoblanadi. Bugungi kunda keng ommalashib borayotgan sun'iy intellekt asosida ishlovchi dasturlarning aksari Pythonda yozilgan. **Bu sohalaridagi mutaxassislar bugungi kunda eng noyob va qimmatbaho kadrlar hisoblanadi.**

3. Keng qamrovli va universal til. Python dasturlari deyarli barcha operativ tizimlarda va platformalarda ishlaydi.

4. O'rganish uchun ham, tushunish uchun ham juda qulay va soddalik kod. Quyidagi ikki tilda yozilgan kodlarga e'tibor bering va ulardan qay biri tushunarliroq ekanini aytib (ikkisi ham bir vazifani bajaradi):

```
# PYTHON DASTURLASH TILI
print("Assalomu Alaykum!")

// JAVA DASTURLASH TILI
public class Main {
    public static void main(String[] args) {
        System.out.println("Assalomu Alaykum!");
    }
}
```

5. Moslashuvchanlik. Python dasturlash tili ma'lum bir masalalarni yechish bilan chegaralanmagan. Bu til dasturchilarga yangi va yangi yo'nalishlarga kirish imkonini beradi. Python quyidagi sohalarida qo'llaniladi: Veb va Internet dasturlash, kompyuter o'yinlarini yaratish, ma'lumotlar bazasi bilan ishlash (DB), computer vision, foydalanuvchilar uchun grafik interfeys (GUI), mobil ilovalar, buyumlar interneti (IoT) texnologiyasi va hokazo.

Darslarni qanday o'zlashtiramiz?

Ushbu darslik bugungi kunda dasturlash asoslariga oid bo'lgan o'zbek tilidagi eng mukammal qo'llanma desak xato qilmaymiz. Bunday deyshimizga sabab, qo'lingizdagi kitobga qo'shimcha ravishda har bir mavzu uchun alohida onlayn video darslarni ham qo'lga kiritasiz. Video darslarga bog'lama QR kod va sahifa manzili ko'rinishida beriladi. Bundan tashqari, dars davomida yozilgan dasturlar, amaliy mashg'ulotlar, uy vazifalari va ularning javoblarini ham bizning internet sahifamizdan yuklab olishingiz mumkin.

Dasturchilik faqatgina bilim emas, balki dastur yozishga ko'nikma hosil qilish hamdir. Shuning uchun biror mavzuni o'rganish jarayonida kitobdagi kodlarni biz bilan birga yozib boring. Mavzu so'ngida berilgan amaliy mashg'ulotlarni albatta bajaring. So'ngra biz bergan javoblar bilan solishtiring va, muhimi, javobingizni tahlil qiling. Umuman olganda, ushbu kitob va u bilan birga keluvchi qo'shimcha manbalardan unumli foydalaning.

Kitodagi har bir mavzuga oid videoni ko'rish uchun mavzu boshida berilgan QR kodni skanerda o'qing yoki QR kod ostidagi manzilga kiring.

Shuningdek, videolarimiz uchun Telegramda ham maxsus kanal ochganimiz: t.me/sdvideolar. Telegramdagi @saridev kanali orqali esa boshqa tinglovchilar yoki bevosita kitob muallifi bilan bog'lanishingiz va o'zingizni qiziqtirgan savollarga javob olishingiz mumkin.

GitHub sahifasidagi kodlarni arxiv ko'rinishida yuklab
olish uchun sahifa tepasidagi Code menyusiga kiring va Down-
load ZIP tugmasini bosing.

<https://cli.co/kirish>

1-QISM

ASOSLAR

Kitobimizning 1-qismi har bir dasturchi uchun muhim bo'lgan mavzularni o'z ichiga oladi. Bu qismda yoritilgan g'oyalardan va tushunchalardan barcha dasturlash tillari uchun umumiy bo'lgani uchun kelajakda juda ko'p asqatadi.

1-bobda siz kompyuter, dastur va dasturlash haqida tushunchalarga ega bo'lasiz. Dasturchi va koder o'rtasidagi farqni o'rganasiz.

2-bobda dasturlash uchun kerak bo'lgan vositalar bilan tanishamiz. Har bir dasturchining ilk dasturi bo'lgan "Hello world!" dasturini yozamiz.

3-bobda ma'lumotlarni kompyuter xotirasida saqlashni o'rganamiz va turli ma'lumot turlari bilan tanishamiz. Jumladan, sonlar, matnlar va ro'yxatlar bilan ishlashni o'rganamiz. Bo'lim yakunida for sikli bilan tanishamiz.

4-bob shartlar va tarmoqlanish haqida bo'ladi. Bu bobda biz turli ifodalarni taqqoslashni va taqqoslash natijasiga ko'ra dasturni tarmoqlashni o'rganamiz.

5-bobda ma'lumotlarni lug'at va to'plam ko'rinishida saqlashni, ularga murojaat etishni va ular ustida turli amallar bajarishni ko'rib chiqamiz.

6-bob while sikli yordamida dasturni qayta-qayta takrorlashni o'rganamiz.

7-bob to'liq xatolarga bag'ishlangan. Bu yerda biz xatolarning turlari va ularni dastur jarayonida ushlab olishni o'rganamiz. Shuningdek, dastur yozishda ko'p uchraydigan xatolarning oldini olishni ko'ramiz.

8-bobda dasturlashning muhim tamoyillaridan biri bo'lgan funksiya tushunchasi bilan tanishamiz. Funksiyalar yordamida ma'lum bir kodlarni jamlab qo'yish va istalgan payt ularga murojaat etishni o'rganamiz.

9-bob to'liqligicha obyektga yo'naltirilgan dasturlashga bag'ishlangan. Bu bobda biz klasslar va obyektlar tushunchasi bilan, obyektga yo'naltirilgan dasturlashning muhim tamoyillari bilan tanishamiz.

10-bobda fayllar bilan ishlash, jumladan, turli ma'lumotlarni fayldan o'qish, faylga yozish va ishlov berishni ko'rib o'tamiz.

11-bobda dasturlarni tekshirish uchun maxsus test dasturlar yozishni o'rganamiz. Test dasturlar yordamida funksiyalar va klasslarning to'g'ri ishlashini tekshiramiz.

12-bobda Python dasturlash tilining kutubxonalari bilan tanishamiz. Dasturlash davomida kerak bo'ladigan foydali ichki va tashqi modullar haqida gaplashamiz.

1-BOB

KOMPYUTER VA DASTUR

Dastur nima ekanligini tushunish uchun avval kompyuterlar haqida gaplashaylik. Kompyuter – bu ma'lumotlarga ishlov beruvchi elektron qurilma. Kompyuterining asosiy vazifasi ma'lumotlarni o'qish, ularning ustida turli hisob-kitoblarni bajarish va saqlashdir.

① Kompyuter ingliz tilidagi "compute" – hisoblash so'zidan olingan.

Asrlar davomida odamlarning aqliy mehnati samaradorligi ularning miyasi gay darajada tez ishlashi bilan chegaralangan. Ilk kompyuterlarning vazifasi mana shu chegarani buzib o'tish, murakkab matematik hisob-kitoblar orgali shu vaqtgacha yechimi yo'q deb bilingan muammolarga yechim topish bo'ldi. Vaqt o'tishi bilan kompyuterlar deyarli barcha sohalarida qo'llana boshladi va insoniyatning asrlar davomida saqlanib kelgan turmush tarzini butkul o'zgartirib yubordi.

Bugungi kunda kompyuterlar ishlatilmaydigan sohalarining o'zi yo'q. Ishlab chiqarish, sanoat, qishloq xo'jaligi, arxitektura, san'at, savdo, aloqa, ta'lim, boshqaruv, huquq, musiqa, siyosat, ilm-fan, transport va boshqa sohalarini kompyutersiz tasavvur qilib bo'lmaydi.

Statista.com portalining tadqiqotlariga ko'ra, bugungi kunda yer yuzida internet tarmog'iga ulangan kompyuterlar soni 22 milliarddan oshiq bo'lib, bu ko'rsatkich 2025-yilda 38 milliardga, 2030-yilda esa 50 milliardga yetishi taxmin qilinmoqda. Ya'ni bugungi kunda kompyuterlar soni yer kurrasi aholisi sonidan salkam 3 barobar ko'prog ekan.

1.1. Kompyuterning ichki tuzilishi

Yugorida zikr qilinganidek, kompyuterning asosiy vazifasi ma'lumotlarga ishlov berish va saqlashdir. Bu vazifalarni bajarish uchun har qanday kompyuter (xoh qo'lingizdagi aqlli soat, xoh NASA raketalarini fazoga uchiruvchi super-kompyuterlar) 3 qismdan iborat bo'ladi:

1. **Protessor** – bu kompyuterning miyasi. Barcha hisob-kitoblar aynan protessor tomonidan bajariladi.
2. **Xotira** – ma'lumotlarni saqlash uchun.
3. **Kirish/Chiqish qurilmalari** – foydalanuvchidan ma'lumot olish va qaytarish uchun (klaviatura, ekran, sichqoncha, mikrofon va hokazo).

Ilk kompyuterlar muayyan vazifani bajarish uchun qurilgan va faqatgina bir turdagi muammolarga yechim bergan, xolos. Misol uchun, Ikkinchi jahon urushi paytida Alan Tyuring tomonidan yaratilgan "Bombe" elektron hisoblash mashinasi Germaniya armiyasining Enigma shifrini buzishga qaratilgan. Yoki 1941-yilda Jon Vinsent Atanasoff va uning shogirdi Klifford Berri tomonidan yaratilgan ABC kompyuteri (Atanasoff-Berry Computer) faqatgina chiziqli tenglamalar sistemasi yechma olgan, xolos. Ya'ni ilk kompyuterlarda dasturiy ta'minot tushunchasi bo'lmagan.

Vagt o'tishi bilan ko'p maqsadli protessorlar ixtiro qilindi. Bunday protessorlar bir turdagi emas, istalgan arifmetik hisob-kitoblarni bajara oladigan bo'ldi. Natijada bita kompyuterda turli xil masalalarga yechim topish imkoniyati paydo bo'ldi. Bunday kompyuterlarning xotirasiga turli ma'lumotlarni va bu ma'lumotlarga qanday ishlov berish haqidagi buyruqlarni joylash orqali istalgan natijani olish imkoniyati yaratildi. Boshqacha qilib aytganda, "dasturlash" tushunchasi yuzaga keldi.

Demak, zamonaviy kompyuterlardan foydalanish uchun qurilmalarning o'zi yetarli emas. Bu qurilmalarni to'g'ri ket-

ma-ketlikda ishga tushiruvchi dasturlar ham kerak ekan. Xo'sh, dastur nima? Keyingi bo'limda shu haqida gaplashamiz.

1.2. Dastur va dasturlash

Dastur – kompyuterga qaysi amallarni qanday ketma-ketlikda bajarishni ko'rsatuvchi ko'rsatmalar to'plami. Dasturlarga veb-brauzerlar, o'yinlar, matn muharrirlari, video pleyerlar misol bo'ladi. Kompyuterda istalgan vazifani bajarish uchun albatta qurilma va dasturdan foydalanamiz.

Masalan, telefon yordamida suratga olish uchun telefonimizda kamera va kamera bilan ishlaydigan maxsus dastur o'rnatilishi kerak. Xuddi shu kabi, elektron kitoblarni o'qish uchun kompyuter ekranidan va kitoblarni ochishga mo'ljallangan dasturdan foydalanamiz.

Demak, kompyuterdan foydalanish uchun nafaqat elektron qurilmalar, balki bu qurilmalarni ishga tushiruvchi dasturlar ham juda muhim ekan.

Bevosita dasturlarni yaratish jarayoni dasturlash deb ataladi. Yugoridagi gaplardan xulosa qiladigan bo'lsak, dasturlash kompyuterga turli xil vazifalarni bajarishga ko'rsatma berish ekan. Vazifalar soddaga (ikki sonning ko'paytmasini hisoblab, ekranga chiqar) yoki murakkab (A nuqtadan B nuqtaga marshrut topib ber) bo'lishi mumkin.

Vazifa qanchalik murakkab bo'lmasin, uni bir nechta soddaga qadamlarga (ko'rsatmalarga) bo'lishimiz mumkin. Ko'rsatmalarni kompyuter tushunadigan tilda yozish jarayoni kodlash (coding) deb ataladi.

Demak, dasturlash masala ustida fikrlash va muammoni bir nechta soddaga ko'rsatmalarga olib kelish bo'lsa, kodlash aytilgan qadamlarni bevosita kompyuter tiliga tarjima qilish ekan.



<https://olico/Da-dasturlash>

Odamlar yillar davomida kompyuterlar bilan mulogot qilish, odam tilidagi ko'rsatmalarni kompyuterga tushuntirish uchun turli dasturlash tillarini ixtiro qildilar. Bugungi kunda yuzlab dasturlash tillari mavjud bo'lib, ularning barchasi o'ziga xos xususiyatlarga ega va har birining o'ziga yaraisha afzalliklari va kamchiliklari bor. Ba'zi tillar mobil ilovalar yaratish uchun qulayroq bo'lsa, boshqa tillar veb-dasturlash, yana boshqasi esa murakkab tizimlar yoki robototexnika uchun maxsus mo'ljallangan. Shuning uchun ham "eng yaxshi" bo'lgan bitta til yo'q.

Umuman olganda, dasturlashga kirishda qaysi tilni tanlash o'ta muhim emas. Sababi, avval zikr qilganimizdek, dasturlash asoslari barcha tillar uchun umumiy.

2-BOB

ILK QADAMLAR

Kompyuterimizda dastur yozish uchun dastavval kompyuterimizda dasturlash tili, so'ngra dasturlash uchun muhit o'rnatilgan bo'lishi kerak. Dasturlash tili biz yozgan dasturimizni kompyuterga "uqtirish" uchun ishlatilsa, dasturlash muhitini dastur yozish uchun matn muharriri desak ham bo'ladi.

Agar turli sabablarga ko'ra sizda kompyuter yo'q bo'lsa, dasturlashni o'rganish uchun smartfon yoki planshetdan ham foydalanishingiz mumkin. Bu haqida batafsil ma'lumot olish uchun 2.3-bo'limga o'ting.

2.1. Python o'rnatamiz

Kod yozish uchun, avvalo, kompyuterimizda Python o'rnatilgan bo'lishi kerak. Agar sizning kompyuteringizda Windows yoki Mac operatsion tizimi o'rnatilgan bo'lsa, www.python.org/downloads/ sahifasiga kirasiz va Download tugmasini bosasiz.

Yuklab olingan faylni ochamiz va yangi ochilgan oynada Add Python 3.x to PATH matnini belgilaymiz va Install Now tugmasini bosamiz.



1 Darslarimiz Python 3.6 va undan yuqori versiya asosida tuzilgan.

Dastur o'rnatilgach, oynani yopamiz va Python to'g'ri o'rnatilganini tekshirib ko'ramiz. Buning uchun kompyuterimizda Command Prompt dasturini ochamiz va qora oynada quyidagi kodni yozamiz:

python -V

Dastur to'g'ri o'rnatilgan bo'lsa, ekranda Pythonning versiyasi ko'rsatiladi.



3-RASM. Python o'rnatilganini tekshirish

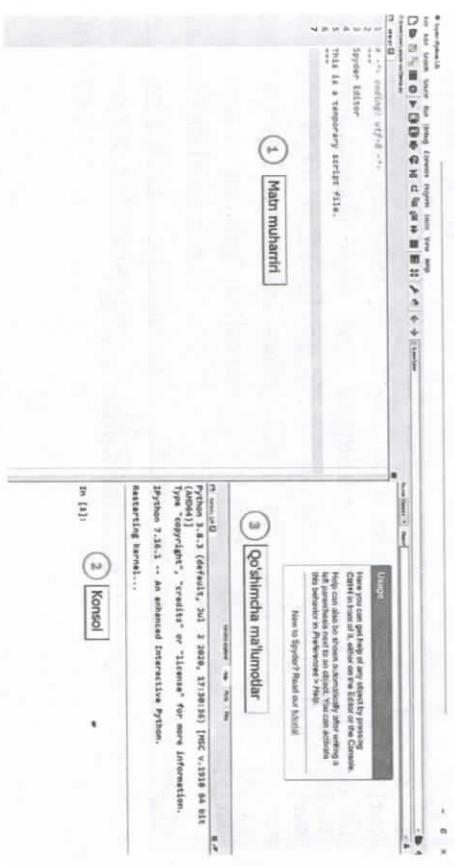
Agar yuqoridagi xabar o'rniiga xatolik chiqsa, demak, siz Python o'rnatish jarayonida Add Python 3.x to PATH degan joyini belgilamadan o'tib ketgansiz. Agar sizning kompyuteringizda Linux operatsion tizimi o'rnatilgan bo'lsa, katta ehtimol bilan Python ham o'rnatilgan.

2.2. Dasturlash muhiti

Dasturlash muhiti bevosita dastur yozish uchun kod muharriri vazifasini bajaradi. Dastur yozish uchun yuzlab turli muhirlar bor. Aslida, oddiy matn muharririda ham kod yozishimiz mumkin. Lekin dasturlash muhirlari dasturchilarning ishini yengillatishga qaratilgan qo'shimcha funksiyalar va foydali xossalarga boy bo'ladi.

Hozirgi kunda eng mashhur muhirlar PyCharm, Spyder, IDLE, Jupyter Notebook, Sublime va boshqalar. Bularning qay biridan foydalanish dasturchining o'ziga bog'liq. Matn muharriri kodimizning sifatiga yoki uning tez/sekin bajari-lishiga ta'sir qilmaydi. Biz darslarimiz davomida Spyder dasturlash muhitidan foydalanamiz va boshlanishiga sizga ham shu muhitni tavsiya etamiz.

Dasturni yuklab olish uchun www.spyder-ide.org sahifasiga kiring va yuqoridagi menyuda Download tugmasini bosib. Yangi oynada Download for Windows tugmasini bosib, Spyder dasturini yuklab oling. Dasturni o'rnatish jarayoni juda ham soddagina, faqatgina so'ralganda Next, Accept, Close tugmalarini bosib tursangiz kifoya.



4-RASM. Spyder dasturlash muhiti

Dastur o'rnatilgach, Windows menyusidan Spyder IDE dasturini topib, ochamiz.

Spyder muhiti uch qismdan iborat:

1. **Matn muharriri** – dasturlar yozish uchun.
2. **Konsol** – qisqa kodlarni bajarib, tekshirib ko'rish uchun.
3. **Qo'shimcha ma'lumotlar** oynasi. Bu yerda quyidagi ma'lumotlarni ko'rish mumkin:

- Turli funksiyalar haqida ma'lumotlar (Help)
- Dasturdagi o'zgaruvchilar ro'yxati (Variables)
- Grafiklar (Plots)
- Fayllar (Files)

Spyder konsolida qisqa kodlarni bajarib, natijasini ko'rish mumkin. Misol uchun, quyidagi rasmda ko'rsatilgan oddiy arifmetik amallarni konsolda bajarganmiz:

```
Python 7.16.1 -- An enhanced Interactive Python.
Restarting kernel...

In [1]: 7+8
Out[1]: 15

In [2]: 20/10
Out[2]: 2.0

In [3]: 9*8-55
Out[3]: 17

In [4]: |
```

5-RASM: Spyder konsoli

Amaliyot

- Kompyuterga Python dasturlash tilini o'rnatish.
- Kompyuterga Spyder (yoki o'zingizga yoqqan) dasturlash muhiti o'rnatish.
- Spyder konsolida turli arifmetik amallar bajarib ko'ring.
- Windowsda Command Prompt dasturini oching. Qora ekranda python deb yozing.
- Agar >>> belgisini ko'rgan bo'lsangiz, siz Python konsolidasiz. Yuqoridagi arifmetik amallarni shu oynada ham bajarib ko'ring.

2.3. Brauzerda dastur yozish

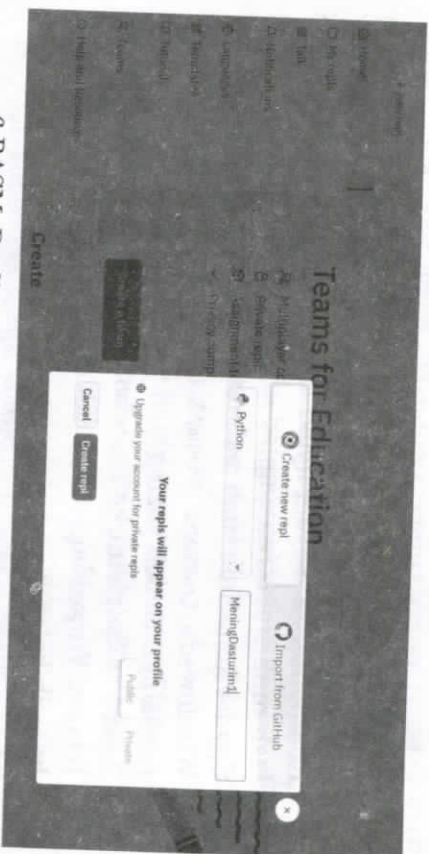
Agar siz turli sabablarga ko'ra kompyuteringizga Python o'rnatolmasangiz (yoki kompyuteringiz bo'lmasa), **replit.com** sahifasi yordamida Python dasturlarini to'g'ridan to'g'ri internet brauzerda ham yozishingiz mumkin (telefon, planshet, kompyuter).

Buning uchun dastavval [Replit.com](https://replit.com) sahifasidan ro'yxatdan o'ting. So'ngra yangi ochilgan oynada New repl tugmasini bosing. Dasturlash tili sifatida Pythonni tanlang, dasturga tushunarli nom bering va Create repl tugmasini bosing. Biz darsimiz davomida yozilgan kodlarni va mashg'ulotlarning javoblarni [Replit.com](https://replit.com) sahifasiga yuklab boramiz.

Yuqoridagi vazifalarni muvaffaqiyatli yakunlagan bo'lsangiz, tabriklaymiz, siz birinchi dasturingizni yozishga tayyorsiz!



<https://replit.com>



6-RASM. Replit.com sahifasida yangi dastur yaratish

2.4. Birinchi dasturimiz

Keling, an'anaga ko'ra barcha dastur-chilar birinchi yozadigan dastur – "Hello World!" dasturini yozamiz.

Buning uchun Spyder IDE yuqorisidagi menyuda File → New File bo'limini tanlaymiz (yoki klaviaturada Ctrl+N tugmalarini bosamiz). Matn muharririda yangi, untitled0.py fayli yaratildi.



<https://elico/Da-HelloWorld>

Faylga mazmunli nom beramiz, buning uchun File → Save as... menyusini tanlaymiz (yoki Ctrl+Shift+S) va yangi ochilgan oynada faylga ma'nili nom beramiz.

① Darslar davomida yaratilgan barcha fayllarni alohida papkada, tartib bilan saqlab boring

Faylni nomlashda quyidagi qoidalariga amal qiling:

- Fayl nomi qisqa, kichkina lotin harflari bilan va, eng muhim, bo'shiq (probe)l qo'ymasdan yozilgan bo'lishi kerak.
- Fayl nomi .py bilan tugashi lozim (misol uchun, fayl-nomi.py).

- Faylga ikki so'zdan iborat nom qo'ymog'chi bo'lsangiz, so'zlar orasini tire (-) yoki pastki chiziq (_) bilan ajratib. Misol uchun: hello-world.py yoki hello_world.py.
 - Fayl nomini sonlar bilan boshlamang.
- Yuqoridagi qoidalariga amal qilgan holda faylga hello.py deb nom beramiz (siz istalgancha nomlashingiz mumkin) va Save tugmasini bosamiz.

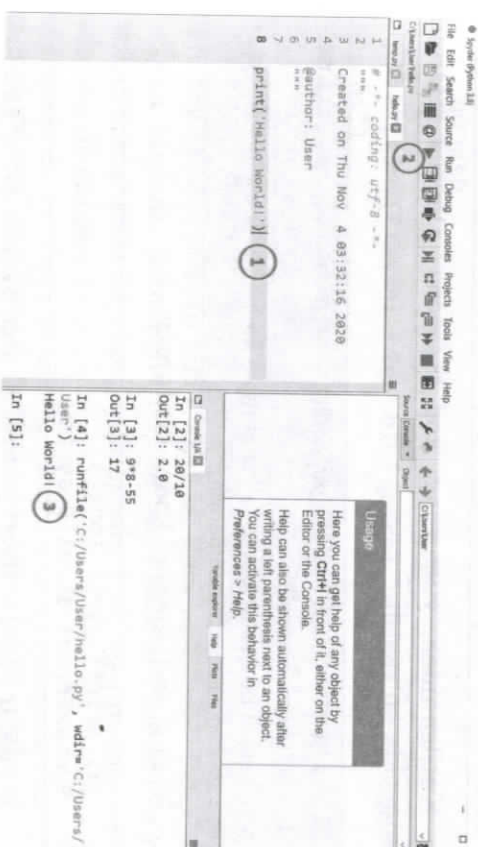
Spyder matn muharririda quyidagi kodni yozamiz:

```
print("Hello World!")
```

print() Python'dagi maxsus funktsiya bo'lib, qavslar ichida berilgan matn yoki ifodalarni konsolga chiqarish vazifasini bajaradi.

⚠ Dars davomida berilgan vazifalarni biz bilan birga bajarib boring. Bu juda muhim!

Kodimizni bajarish uchun muharrir tepasidagi Run (▶) yoki klaviaturadagi F5 tugmasini bosamiz (yangi ochilgan oynada ham Run tugmasini bosamiz) va konsolda natijani ko'ramiz:



7-RASM. Hello World dasturi va natijasi

Konsolda Hello World! yozuvini ko'rgan bo'lsangiz, tab-riklaymiz, siz Python'da birinchi dasturingizni yozdingiz.

Amaliyot

Spyder konsolida va matn muharririda quyidagi kodlarni bajarib ko'ring. Siz kutgan natijalar chiqdimi?

```
print('Assalomu alaykum')
print(Xayrli tong!)
print(2+4*2)
print(19/5)
print(2**4)
```

2.5. print() FUNKSIYASI

Avvalgi bo'limning yakunida bir nechta kodlarni Python'da bajarib ko'rishni vazifa qilib bergan edik. Keling, shu kodlar-ning natijasini ko'ramiz:

```
>>> print("Assalomu alaykum")
Assalomu alaykum
```

Kutilganidek, yuqoridagi kod Assalomu alaykum matnini konsolda ko'rsatdi.

① >>> belgilari bilan boshlangan kodlar matn muharririda emas, konsol-
da yozilganini bildiradi.
>>> belgisi bor qator bevosita biz kiritgan kodni, >>> belgisi yo'q qator
esa kodning natijasini anglatadi.

Keling, endi keyingi kodni kiritamiz:

```
>>> print(Xayrli tong!)
SyntaxError: invalid syntax
```

Bu safar Xayrli tong! yozuvi o'rniiga Syntax Error (sintak-
sis xatolik) xabari chiqdi. Xatolik qayerda?

Avval aytganimizdek, print() funk-
yasi matn yoki ifodalarni konsolga chiq-
arish vazifasini bajaradi. Lekin bu funk-
siya to'g'ri ishlashi uchun bir nechta qoidalarga
amal qilish lozim. Jumladan, agar konsol-
ga matn chiqarmoqchi bo'lsak, matnimiz
albatta qo'shtirnoqlar yoki (") yoki birtir-
noqlar (') orasida yozilishi kerak. Demak,
"Xayrli tong!" matnini konsolda chiqarish uchun to'g'ri kod
print("Xayrli tong!") yoki
print('Xayrli tong!') bo'ladi.

Agar matn bir nechta qatorga bo'lib yozish talab qilinsa,
matnni uchtalik qo'shtirnoqlar (""" """) yoki birtirnoqlar ('''
'') ichiga olib yozamiz:

```
>>> print("""Yaxshi otga bir qamchi,  
>>> Yomon otga ming qamchi""")
Yaxshi otga bir qamchi,  
Yomon otga ming qamchi
```

Qatorga bo'lishning yana bir usuli – qator so'ngida \n bel-
gisini qo'yish:

```
>>> print("Yaxshi otga bir qamchi,\nYomon otga ming  
qamchi")
Yaxshi otga bir qamchi,  
Yomon otga ming qamchi
```

Matnlar bilan ishlashda qo'shtirnoq yoki birtirnoq ishla-
tishning afalliklaridan biri – agar siz konsolga chiqarmoqchi
bo'lgan matnda ikkovidan biri qatnashgan bo'lsa, print()
funksiyasida ikkinchisidan foydalanasiz. Keling, quyidagi mi-
solni ko'rib chiqamiz:

```
>>> print('Men "Dell" markali noutbuk sotib oldim')
Men "Dell" markali noutbuk sotib oldim
```



<https://olicoo/Da-print>

Yuqoridagi matnda qo'shtirnoq ichidagi "Dell" so'zi bor matni konsolga chiqarish uchun `print()` funksiyasi ichida matni birtirnoq ichiga oldik. Bunda Python matn tarkibidagi qo'shtirnoqlarni e'tiborsiz qoldiradi.

Agar siz matni yozishda ham, matn tarkibida ham qo'shtirnoqlardan foydalanishni istasangiz, matn tarkibidagi qo'shtirnoqlardan avval \ belgisini qo'yasiz:

```
>>> print("Men \"Dell\" markali noutbuk sotib oldim")
Men "Dell" markali noutbuk sotib oldim
```

Yuqoridagi kodga e'tibor bergan bo'lsangiz, matn tarkibidagi "Dell" so'zi \"Dell\" ko'rinishida yozilgan. Umuman olganda, \ belgisi har qanday maxsus belgi oldidan qo'yiladi.

Agar yuqoridagi kodimizda \ belgisini ishlatmaganimizda, natija qanday bo'lar edi?

```
>>> print("Men \"Dell\" markali noutbuk sotib oldim")
SyntaxError: invalid syntax
```

Keling, shu o'rinda `SyntaxError` (sintaksis xatolik) haqida gaplashaylik.

2.6. Sintaksis xatolik (Syntax Error)

Har bir tilda bo'lgani kabi, dasturlash tillarining ham o'ziga yarashta qonun-qoidalar bor. Bu qoidalar to'plami **sintaksis (syntax)** deb ataladi. **Sintaksis xatolik** (`SyntaxError`) deganda esa shu qoidalarining buzilishi tushuniladi.

Misol uchun, keraksiz joyda qo'yilgan nuqta, vergul yoki bo'sh joy, shuningdek, ma'lum funksiyalar nomini xato yozish (`print()` o'rniga `prit()`), ochilmay yoki yopilmay qolgan qavs, no'rin bo'shliq, qolib ketgan kalit so'z kabilar ham `SyntaxError` hisoblanadi.

`Syntax Error` eng ko'p uchraydigan xatolik bo'lib, Python bunday xatolik bor dasturlarni **bajarmaydi**. Odatda, `Syntax Error` yuz berganida Python xato yuz bergan fayl, qator va kodni konsolga chiqaradi:

```
File "C:\python-darslar\03-print\helloWorld.py", line 1
    print("Hello World")
    ^
SyntaxError: EOL while scanning string literal
```

Yuqoridagi natijani quyidagicha o'qishimiz mumkin: "C:\python-darslar\03-print\" papkasidagi `helloWorld.py` faylining 1-qatorida (line 1), `print("Hello World")` kodida sintaksis xatolik yuz berdi. Xatoni topdingizmi?

Biz darslarimiz davomida turli sintaksis qoidalar va xatolar haqida o'z o'rnida yana to'xtalamiz.

2.6. Arifmetik amallar

`print()` funksiyasi nafaqat matn, balki turli ifodalarni ham konsolga chiqaradi. Ifodalar qo'shtirnoq (birtirnoq) ichida yozilmaydi, sababi, Python tirnoqlar orasidagi har qanday ifodani matn deb qabul qiladi.

Keling, amaliyotdagi quyidagi kodlarni ham bajaramiz:

```
>>> print(2+4*2)
10
```

Python arifmetik amallarni bajarishda matematikadagi ketma-ketlik qoidalariga amal qiladi:

- Gavs ichidagi amallar qavs ortidagilardan avval bajariladi.
- Darajaga oshirish (ildiz chiqarish) ko'paytirish va bo'lishdan avval bajariladi.

- Ko'paytirish va bo'lish qo'shish va ayirishdan avval bajariladi.
- Boshqa holatlarda ifodalar chapdan o'ngga qarab bajariladi.

Yuqoridagi misolda ham avval ko'paytirish ($4*2=8$), keyin esa qo'shish amali ($2+8=10$) bajarildi.

```
>>> print(19/5)
3.8
```

Ko'rib turganingizdek, / belgisi bo'lish amali bajaradi va natija har doim o'nlik son ko'rinishida bo'ladi (garchi bo'lish amali natijasida butun son hosil bo'lsa ham):

```
>>> print(20/5)
4.0
```

Bo'lish amalidan butun son ko'rinishidagi natija olish uchun // belgisidan foydalanamiz, bunda sonning o'nlik qismi tashlab yuboriladi:

```
>>> print(10//3)
3
```

Amaliyotimizdagi keyingi kodni ham bajaraylik:

```
>>> print(2**4)
16
```

Yuqoridagi ** belgisi darajaga oshirishni anglatadi, ya'ni $2**4$ ifodasi 2 ning 4-darajasini hisoblaydi.

print() yordamida matn va ifodalarni jamlab chiqarish ham mumkin. Buning uchun har bir ifoda va matn vergul (,) bilan ajratiladi:

```
>>> print("To'qqizning kvadrati", 9**2, "ga teng")
To'qqizning kvadrati 81 ga teng
```

Pythonda oddiy arifmetik amallar quyidagi jadvalda berilgan.

-	Ayirish	5-6=-1
/	Bo'lish	5/6=0.833333
**	EkspONENTA (daraja/ildiz)	5**6=15625 9**(0.5)=3

2.7. Izohlar

Yaxshi dasturlarning odatlaridan biri har qanday kodni izohlar bilan tushuntirib ketishdir. Izohlar kelajakda o'zimiz uchun ham, boshqalar uchun ham dasturimiz qanday ishlashini tushunishda yordam beradi.

Quyidagi ikki misolga e'tibor bering va ulardan qay biri tushunarliroq ekanini solishtiring.

Birinchi dastur:

```
print(2*5*3.14159)
```

Ikkinchi dastur:

```
# Radiusi 5 ga teng bo'lgan aylananing uzunligi:
print(2*5*3.14159)
```

Ikkinchi dasturning 1-qatorida # belgisi bilan boshlangan matn **izoh (comment)** deyiladi.

Izoh alohida qatoridan yoki qator oxiridan ham yozilishi mumkin. Python # belgisidan keyingi har qanday matnmi (qa-

tor oxirigacha) e'tiborsiz qoldiradi. # dan keyin yozilgan kodlar ham bajarilmaydi:

```
print("Assalomu alaykumi") # ushbu qator bajariladi
# Keyingi qator esa bajarilmaydi
# print("Mening ismim Anvar")
```

Natija: Assalomu alaykumi

❶ >>> belgisi yo'q kodlar matn muharririda yozilgan. Bu kodlarni bajarish uchun Run tugmasini bosasiz. Konsolda chiqadigan natijani alohida qatordan ajratib ko'rsatganimiz.

Amaliyot

Quyidagi matni aynan shunday ko'rinishda konsolda chiqaring:

- "Nexia", "Tico", 'Damas' ko'rganlar qilar havas.

Quyidagi misollarga yechimni Pythonda chiqaring. Har bir misoldan avval misol matnini izoh ko'rinishida yozing. Javoblarni savol va yechim ko'rinishida chiqaring: 5 ning 4-darajasi 625

- 5 ning 4-darajasini toping.
- 22 ni 4 ga bo'lganda qancha qoldiq qoladi?
- Tomonlari 125 ga teng kvadratning yuzi va perimetrlini toping.
- Diametri 12 ga teng bo'lgan doiraning yuzini toping ($\pi=3.14$).

Katetlari 6 va 7 bo'lgan to'g'ri burchakli uchburchakning gipotenuzasini toping (Pifagor teoremasidan foydalaning).

❶ Amaliyot javoblarini <https://github.com/anvarnarz/python-darslar> yoki <https://t.me/sdvideolar/56> bog'lamasi orqali yuklab olishingiz mumkin.

3-BOB

O'ZGARUVCHILAR VA MA'LUMOT TURLARI

Ushbu bobda dasturlash poydevori bo'lgan o'zgaruvchilar va turli ma'lumot turlari bilan tanishamiz.

3.1. O'zgaruvchilar

O'zgaruvchi (variable) – kompyuter xotirasida ma'lum bir qiymatni saqlash uchun ajratilgan joy. Biz o'zgaruvchilarga o'zimiz uchun tushunarli nom berib, istalgan payt o'zgaruvchi saqlayotgan qiymatni xotiradan chaqirib olishimiz mumkin. Pythonda qiymatlar son, matn, ro'yxat va boshqa ko'rinishda bo'lishi mumkin.

Quyidagi misolga e'tibor bering, biz 2 ta o'zgaruvchi yaratdik (ism va yosh) va ularga qiymatlar yukladik (Pythonda boshqa tillardagi kabi o'zgaruvchlarni avvaldan e'lon qilish yo'q):



<https://oliceo/Da-variables>

```
ism = "Abdulloh"
yosh = 25
print(ism)
print(yosh)
```

Natija:

```
Abdulloh
25
```

O'zgaruvchi bunday devilishiga sabab, uning qiymatini istalgan vaqt o'zgartirish mumkin:

```
>>> ism = "Abdulloh"
>>> print(ism)
Abdulloh
>>> ism = "Muhammad"
>>> print(ism)
Muhammad
```

Yuqoridagi misolda ism nomli o'zgaruvchiga avval Abdulloh, keyin esa Muhammad qiymatlarini berdik.

O'zgaruvchilarni nomlash

O'zgaruvchilarga nom berishda quyidagi qoidalarga amal qiling:

- O'zgaruvchi nomi harf yoki pastki chiziq (_) bilan boshlanishi kerak.
- O'zgaruvchi nomi raqam bilan boshlanishi mumkin emas.
- O'zgaruvchi nomida faqatgina lotin alifbosi harflari (A-Z), raqamlar (0-9) va pastki chiziq (_) qatnashishi mumkin.
- O'zgaruvchi nomida bo'shliq (probel) bo'lishi mumkin emas.
- O'zgaruvchi nomida bosh va kichik harflar turlicha talqin qilinadi (ism, ISM, va Ism uchta turli o'zgaruvchi).
- Qo'shimcha qoida sifatida:
 - O'zgaruvchi nomini kichik harflar bilan yozing.
 - O'zgaruvchi nomida 2 va undan ortiq so'z qatnashsa, ularning orasini pastki chiziq (_) bilan ajrating: `ism_sharif="Anvar Narzullayev"`
 - Ikki yoki undan ortiq so'zdan iborat nomlarni "tuya qilib" (camelCase) yozish ham mumkin: `foriyil=2021`.
- O'zgaruvchiga tushunarli nom bering (y=20 emas, yosh=20, d="Korea" emas, davlat="Korea" va hokazo).

Shuningdek, o'zgaruvchilarga Pythonida ishlatiladigan funksiyalar va maxsus kalit so'zlarning nomini berish mumkin emas. Kalit so'zlar ro'yxatini ko'rish uchun Spyder konsolida avval `help()` deb yozing va Enter tugmasini bosing. Keyin esa keywords deb kiritib, yana Enter bosing. Marhamat, ekraningizda Pythonidagi maxsus kalit so'zlar ro'yxatini ko'ryapsiz:

`help> keywords`

Here is a list of the Python keywords. Enter any keyword to get more help.

False	class	from	or
None	continue	global	pass
True	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield
break	for	not	

8-RASM. Pythonidagi kalit so'zlar

AMALIYOT

Quyidagi mashqlarni bajarang:

- "Hello World!" matnini yangi o'zgaruvchiga yuklang va `print()` yordamida konsolga chiqaring.
- xabar deb nomlangan o'zgaruvchiga biror matn yuklang va konsolga chiqaring, keyin esa o'zgaruvchiga yangi qiymat berib, uni ham konsolga chiqaring.
- class deb nomlangan o'zgaruvchi yarating, unga biror qiymat bering va konsolga chiqaring (siz kutgan natija chiqdimi?).

Quyidagi kodni matn muharririga yozing va bajarang:

```
radius = 5
pi = 3.14159
yuza = pi * radius**2
print("Radiusi", radius, "ga teng doira yuzi=", yuza)
```


3.2. STRING – MATN

STRING (matn) – Pythondagi eng muhim ma'lumot turlaridan biri. Avvalgi darsda ko'rganimizdek, o'zgaruvchiga matn yuklash uchun matn qo'shtirmoq (" ") yoki birtirmoq (' ') ichida yozilishi kerak.



<https://cli.co/DA-matn>

```
>>> viloyat = 'Фарғона'
>>> avto = 'Nexia 3'
```

Pythonda matnlar Unicode jadvalidagi istalgan belgilardan iborat bo'lishi mumkin (jumladan, kirill, lotin, arab, hind, xitoy alifbosidagi harflar yoki turi emoji-smayliklar).

```
>>> matn = "Men yangi" oldim
>>> print(matn)
Men yangi oldim
```

To'liq unicode jadvali bilan unicode-table.com sahifasida tanishish mumkin.

Currency Symbols		Roman Numerals		Card Suits	
¥	₹	I	♠	♠	♠
₹	₹	II	♣	♣	♣
₹	₹	III	♦	♦	♦
₹	₹	IV	♥	♥	♥
₹	₹	V	♠	♠	♠
₹	₹	VI	♣	♣	♣
₹	₹	VII	♦	♦	♦
₹	₹	VIII	♥	♥	♥
₹	₹	IX	♠	♠	♠
₹	₹	X	♣	♣	♣
₹	₹	XI	♦	♦	♦
₹	₹	XII	♥	♥	♥

9-RASM. UNICODE jadvalidagi harf, son va belgilar

MATNINLAR USTIDA AMALLAR

Matnlarni qo'shish

Matnlarni qo'shish uchun + operatoridan foydalanamiz:

```
ism = 'Ahmad'
print("Mening ismim" + ism)
```

Natija: Mening ismim Ahmad
Yana bir misol:

```
ism = 'Ahad'
familya = 'Qayum'
print(ism + familya)
```

Natija: Ahadqayum

Yuqoridagi koddagi ism va familiya orasiga bo'shliq belgisi qo'shmaganimiz uchun ikki matn qo'shib yozildi. Buni to'g'rilash uchun 3-qatorida ikki o'zgaruvchi orasiga bo'sh joy qo'shib yozamiz:

```
ism = 'Ahad'
famiIiya = 'Qayum'
print(ism + ' ' + famiIiya)
```

Natija:

Ahad Qayum

f-string

Ikki (va undan ko'p) matn ko'rinishidagi o'zgaruvchilarni birlashtirish uchun `f-string` usulidan (`f"{matn1} {matn2}"`) ham foydalanish mumkin:

```
ism = "Ahad"
familiya = 'Qayum'
ism_sharif = f"{ism} {familiya}"
print(ism_sharif)
```

Natija: Ahad Qayum

① f-string ichidagi o'zgaruvchilar katta qavslar ichida yoziladi.

Bu usul yordamida uzun matnlarni ham yasash mumkin:

```
fname = "James"
lname = 'Bond'
matn = f"Salom, mening ismim {lname}. {fname}
{name}!"
print(matn)
```

Natija: Salom, mening ismim Bond. James Bond!

f-string yordamida nafaqat matnlarni, balki turli ifodalarni ham jamlab yozishimiz mumkin. Bunda ifodalar va o'zgaruvchilar katta qavs {} ichida yoziladi.

```
tyil=2002
print(f"Siz {tyil}-yilda tug'ilgansiz.")
print(f"Yoshingiz {2021-tyil}da")
```

Natija:

```
Siz 2002-yilda tug'ilgansiz.
Yoshingiz 19 da
```

Maxsus belgilar

Maxsus belgilar yordamida matnga turli o'zgarishlar kiritish mumkin. Masalan, matnga bo'shliq qo'shish uchun \t belgisidan, yangi qatordan boshlash uchun \n belgisidan foydalanamiz:

```
>>> print('Hello World!')
Hello World!
>>> print('Hello \tWorld!') # \t - bo'sh joy qo'shish
Hello          World!
>>>> print('Hello \nWorld!')# \n - qator tashlash
Hello
World!
```

3.3. MATNLAR BILAN ISHLASH

Pythonda matnlar ustida amalga oshirish mumkin bo'lgan tayyor funksiyalar mavjud. Odatda, biror funksiya ma'lum bir obyektga (o'zgaruvchi, ma'lumot turiga) xos bo'lsa, bunday funksiyalar **metodlar** deb ataladi.

Metodlarni qo'llash uchun metod nomi matndan so'ng .metod_nomi() ko'rinishida yoziladi. Keling, shunday metodlarning ba'zilari bilan tanishamiz.

upper() va lower() metodlari

upper() metodi matndagi har bir harfni bosh harfga o'zgartiradi.

```
ism = 'anvar'
familiya = 'narzullayev'
ism_sharif = f"{ism} {familiya}"
print(ism_sharif.upper())
```

Natija: ANVAR NARZULLAYEV

lower() metodi har bir harfni kichik harfga o'zgartiradi.

```
ism = 'Anvar'
familiya = 'Narzullayev'
ism_sharif = f"{ism} {familiya}"
print(ism_sharif.lower())
```

Natija: anvar narzullayev

title() va capitalize() metodlari

title() metodi matndagi har bir so'zning birinchi harfini bosh harf bilan yozadi.

```
ism_sharif = 'james bond'
print(ism_sharif.title())
```

Natija: James Bond

capitalize() esa matndagi birinchi so'zning birinchi harfini katta bilan yozadi.

```
ism_sharif = 'james bond'
print(ism_sharif.capitalize())
```

Natija: James bond

Metodlarni faqat o'zgaruvchilarga emas, balki to'g'ridan to'g'ri matnga ham qo'llash mumkin (aslida, o'zgaruvchi ham matning (yoki boshqa ma'lumotning) manzili, xolos)

```
>>> print('james bond'.upper())
JAMES BOND
```

strip(),rstrip() va lstrip() metodlari

Bu metodlar matning boshi va oxiridagi bo'sh joylarni olib tashlaydi.

lstrip() matn boshidagi bo'shliqni, rstrip() matn oxiridagi bo'shliqni, strip() matn boshi va oxiridagi bo'shliqlarni olib tashlaydi. Quyidagi misolga e'tibor bering. Biz meva o'zgaruvchisi ichidagi olma so'zini ikki tomonidan ko'p joy tashlab yozdik. Keling, yuqoridagi metodlar shu matnga qanday ta'sir qilishini ko'ramiz:

```
>>> meva = "      olma      "
>>> print("Men " + meva.lstrip() + " yaxshi ko'raman")
Men olma      yaxshi ko'raman
>>> print("Men " + meva.rstrip() + " yaxshi ko'raman")
Men      olma yaxshi ko'raman
>>> print("Men " + meva.strip() + " yaxshi ko'raman")
Men olma yaxshi ko'raman
```

△ Yuqoridagi metodlar o'zgaruvchi ichidagi asl matni o'zgartirmaydi! O'zgartirilgan matni saqlab qolish uchun uni qaytadan o'zgaruvchiga yuklash lozim: meva = meva.strip()

```
>>> print("Men " + meva + " yaxshi ko'raman")
Men      olma      yaxshi ko'raman
>>> meva = meva.strip()
>>> print("Men " + meva + " yaxshi ko'raman")
Men olma yaxshi ko'raman
```

Matnlar bilan ishlaydigan metodlar ko'p. Ularning ba'zilari bilan kelajakda yana tanishamiz, to'liq ro'yxatni esa quyidagi sahifada ko'rishingiz mumkin:

www.w3schools.com/python/python_ref_string.asp

3.4. input() – FOYDALANUVCHI BILAN MULOQOT

Shu paytgacha biz o'zgaruvchilarning qiymatini dasturning ichida berayotgan edik. Keling, endi qiymatni o'zimiz emas, balki dastur foydalanuvchilariga kiritish imkonini beramiz.

Buning uchun Pythonda `input()` funksiyasidan foydalanamiz.

```
ism = input("Ismingiz nima?")
print("Assalomu alaykum, " + ism)
```

Yuqoridagi dastur avval 1-qatorda foydalanuvchining ismini so'raydi. Foydalanuvchi ismini kiritib, Enter tugmasini bosgach, foydalanuvchi kiritgan matn `ism` degan o'zgaruvchiga yuklanadi va dasturining 2-qatori bajaradi:

Natija:

```
Ismingiz nima? anvar
Assalomu alaykum, anvar
```

Keling, yuqoridagi kod va uning natijasini chiroyliroq ko'rinishga keltiramiz. Dastavval savol so'ngida `\n` belgisi yordamida foydalanuvchiga ismini yangi qatordan kiritish imkonini beramiz. Keyin esa `.title()` metodi yordamida ismini bosh harf bilan yozamiz. Darvoqe, bu safar `f-string` usulini qo'llaymiz:

```
ism = input("Ismingiz nima?\n>>>")
print(f"Assalomu alaykum, {ism.title()}")
```

Natija:

```
Ismingiz nima?
>>>anvar
Assalomu alaykum, Anvar
```

AMALIYOT

1. Pythonda quyidagi o'zgaruvchlarni yarating:

```
ko'cha="Bog'bon"
mahalla="Sag'bon"
tuman="Bodomzor"
viloyat="Samarqand"
```

2. Yuqoridagi o'zgaruvchlarni jamlab, quyidagi ko'rinishda konsolga chiqaring: Bog'bon ko'chasi, Sag'bon mahallasi, Bodomzor tumani, Samarqand viloyati

3. Yuqoridagi o'zgaruvchlarning (ko'cha, mahalla, tuman, viloyat) qiymatini foydalanuvchidan so'rang. Va avvalgi mashqni takrorlang.

4. Yuqoridagi matnni konsolga chiqarishda har bir verguldan keyin yangi qatordan yozing.

5. Yuqoridagi matnni `f-string` yordamida yangi, manzil deb nomlangan o'zgaruvchiga yuklang.

6. manzil o'zgaruvchisiga biz yuqorida o'rgangan `title()`, `upper()`, `lower()`, `capitalize()` metodlarini qo'llab ko'ring.

3.5. SONLAR

Dasturlash davomida turli sonlar bilan ishlash tabiiy hol. Pythonda sonlarning bir necha turlari bor. Quyida ular bilan yaqindan tanishamiz.



<https://oli.oo/Da-sonlar>

INTEGERS – BUTUN SONLAR

Butun sonlarni ham o'zgaruvchida saqlash, ularning ustida qo'shish (+), ayirish (-), ko'paytirish (*), bo'lish (/) kabi arifmetik amallarni bajarish mumkin:


```
>>> a = 20 # Sonlar musbat,
>>> b = -30 # manfiy yoki
>>> c = 0 # 0 ga teng bo'lishi mumkin
>>> d = a + b
>>> print(d)
-10
```

Python operatorlar orasidagi bo'shliqlarni inobatga olmaydi, shuning uchun tenglik belgisi yoki arifmetik operatorlardan oldin va keyin bo'sh joy tashlab yozish kodimizni o'qish uchun qulay qiladi.

```
# Kvadratning yuzini hisoblaymiz
kvdr_tmini = 20 # Kvadratning tomoni 20 ga teng
kvdr_yuzi = kvdr_tmini**2 # Kvadrat yuzini hisoblaymiz
print(kvdr_yuzi)
```

Natija: 400

FLOATS – O'NLIK SONLAR

Pythonda o'nluk sonlar **floating point numbers** yoki qisqa qilib **floats** deyiladi. "Floating point numbers" atamasini o'zbek tiliga "suzuvchi nuqtali sonlar" deb tarjima qilish mumkin. Ingliz tilida o'nluk sonlarni yozishda vergul (,) emas, nuqta (.) belgisi ishlatiladi va bu nuqta sonning katta-kichikligiga qarab joyi o'zgarгани uchun "floating" (suzuvchi) deyiladi.

```
pi = 3.14159 # o'nluk son (float)
radius = 10 # butun son (integer)
diametr = 2*radius
print("Aylana uzunligi ", pi*diametr, " ga teng.")
```

Natija: Aylana uzunligi 4.712384999999999 ga teng.

BUTUN SONDAN O'NLIK SONGA

Avval aytganimizdek, ikki butun sonni bo'lish (/) natijasida o'nluk son hosil bo'ladi (natija butun bo'lsa ham).

```
>>> a = -20
>>> b = 40
>>> c = b/a
>>> print(c) # natija o'nluk son bo'ladi
-2.0
```

Shuningdek, butun va o'nluk sonlar o'rtasidagi har qanday arifmetik amallarning natijasi ham o'nluk son bo'ladi.

```
>>> a = 2 # butun son
>>> b = 3.0 # o'nluk son
>>> print(a+b)
2.0
>>> print(a*b)
6.0
>>> print(a**b)
8.0
>>> print(2*(a+b))
10.0
```

UZUN SONLARNI KIRITISH

Uzun sonlarni kiritishda qulaylik uchun raqamlarni pastki chiziq () yordamida guruhlash mumkin. Python son tarkibidagi pastki chiziq(larni () inobatga olmasdan, uzun sonligicha qabul qiladi.

```
>>> aholi = 33_580_000
>>> print(f"O'zbekiston aholisi {aholi} dan ortiq")
O'zbekiston aholisi 33580000 dan ortiq
```

KONSTANTA

Aksar dasturlash tillarida konstant qiymatlar tushunchasi bor. Konstantalar o'zgarmas bo'ladi (misol uchun, π ning qiymati konstanta, o'zgarmas qiymat). Pythonda konstanta tushunchasi yo'q, shuning uchun dasturchilar bunday o'zgaruvchilarning nomini bosh harflar bilan yozishga kelishib olganlar (ogohlantirish sifatida). Bu, albatta, qat'iy qonun emas, lekin kelajakda o'zgaruvchilar orasida konstanta qiymatlarini ajratish uchun yaxshi usul.

```
PI = 3.14159 # konstanta
radius = 21.2
```

BIR NECHTA O'ZGARUVCHIGA QIYMAT BERISH

Birdaniga bir nechta o'zgaruvchiga qiymat berish uchun o'zgaruvchilar va ularga mos qiymatlar vergul (,) bilan ajratiladi:

```
>>> x, y, z = 10, -7.25, -30
```

Yuqoridagi kod x ga 10, y ga -7.25, va z ga -30 qiymatini yuklaydi.

Bu usulni istalgan turdagi o'zgaruvchilarga qo'llash mumkin:

```
>>> yosh, ism = 36, 'olimjon'
>>> print(f"{ism.title()} {yosh} yoshda")
olimjon 36 yoshda
```

O'ZGARUVCHI TURINI ALMASTIRISH

Keling, quyidagi misolni ko'rib chiqaylik, maqsadimiz ism va $yosh$ degan ikki o'zgaruvchini yangi xabar degan o'zgaruvchiga yuklab, "Jobir 16 yoshda" degan matnни konsolga chiqarish:

```
ism = 'Jobir'
yosh = 36
xabar = ism + ' ' + yosh + ' yoshda'
print(xabar)
```

Natija:

`TypeError: can only concatenate str (not "int") to str` Afsuski, kutilgan natija o'rniга xatolik chiqdi. Agar xatoni ingliz tilidan tarjima qilsak, matn (str) va son (int) ni jamlab bo'lmaydi degan ma'no chiqadi.

Demak, Pythonda matn (string) va son (int, float) turidagi o'zgaruvchilarni qo'shib bo'lmas ekan. Xo'sh, bunga yechim bormi? Albatta.

Pythonda bir turdagi o'zgaruvchini boshqa turga o'tkazish mumkin, bu ingliz tilida "typecasting" deyiladi. Buning uchun Pythonda maxsus funksiyalar bor, ular bilan tanishamiz:

`str()` – int yoki float turidagi sonlarni matn ko'rinishida qaytaradi.

`int()` – string yoki float ko'rinishidagi qiymatlarni butun son ko'rinishida qaytaradi. Bunda matn butun son ko'rinishida berilishi kerak.

`float()` – string yoki int ko'rinishidagi qiymatlarni o'nlik son ko'rinishida qaytaradi.

Demak, yuqoridagi kod to'g'ri ishlashi uchun 3-qatorni quyidagicha o'zgartiramiz:


```
ism = 'Jobir'
yosh = 36
xabar = ism + ' ' + str(yosh) + ' yoshda'
print(xabar)
```

Natija: Jobir 36 yoshda

① str(), int() va float() funksiyalari berilgan qiymatni boshqa ko'rinishda qaytaradi, xolos. Bunda asl o'zgaruvchining turi o'zgarмай qoladi.

O'ZGARUVCHI TURINI TEKSHIRISH

Kodimizda o'zgaruvchilar ko'payib ketdi. Yuqoridagi kabi xatolar qilmaslik uchun ba'zida o'zgaruvchining turini tekshirish talab etiladi. Buning uchun type() funksiyasidan foydalanamiz:

```
ism = 'Jobir'
yosh = 36
```

ism degan o'zgaruvchining turini konsolga chiqaramiz

```
>>> print(type(ism))
<class 'str'>
```

yosh degan o'zgaruvchining turini konsolga chiqaramiz

```
>>> print(type(yosh))
<class 'int'>
```

Kurib turganingizdek, ism nomli o'zgaruvchi str, ya'ni matn, yosh esa int, son turida ekan.

input() VA SONLAR

Avvalgi bo'limda foydalanuvchidan ma'lumot olish uchun input() funksiyasidan foydalanishni o'rgandik. Keling, endi shu funksiya yordamida foydalanuvchidan son olishni ko'rib chiqaylik. Quyidagi kod foydalanuvchining tug'ilgan yilini so'raydi va uning yoshini hisoblab beradi:

```
1. #1 foydalanuvchining tug'ilgan yilini so'raymiz
2. t_yil = input("Tug'ilgan yilingizni kiriting: ")
3. #2 foydalanuvchi yoshini hisoblaymiz
4. yosh = 2020 - t_yil #
5. #3 foydalanuvchi yoshini konsolga chiqaramiz
6. print("Siz " + yosh + " yoshda ekansiz")
```

Natija:

```
Tug'ilgan yilingizni kiriting: 1983
Traceback (most recent call last):
  File "C:\python-darslar\inputSon.py", line 4, in
<module>
    yosh = 2020 - t_yil #
TypeError: unsupported operand type(s) for -: 'int'
and 'str'
```

Kutilgan natija o'rniiga xatolik. Lekin xato qayerda? Dastur tug'ilgan yilinni so'radi, men 1983 deb kiritdim, shu zahoti xato ro'y berdi va dastur to'xtadi. Xatoni tarjima qilsak, son (int) va matn (str) o'rtasida ayirish (-) amalini bajarib bo'lmaydi degan ma'no chiqadi.

Gap shundaki, input() funksiyasi har qanday kiritilgan qiymatni matn ko'rinishida qabul qiladi (garchi biz son kiritgan bo'lsak ham). Keling, konsolda t_yil degan o'zgaruvchining turini tekshirib ko'ramiz:

```
>>> type(t_yil)
str
```

Ko'rib turganingizdek, `t_yil` o'zgaruvchisi saqlayotgan qiymat `str`, ya'ni matn ekan. Kodimizning 2- va 6-qatorlarini o'zgartiramiz:

```
1. #1 foydalanuvchining tug'ilgan yilini so'raymiz va
   qiymatni int ga aylantiramiz
2. t_yil = int(input("Tug'ilgan yilingizni kiriting: "))
3. #2 foydalanuvchi yoshini hisoblaymiz
4. yosh = 2020 - t_yil #
5. #3 foydalanuvchi yoshini konsolga chiqaramiz
6. print("Siz " + str(yosh) + " yoshda ekansiz")
```

Natija:

```
Tug'ilgan yilingizni kiriting: 1983
Siz 37 yoshda ekansiz
```

Yuqoridagi kodning 2-qatoriga e'tibor bersangiz, biz ikki funksiyani bir-biriga joylab yozdik: `int(input())`. Aslida, ularni ajratib ham yozishimiz mumkin edi:

```
1. #1 foydalanuvchining tug'ilgan yilini so'raymiz
2. t_yil = input("Tug'ilgan yilingizni kiriting: ")
3. #2 t_yil o'zgaruvchini int ga aylantiramiz
4. t_yil = int(t_yil)
5. #3 foydalanuvchi yoshini hisoblaymiz
6. yosh = 2020 - t_yil #
7. #4 foydalanuvchi yoshini konsolga chiqaramiz
8. print("Siz " + str(yosh) + " yoshda ekansiz")
```

AMALIYOT

Quyidagi dasturlarning har birini alohida fayl ko'rinishida yozing va bajaring:

1. Foydalanuvchi kiritgan sonning kvadrati va kubini konsolga chiqaruvchi dastur.

2. Foydalanuvchining yoshini so'rab, uning tug'ilgan yilini hisoblab, konsolga chiqaruvchi dastur.

3. Foydalanuvchidan ikki son kiritishni so'rab, kiritilgan sonlarning yig'indisi, ayirmasi, ko'paytmasi va bo'linmasini chiqaruvchi dastur.



<https://olicoo/Da-list>

3.6. RO'YXATLAR – LISTS

Avalgi darsimizda biz o'zgaruvchi yaratish va uning ichida biror qiymatni (matn yoki son) saqlashni o'rgandik. Bunda bitta o'zgaruvchiga bitta qiymat berdik, xolos.

Bugun o'rganadigan navbatdagi ma'lumot turi list (**ro'y-xat**) deb ataladi. Ro'yxat o'z nomi bilan bitta o'zgaruvchida bir nechta qiymatlarni saqlash imkonini beradi. Bu qiymatlar ro'yxat **elementlari** deyiladi. Ro'yxatda son, matn yoki aralash turdagi elementlarni saqlash mumkin.

List quyidagicha yaratiladi:

```
mevalar = ['olma', 'anjir', 'shaftoli', "o'rik"]
narxlar = [12000, 18000, 10900, 22000]
sonlar = ['bir', 'ikki', 3, 4, 5] # <- aralash ro'yxat
ismlar = [] # <- bo'sh ro'yxat
```

① Ro'yxat saqlaydigan o'zgaruvchilarni nomlashda -lar (ko'plik) qo'shimchasini qo'shish maqsadga muvofiq bo'ladi (ingliz tilida -s). Misol uchun: mevalar, uylar, cars, toys, books

RO'YXAT ELEMENTLARI

Ro'yxatdagi har bir element tartib bilan joylashgani sababli biz istalgan elementga uning tartib raqami (indeksi) bo'yicha murojaat etishimiz mumkin. Buning uchun avval ro'yxat nomi, keyin esa to'rtburchak qavslarda element indeksi yoziladi.

① **Dasturlash olamida indeks 0 dan boshlanadi!** Ya'ni ro'yxatdagi birinchi elementning tartib raqami (indeksi) 0 ga, ikkinchi elementning indeksi 1 ga teng va hokazo.

```
>>> mevalar = ['olma', 'anjir', 'shaftoli', "o'rik"]
>>> print("Birinchi meva: ", mevalar[0])
Birinchi meva: olma
>>> print("Ikkinchi meva: ", mevalar[1])
Ikkinchi meva: anjir
>>> print("Oxirgi meva: ", mevalar[3])
Oxirgi meva: o'rik
```

Agar list ichidagi elementlar matn ko'rinishida bo'lsa, ularga string metodlarni qo'llashimiz mumkin:

```
mevalar = ['olma', 'anjir', 'shaftoli', "o'rik"]
print("Birinchi meva: ", mevalar[0].title())
print("Ikkinchi meva: ", mevalar[1].upper())
Natija:
Birinchi meva: Olma
Ikkinchi meva: ANJIR
```

List elementlari ustida arifmetik amallar bajarish:

```
narxlar = [12000, 18000, 10900, 22000]
print(narxlar[2] + narxlar[3])
```

Natija: 32900

Pythonda Listning eng oxirgi elementiga -1 indeksi orqali ham murojaat etish mumkin. Bu usul listning uzunligini bilmaganda juda asqatadi.

```
>>> sonlar = [10, 12, 345, -23, 445, 61, -45, 56, -34]
>>> print(sonlar[-1])
-34
```

Elementlarni qo'shish, o'chirish va o'zgartirish

Dastur davomida listning tarkibi o'zgarishi, yangi elementlar qo'shilishi, ba'zi elementlar o'chirilishi tabiiy hol. Misol uchun, "Bozorlik ro'yxati" degan dasturni tasavvur qilaylik, foydalanuvchi ro'yxatga yangi mahsulotlar qo'shishi, sotib olganlarini esa o'chirishi mumkin. Keling, shunday amallarni qanday bajarishni o'rganamiz.

Elementni o'zgartirish

Ro'yxatdagi biror elementning qiymatini o'zgartirish uchun kerakli elementga indeks bo'yicha murojaat etamiz va yangi qiymat yuklaymiz.

```
narxlar = [12000, 18000, 10900, 22000]
narxlar[0] = 13000 # 1-qiymatni 13000 ga o'zgartiramiz
narxlar[2] = 11000 # 3-qiymatni 11000 ga o'zgartiramiz
narxlar[3] = narxlar[3]+2000 # 4-qiymatga 2000 qo'shamiz
print(narxlar)
```

Natija: [13000, 18000, 11000, 24000]
Mavjud bo'lmagan indeksga qiymat yuklab bo'lmaydi:

```
>>> narxlar[4] = 20000
IndexError: list assignment index out of range
```

Yangi element qo'shish

Ro'yxatga element qo'shishning ikki usuli bor, quyida ular bilan tanishamiz.

Birinchi usul: `.append()` metodi yordamida ro'yxatning oxiriga qiymat qo'shish:

```
mevalar = ['olma', 'anjir', 'shaftoli', "o'rik"]
mevalar.append("tarvuz") # mevalar ga tarvuz qo'shamiz
print(mevalar)
```

Natija: ['olma', 'anjir', 'shaftoli', "o'rik", 'tarvuz']
`.append()` metodi bo'sh ro'yxatni to'ldirishda juda qulay. Odatda, dastur boshida bo'sh ro'yxat yaratilib, dastur davomida ro'yxat foydalanuvchi tomonidan to'ldirib borilishi odatiy hol.

```
cars = [] # bo'sh ro'yxat yaratamiz
cars.append('Lacetti') # ro'yxatga lacetti qo'shamiz
cars.append('Nexia 3') # ro'yxatga Nexia 3 qo'shamiz
cars.append('Cobalt') # ro'yxatga Cobalt qo'shamiz
print(cars)
```

Natija: ['Lacetti', 'Nexia 3', 'Cobalt']

Ro'yxatning istalgan joyiga yangi element qo'shish uchun `.insert()` metodidan foydalanamiz. `.insert()` metodi ichida yangi elementning indeksi va qiymati beriladi:

```
>>> cars = ['Lacetti', 'Nexia 3', 'Cobalt']
>>> print(cars)
['Lacetti', 'Nexia 3', 'Cobalt']
>>> cars.insert(0, 'Malibu')
>>> print(cars)
['Malibu', 'Lacetti', 'Nexia 3', 'Cobalt']
>>> cars.insert(2, 'Damas')
>>> print(cars)
['Malibu', 'Lacetti', 'Damas', 'Nexia 3', 'Cobalt']
```

Elementni o'chirish

Ro'yxatdan biror elementni olib tashlash uchun uning indeksini yoki qiymatini bilishimiz lozim.

Elementni indeks yordamida olib tashlash uchun `del` operatoridan foydalanamiz:

```
mevalar = ['olma', 'anjir', 'shaftoli', "o'rik", 'anor']
del mevalar[1] # 2-element (anjir) ni o'chirib tashlaymiz
print(mevalar)
```

Natija: ['olma', 'shaftoli', "o'rik", 'anor']
Elementning qiymati bo'yicha o'chirish uchun esa `.remove(qiymat)` metodidan foydalanamiz. Buning uchun qavs ichida o'chirib tashlash kerak bo'lgan qiymatni yozamiz.

```
mevalar = ['olma', 'anjir', 'shaftoli', "o'rik", 'anor']
mevalar.remove('shaftoli') # Ro'yxatdan shaftolini o'chirdik
print(mevalar)
```

Natija: ['olma', 'anjir', "o'rik", 'anor']

`.remove(qiymat)` metodi ro'yxatda uchragan birinchi mos keluvchi qiymatni o'chiradi. Agar ro'yxatning ichida 2 va undan ko'p bir xil qiymatli elementlar bo'lsa, ulardan faqat birinchisi o'chadi.

```
hayvonlar = ['it', 'mushuk', 'sigir', 'quyon', 'mushuk']
hayvonlar.remove("mushuk")
print(hayvonlar)
```

Natija: ['it', 'sigir', 'quyon', 'mushuk']

Elementni sug'urib olish

Ba'zida biror elementni butunlay o'chirib tashlash emas, balki uni ro'yxatdan sug'urib olib, foydalanish talab qilinadi. Buning uchun `Pythonda .pop()` metodidan foydalanamiz.


```
bozorlik = ["yog", 'un', 'piyoz', 'banan', "go'sht"]
mahsulot = bozorlik.pop(3) # 4-elementni sug'urib olamiz
print("Men " + mahsulot + " sotib oldim")
print("Olimagan mahsulotlar: ", bozorlik)
```

Natija:

```
Men banan sotib oldim
Olimagan mahsulotlar: ["yog", 'un', 'piyoz', "go'sht"]
```

Agar .pop() metodida indeks berilmasa, ro'yxatdan oxirgi qiymat sug'urib olinadi.

```
>>> numbers = [1, 2, 3, 4, 5]
>>> print(numbers)
[1, 2, 3, 4, 5]
>>> numbers.pop()
>>> print(numbers)
[1, 2, 3, 4]
```

AMALIYOT

Quyidagi mashqlarni bajaring:

- ismlar degan ro'yxat yarating va kamida 3 ta yaqin do'stingizning ismini kiriting
- Ro'yxatdagi har bir do'stingizga qisqa xabar yozib konsolga chiqaring:

```
Salom Ali ishlaring yaxshimi?
Hasan va Husan egizaklar
G'ani g'ildirakni g'izillatib g'ildiratdi
```

- sonlar deb nomlangan ro'yxat yarating va ichiga turli sonlarni yuklang (musbat, manfiy, butun, o'nlik).
- Yuqoridagi ro'yxatdagi sonlar ustida turli arifmetik amallar bajarib ko'ring. Ro'yxatdagi ba'zi sonlarning qiymatini o'zgartiring, Ba'zilarini esa almashiting.

- t_shaxslar va z_shaxslar degan 2 ta ro'yxat yarating va biriga o'zingiz eng ko'p hurmat qilgan tarixiy shaxslarning, ikkinchisiga esa zamonamizdagi tirik bo'lgan shaxslarning ismini kiriting.
- Yuqoridagi ro'yxatlarning har biridan bittadan qiymatni sug'urib olib (.pop()), quyidagi ko'rinishda chiqaring:

```
Men tarixiy shaxslardan Imom Buxoriy bilan,
zamonaviy shaxslardan esa Bill Gates bilan
subhat qilishni istar edim
```

- friends nomli bo'sh ro'yxat tuzing va unga .append() yordamida 5-6 ta mehmonga chaqirmoqchi bo'lgan do'stlaringizni kiriting.
- Yuqoridagi ro'yxatdan mehmonga kela olmaydigan odamlarni .remove() metodi yordamida o'chirib tashlang.
- Ro'yxatning oxiriga, boshiga va o'rtasiga yangi ismlar qo'shing.
- Yangi mehmonlar deb nomlangan bo'sh ro'yxat yarating. .pop() va .append() metodlari yordamida mehmonga kelgan do'stlaringizning ismini friends ro'yxatidan sug'urib olib, mehmonlar ro'yxatiga qo'shing.

3.7. RO'YXATLAR BILAN ISHLASH

Ro'yxatni tartiblash

Aksar holatlarda ro'yxat ichidagi elementlarni alifbo ketma-ketligida tartiblash talab qilinishi mumkin. Buning uchun maxsus .sort() metodidan foydalanamiz.



```
cars = ['bmw', 'volvo', 'gm', 'tesla', 'audi']
cars.sort()
print(cars)
```

Natija: ['audi', 'bmw', 'gm', 'tesla', 'volvo']

△ Tartiblashda bosh harflar kichik harflardan avval kelishini hisobga oling. Agar ro'yxatdagi so'zlarning birinchi harfi katta-kichik aralash yozilgan bo'lsa, ularni bir ko'rinishga keltirib olish maqsadga muvofiq bo'ladi.

```
cars = ['bmw', 'mercedes', 'volvo', 'gm', 'tesla', 'audi']
cars.sort()
print(cars)
```

Natija: ['GM', 'audi', 'bmw', 'mercedes', 'tesla', 'volvo']

Yuqoridagi misolda GM elementi bosh harf bilan boshlangani uchun ro'yxatning boshidan joy oldi.

Ro'yxatni teskari tartiblash uchun .sort() metodi ichida reverse=True argumentini ham yozamiz.

```
cars = ['bmw', 'mercedes', 'volvo', 'gm', 'tesla', 'audi']
cars.sort(reverse=True)
print(cars)
```

Natija: ['volvo', 'tesla', 'mercedes', 'gm', 'bmw', 'audi']

.sort() metodi bevosita berilgan ro'yxatni tartiblaydi (ya'ni ichki elementlarining joylashuvini o'zgartiradi).

Ba'zi holatlarda asl ro'yxat ichidagi elementlarning ketma-ketligini buzmag holda, ro'yxat tarkibini tartiblangan ko'rinishida chiqarish talab qilinishi mumkin. Buning uchun sorted() funksiyasidan foydalanamiz:

```
mehmonlar = ['Odil', 'Hamid', 'Temur', 'Avazbek', 'Farrux']
print("sorted() qaytargan ro'yxat:", sorted(mehmonlar))
print("Asl ro'yxat o'zgarmas qoldi:", mehmonlar)
```

Natija:

```
sorted() qaytargan ro'yxat: ['Avazbek', 'Farrux', 'Hamid', 'Odil', 'Temur']
Asl ro'yxat o'zgarmas qoldi: ['Odil', 'Hamid', 'Temur', 'Avazbek', 'Farrux']
```

sorted() funksiyasi yordamida teskari tartiblash uchun ham reverse=True argumentini ishlatamiz:

```
>>> print(sorted(mehmonlar, reverse=True))
['Temur', 'Odil', 'Hamid', 'Farrux', 'Avazbek']
```

Yuqoridagi ikki usul bilan sonli ro'yxatlarni ham tartiblashimiz mumkin:

```
ages = [12, 98, 34, 65, 34, 76, 11]
ages.sort()
print(ages)
print(sorted(ages, reverse=True))
```

Natija:

```
[11, 12, 34, 34, 65, 76, 98]
[98, 76, 65, 34, 34, 12, 11]
```

RO'YXATNI AYLANTIRISH

Ba'zida ro'yxatni (boshini oxiriga, oxirini boshiga qilib) aylantirish talab qilinishi mumkin. Buning uchun .reverse() metodidan foydalanamiz.

```
fruits = ['pear', 'banana', 'apple', 'watermelon', 'lemon']
fruits.reverse()
print(fruits)
```

Natija: ['lemon', 'watermelon', 'apple', 'banana', 'pear']
Natija va asl ro'yxatni solishtiring.

RO'YXATNING UZUNLIGINI TOPISH

Ro'yxatning uzunligi, ya'ni uning ichidagi elementlar sonini aniqlash uchun `len()` funksiyasidan foydalanamiz:

```
fruits = ['pear', 'banana', 'apple', 'watermelon', 'lemon']
print("Elementlar soni:", len(fruits))
```

Natija: Elementlar soni: 5

range() FUNKSIYASI

Bu funksiya yordamida biz ma'lum oralqdagi sonlar ketma-ketligini yaratishimiz mumkin. `list()` funksiyasi yordamida esa bu oralqni ro'yxat shaklida saqlab olamiz:

```
sonlar = list(range(0,10))
print(sonlar)
```

Natija: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Yuqoridagi misolda `range(0,10)` funksiyasi 0 dan 9 gacha sonlar ketma-ketligini shakllantirdi, `list()` funksiyasi esa bu ketma-ketlikni ro'yxatga aylantirdi.

⚠ E'tibor qiling, `range()` funksiyasi ikkinchi indeksdan bitta avval to'xtaydi.

`range()` yordamida qadamni ham berishimiz mumkin:

```
juft_sonlar = list(range(0,20,2)) # 0 dan 20 gacha 2
qadam bilan
toq_sonlar = list(range(1,20,2)) # 1 dan 20 gacha 2
qadam bilan
print("Juft sonlar:", juft_sonlar)
print("Toq sonlar:", toq_sonlar)
```

Natija:

```
Juft sonlar: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
Toq sonlar: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

Agar sonlar ketma-ketligi 0 dan boshlansa, `range()` funksiyasida ikkinchi argumentni ko'rsatish kifoya. Misol uchun, `range(0,10)` emas, `range(10)` deb yozsak ham bo'laveradi.

SONLI RO'YXAT USTIDA SODDA AMALLAR

Pythonda ro'yxatlar ustida ba'zi soddami amallarni ham bajarish mumkin. Masalan, ro'yxatdagi eng kichik sonni topish uchun `min()` funksiyasidan, eng katta sonni topish uchun `max()` funksiyasidan, sonlarning yig'indisini topish uchun esa `sum()` funksiyasidan foydalanamiz:

```
narxlar = [12000, 22500, 23456, 9800, 5600, 9934, 32874]
arzon = min(narxlar)
qimmat = max(narxlar)
jami = sum(narxlar)
print("Eng arzon narx ", arzon, \
      ". Eng qimmat ", qimmat, \
      ". Jami: ", jami)
```

Natija:

Eng arzon narx 5600. Eng qimmat 32874. Jami: 116164

Ⓢ Agar bir qatorga yozilayotgan kodingiz juda uzun bo'lib ketsa, belgisi yordamida kodni yangi qatordan davom ettirishingiz mumkin.

RO'YXATNI KESISH

Ba'zida ro'yxatning ma'lum bir bo'lagini ajratib olish talab qilinishi mumkin. Deylik, biz quyidagi cars degan ro'yxatdan birinchi 3 ta elementni ajratib olmoqchimiz, buning uchun boshlang'ich va oxirgi indekslarni beramiz:

```
cars = ['bmw', 'volvo', 'gm', 'toyota', 'tesla', 'audi']
my_cars = cars[0:3] # 0 dan boshlab 3 ta element ajratib olamiz
print(my_cars)
```

Natija: ['bmw', 'volvo', 'gm']

△ Python ro'yxatni kesishda 2-ko'rsatilgan indeksdan bitta avval to'xtaydi. Yuqoridagi misolda ham 0, 1, 2-elementlar ajratib olindi.

Bu usul bilan ro'yxatning istalgan joyidan bo'lishimiz mumkin:

```
>>> print(cars[2:5]) # 2-3-4-elementlarni ajratib olamiz
['gm', 'toyota', 'tesla']
```

Agar boshlang'ich indeksni bermasangiz, Python avtomat ravishda 0-indeksdan boshlab kesadi. Agar 2-indeksni kiritmasangiz, ro'yxat oxirigacha kesadi:

```
>>> print(cars[:4]) # Ro'yxat boshidan 4 ta element ajratish
['bmw', 'volvo', 'gm', 'toyota']
>>> print(cars[2:]) # 2-elementdan ro'yxat oxirigacha kesish
['gm', 'toyota', 'tesla', 'audi']
```

RO'YXATDAN NUSXA OLISH

Dastur davomida biror ro'yxatdan nusxa olish talab qilinishi mumkin. Buning uchun tenglik (=) belgisidan foydalansak bo'ladimi? Quyidagi misolga e'tibor bering:

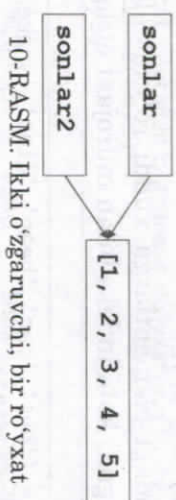
```
sonlar = [1, 2, 3, 4, 5] # sonlar degan ro'yxat yaratamiz
sonlar2 = sonlar # sonlar2 degan ro'yxatni sonlarga tenglaymiz
sonlar2.append(6) # sonlar2 ga 6 sonini qo'shamiz
sonlar2.append(7) # sonlar2 ga 7 sonini qo'shamiz
print("Bu sonlar ro'yxati:", sonlar)
print("Bu sonlar2 ro'yxati:", sonlar2)
```

Natija:

```
Bu sonlar ro'yxati: [1, 2, 3, 4, 5, 6, 7]
Bu sonlar2 ro'yxati: [1, 2, 3, 4, 5, 6, 7]
```

Natija biz kutgandek chiqdimi? Yo'q. Biz 6 va 7 sonlarini sonlar2 degan ro'yxatga qo'shgan edik, lekin bu ikki son sonlar degan asl ro'yxatga ham qo'shib qoldi. Nima uchun?

Gap shundaki, sonlar2=sonlar deb yozgan komandamiz yangi ro'yxat yaratish o'rniiga ikkala o'zgaruvchini ham xotiradagi bitta ro'yxatga bog'lab (yo'naltirib) qo'ydi. Ya'ni bitta ro'yxat ikki nomga ega bo'lib qoldi. Endi sonlar yoki sonlar2 ustida bajarilgan amallarimiz xotiradagi bitta ro'yxat ustida bajarilaveradi.



Unda qanday qilib ro'yxatdan nusxa olamiz? Buning uchun yuqoridagi kabi ro'yxatni kesish usulidan foydalanamiz. Ya'ni ro'yxatning boshidan oxirigacha "kesib" yangi ro'yxatga yuklaymiz. Bunda kvadrat qavs ichida boshlang'ich va yakuniy indeksni ko'rsatish shart emas.

```
sonlar = [1, 2, 3, 4, 5] # donlar degan ro'yxat yaratamiz
sonlar2 = sonlar[:] # [:] ro'yxatni to'liq ko'chirib oladi
sonlar2.append(6) # sonlar2 ga 6 sonini qo'shamiz
sonlar2.append(7) # sonlar2 ga 7 sonini qo'shamiz
print("Bu sonlar ro'yxati:", sonlar)
print("Bu sonlar2 ro'yxati:", sonlar2)
```

Natija:

```
Bu sonlar ro'yxati: [1, 2, 3, 4, 5]
Bu sonlar2 ro'yxati: [1, 2, 3, 4, 5, 6, 7]
```

3.8. TUPLES – O'ZGARMAS RO'YXATLAR

Dastur yaratish davomida o'zgarmas ro'yxat tuzish talab qilinishi mumkin. Pythonda bunday ro'yxatlar **tuples** deb yuritiladi. Tuple ichidagi qiymatlar bir marta, dastur boshida beriladi, so'ngra o'zgartirib bo'lmaydi. Listdan farqli ra-

vishta, Tuple e'lon qilishda kvadrat qavslar [] o'rni oddiy qavslar () ishlatiladi:

```
tomonlar = (20, 30, 55.2)
print(tomonlar)
```

Natija: (20, 30, 55.2)

Tuple ichidagi elementlarga xuddi ro'yxat elementlariga murojaat etilgani kabi indeks bilan murojaat etilaveradi:

```
>>> toys = ('bus', 'car', 'bear', 'dino', 'snake', 'lizard')
>>> print(toys[0])
bus
>>> print(toys[-1])
lizard
>>> print(toys[2:5])
('bear', 'dino', 'snake')
```

Keling, Tuple ichidagi biror elementning qiymatini o'zgartirib ko'ramiz:

```
toys = ('bus', 'car', 'bear', 'dino', 'snake', 'lizard')
toys[3] = 'dragon'
```

Natija:

TypeError: 'tuple' object does not support item assignment

Ko'rib turganingizdek, bu operatsiya xatolikka olib keldi. Shu kabi o'zgarmas ro'yxatdan biror elementni o'chirish yoki yangi element qo'shish ham mumkin emas.

Agar Tuple ga o'zgartirish talab qilinsa, yagona yo'li o'zgar-mas ro'yxatni list() funksiyasi yordamida oddiy ro'yxat ko'rinishiga keltirib olish, o'zgarishlarni bajarish va ro'yxatni qaytarib tuple() funksiyasi yordamida o'zgarmas ro'yxatga konvertatsiya qilishdir:

```
# o'zgarmas ro'yxat yaratamiz
toys = ('bus', 'car', 'bear', 'dino', 'snake', 'lizard')
# o'zgarmas ro'yxatni oddiy ro'yxatga aylantiramiz
toys = list(toys) (list)
# Ro'yxatga o'zgartirishlar kiritamiz
toys.append('dragon')
toys.remove('bus')
toys[1] = 'mcqueen'
# Ro'yxatni qaytadan o'zgarmas ro'yxatga (Tuple) aylantiramiz
toys = tuple(toys)
print(toys)
```

Natija:

('car', 'mcqueen', 'dino', 'snake', 'lizard', 'dragon')

AMALIYOT

- O'zingizga ma'lum davlatlarning ro'yxatini tuzing va ro'yxatni konsolga chiqaring.
 - o Ro'yxatning uzunligini konsolga chiqaring.
 - o sorted() funksiyasi yordamida ro'yxatni tartiblangan holda konsolga chiqaring.
 - o sorted() yordamida ro'yxatni teskari tartibda konsolga chiqaring.
 - o Asl ro'yxatni qaytadan konsolga chiqaring.
 - o reverse() metodi yordamida ro'yxatni ortidan boshlab chiqaring.
 - o sort() metodi yordamida ro'yxatni avval alifbo bo'yicha, keyin esa alifboga teskari tartibda konsolga chiqaring.
- 120 dan 1200 gacha bo'lgan juft sonlar ro'yxatini tuzing.
 - o Ro'yxatdagi sonlar yig'indisini hisoblang va konsolga chiqaring.
 - o Ro'yxatdagi eng katta va eng kichik son o'rtasidagi ayirmani hisoblang va konsolga chiqaring.
 - o Ro'yxatdagi elementlar sonini hisoblang.

- Ro'yxatning boshidan, o'rtasidan va oxiridan 20 ta qiymatni konsolga chiqaring.
- taomlar degan ro'yxat yaratib va ichiga istalgan 5 ta taomni kiritib.
- nonushta degan yangi ro'yxatga taomlar dan nusxa oling.
- Yangi ro'yxatda faqat nonushtaga yeyiladigan taomlarni qoldirib va yana 2 ta taom qo'shing.
- Ikkala ro'yxatni ham (taomlar va nonushta) konsolga chiqaring.
- Yuqoridagi nonushta ro'yxatini o'zgarmas ro'yxatga aylantirib va nonushta[0] = "qaymoq va non" deb qiymat berib ko'ring.

3.9. for SIKLI

Dasturlash davomida kodimizning biror qismini bir necha bor takrorlash talab etilishi mumkin. Misol uchun, ro'yxat ichidagi har bir elementni alohida qatordan konsolga chiqarish yoki bo'lmasa har bir elementni kvadratga oshirish va hokazo.

Mana shunday vaziyatda bizga `for` operatori yordam beradi. Bu operator yordamida biz kodlarni qayta-qayta takrorlashimiz mumkin. Dasturlashda bu **sikl (loop)** deb ataladi.

Quyidagi misolni ko'ramiz. Bizda mehmonlar ro'yxati bor, har bir mehmonning ismini yangi qatordan chiqarmoqchimiz. Buning uchun quyidagi kodni yozamiz:

```
1. mehmonlar = ['Ali', 'Vali', 'Hasan', 'Husan', 'Olim']
2. for mehmon in mehmonlar:
3.     print(mehmon)
```



<https://clicco/Da-for>

Natija:

```
Ali
Vali
Hasan
Husan
Olim
```

Kodni tahlil qilaylik:

1-qatorda biz mehmonlar degan ro'yxat yaratdik va uni mehmonlarning ismi bilan to'ldirdik.

2-qatorda `for` siklini bohladik. Bu qator Pythonga mehmonlar degan ro'yxatdan har bir elementini olib, uni yangi, mehmon degan o'zgaruvchiga yuklashni buyuryapti (o'zgaruvchiga istalgan nom berishingiz mumkin. Biz tushunarli bo'lishi uchun mehmon deb atadik).

3-qatorda mehmon degan o'zgaruvchining qiymatini konsolga chiqardik. Bu sikl to mehmonlar ro'yxatida elementlar tugagunga qadar takrorlanadi.

① "For" so'zi ingliz tilidan "uchun" deb tarjima qilinadi.

Yuqoridagi kodni oddiy tilga tarjima qilsak, mehmonlar ro'yxatidagi har bir mehmon **uchun** uning ismini konsolga chiqar degan ma'no kelib chiqadi.

for SIKLI QANDAY ISHLAYDI

Keling, yana bir misolga e'tibor beraylik.

```
mehmonlar = ['Ali', 'Vali', 'Hasan', 'Husan', 'Olim']
for mehmon in mehmonlar:
    print(f"Hurmatli {mehmon},")
    print("Sizni 20-Mart kuni nahorgi oshga taklif qilamiz.")
    print("Hurmat bilan, Palonchiyevlar oilasi")
```


Natija:

```
Hurmatli Ali,  
Sizni 20-Mart kuni nahorgi oshga taklif qilamiz.  
Hurmat bilan, Palonchiyevlar oilasi  
Hurmatli Vali,  
Sizni 20-Mart kuni nahorgi oshga taklif qilamiz.  
Hurmat bilan, Palonchiyevlar oilasi  
Hurmatli Hasan,  
Sizni 20-Mart kuni nahorgi oshga taklif qilamiz.  
Hurmat bilan, Palonchiyevlar oilasi  
Hurmatli Husan,  
Sizni 20-Mart kuni nahorgi oshga taklif qilamiz.  
Hurmat bilan, Palonchiyevlar oilasi  
Hurmatli Olim,  
Sizni 20-Mart kuni nahorgi oshga taklif qilamiz.  
Hurmat bilan, Palonchiyevlar oilasi
```

Yuqoridagi kodda 2-qator sikl sharti deyiladi. Aynan shu qator kodimiz necha marta takrorlanishini belgilaydi. Bizning holatimizda sikl mehmonlar ro'yxati ichidagi elementlar tugagunga qadar takrorlanadi. Sikl sharti ikki nuqta (:) bilan tugaydi. Undan keyingi 3 qator siklning badani deyiladi.

```
1 mehmonlar = ['Ali', 'Vali', 'Hasan', 'Husan', 'Olim']  
2 for mehmon in mehmonlar:   
3     print(f"Hurmatli {mehmon},")   
4     print("Sizni 20-Mart kuni nahorgi oshga taklif qilamiz.")   
5     print("Hurmat bilan, Palonchiyevlar oilasi")   
6     print("Dastur tugadi!")   
7
```

→ Tsikl sharti

→ Tsikl badanidagi kod

→ Tsikl tashqarisidagi kod

Sikl badani surilish (indentation) bilan ajratiladi, ya'ni siklning takrorlanuvchi qismi asosiy koddan birmuncha o'ngroqqa surilgan bo'ladi. Agar biz mana shu surilishni tark etsak, kodimiz xato beradi:

```
mehmonlar = ['Ali', 'Vali', 'Hasan', 'Husan', 'Olim']  
for mehmon in mehmonlar:  
    print(f"Hurmatli {mehmon},")  
    print("Sizni 20-Mart kuni nahorgi oshga taklif qilamiz.")  
    print("Hurmat bilan, Palonchiyevlar oilasi")
```

Natija:

IndentationError: expected an indented block
Ko'rib turganingizdek, for dan keyingi qatorni o'ngga surmaganimiz uchun **indentation error** (surishda xatolik) degan xabarni oldik.

Sikl shartidan so'ng o'ngga surib yozilgan barcha qatorlar sikl badanida deb hisoblanadi (garchi qatorlar orasida joy tashlab yozsak ham).

```
for mehmon in mehmonlar:  
    print(f"Hurmatli {mehmon},")  
    print("Sizni 20-Mart kuni nahorgi oshga taklif qilamiz.")  
    # keyingi qator ham sikl ichida hisoblanadi  
    print("Hurmat bilan, Palonchiyevlar oilasi")
```

Sikl badanidan chiqish uchun kodni yana qator boshidan yozish kifoya:

```
cars = ['nexia', 'tico', 'damas']  
for car in cars: # <- sikl sharti  
    print(car.title()) # <- sikl badani  
    print("Ko'rganlar qilar havasi!") # <- sikldan keyingi kod
```

Natija:

```
Nexia  
Tico  
Damas  
Ko'rganlar qilar havasi!
```

Agar yuqoridagi kodimizda 4-qatorni surmasdan yozsak, Python bu qatorni ham siklning ichida deb qabul qiladi va qayta-qayta takrorlaydi:

```
cars = ['nexia', 'tico', 'damas']
for car in cars:
    print(car.title())
print("Ko'rganlar qilar havasi!")
```

Natija:

```
Nexia
Ko'rganlar qilar havasi!
Tico
Ko'rganlar qilar havasi!
Damas
Ko'rganlar qilar havasi!
```

for YORDAMIDA SONLI RO'YXATLAR BILAN ISHLASH

Ushbu misolni ko'zdan kechiramiz

```
sonlar = list(range(1,11))
for son in sonlar:
    print(f"{son} ning kvadrati {son**2} ga teng")
```

Natija:

```
1 ning kvadrati 1 ga teng
2 ning kvadrati 4 ga teng
3 ning kvadrati 9 ga teng
4 ning kvadrati 16 ga teng
5 ning kvadrati 25 ga teng
6 ning kvadrati 36 ga teng
7 ning kvadrati 49 ga teng
8 ning kvadrati 64 ga teng
9 ning kvadrati 81 ga teng
10 ning kvadrati 100 ga teng
```

for yordamida yangi ro'yxat ham shakllantirish mumkin.

Quyidagi dasturda avval 1 dan 10 gacha sonlar ro'yxatini yaratib olaymiz. Keyin esa sonlar kvadratini joylash uchun bo'sh

ro'yxat yaratdik. for sikli ichida esa har bir sonning kvadratini hisoblab, sonlar_kvadrati ro'yxatiga qo'shib boryapmiz.

```
sonlar = list(range(11)) # 1 dan 10 gacha sonlar ro'yxati
sonlar_kvadrati = [] # bo'sh ro'yxat
for son in sonlar:
    sonlar_kvadrati.append(son**2)
print(sonlar)
print(sonlar_kvadrati)
```

Natija:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

for va input()

for operatori va input() funksiyasi yordamida foydalanuvchidan bir nechta qiymatlar qabul qilish va olingan qiymatlarni ro'yxatga joylash mumkin:

```
1. dostlar = [] # bo'sh ro'yxat
2. print("5 ta eng yaqin do'stingiz kim?")
3. for n in range(5):
4.     dostlar.append(input(f"{n+1}-isini kiriting: "))
5. print(dostlar)
```

Natija:

```
5 ta eng yaqin do'stingiz kim?
1-ismini kiriting: Ali
2-ismini kiriting: Vali
3-ismini kiriting: Hasan
4-ismini kiriting: Husan
5-ismini kiriting: Ivan
['Ali', 'Vali', 'Hasan', 'Husan', 'Ivan']
```


Kodni tahlil qilamiz:

1-qatorda bo'sh do'stlar ro'yxatini yaratdik.

2-qatorda ekranga "5 ta eng yaqin do'stingiz kim?" degan xabarni chiqardik.

3-qatorda siklni boshladik. range(5) funksiyasi 0 dan 5 gacha sonlar ketma-ketligini yaratadi, (0,1,2,3,4) sikl esa n shularning har biriga teng bo'lib chiqquncha davom etadi.

4-qatorda sikl badani kelgan. Bu yerda biz foydalanuvchidan n+1 do'stingizni kiriting deb so'radik. Nima uchun n+1 (n emas)? Sababi, n 0 dan 4 gacha qiymatlarni oladi, foydalanuvchiga tushunarli bo'lishi uchun esa biz "0-ismni kiriting: " deb emas, balki n+1, ya'ni 1-ismni kiriting deb murojaat etyapmiz.

5-qatorda shakllangan ro'yxatni konsolga chiqardik.

for sikli har qanday dasturlash tilining muhim qismlaridan hisoblanadi va biz bu operatorga hali takror-takror qaytamiz.

AMALIYOT

- Kamida 5 elementdan iborat ismlar degan ro'yxat tuzing va ro'yxatdagi har bir ismga takrorlanuvchi xabar yozing.
- Yuqoridagi sikl tugaganidan so'ng ekranga "Kod n marta takrorlandi" degan xabarni chiqaring (n o'rniga kod nechta marta takrorlanganini yozing).
- 10 dan 100 gacha bo'lgan toq sonlar ro'yxatini tuzing. Ro'yxatning har bir elementi kubini yangi qatordan konsolga chiqaring.
- Foydalanuvchidan 5 ta eng sevimli kinolarini kiritishni so'rang va kinolar degan ro'yxatga saqlab oling. Natijani konsolga chiqaring.
- Foydalanuvchidan bugun nechta odam bilan uchrashganini (subbatlashganini) so'rang va har bir subbatlashgan odamning ismini birma-bir so'rab ro'yxatga yozing. Ro'yxatni konsolga chiqaring.

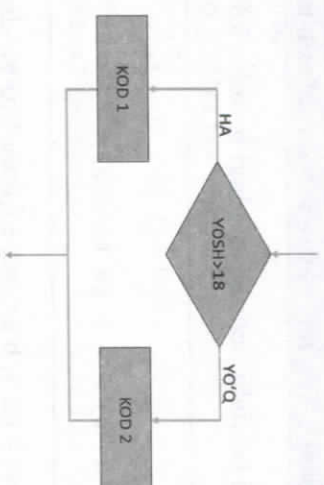
4-BOB

SHARTLAR VA TARMOQLANISH

Shu vaqtgacha yozgan dasturlarimizga e'tibor bersangiz, dasturimiz yuqoridan pastga qarab qatorma-qator bajarilib keldi. Bu chiziqli dastur deyiladi. Vogelikda esa aksar dasturlar ma'lum bir shart bajarilishiga (yoki bajarilmaganiga) ko'ra kodning bir qismidan boshqa qismiga "sakrab" o'tishi tabiiy hol. Dasturlashda bu **tarmoqlanish** deb ataladi.

Misol uchun, ijtimoiy tarmoqlar yoshingizdan kelib chiqqan holda sizga mos mavzularni ko'rsatishi tarmoqlanishga misol bo'ladi. Yoki ba'zi internet sahifalar sizga sahifaning tilini tanlash imkonini beradi (Rus/O'zbek/Ingliz) va tanlovingizga qarab turli yo'nalishlarda davom etadi.

Dastur tarmoqlanishi uchun avval ma'lum bir shartlar bajarilishi kerak (yosh>7, til=0'zbek, jins=Ayol va hokazo). Ushbu bobda mana shunday shartlarni yozish, tekshirish va shart bajarilishi (yoki bajarilmaganiga) ko'ra dasturni tarmoqlashni o'rganamiz.



12-RASM. Tarmoqlanishga misol

4.1. TAQQOSLASH OPERATORLARI

Dastur tarmoqlanishi uchun, avvalo, biror shart bajarilishi kerak. Shart bajarilishini tekshirish uchun esa maxsus taqqoslash operatorlari mavjud. Bu operatorlar yordamida biz o'zgaruvchilarning ma'lum qiymatga tengligini yoki, aksincha, teng emasligini yoxud ifodalarning natijasi berilgan qiymatdan katta yoki kichikligini va hokazo shartlarni tekshirib ko'rishimiz mumkin.

Quyidagi jadvalda Pythondagi taqqoslash operatorlari keltirilgan.

Taqqoslash operatori	Ma'nosi	Ishlatilishi
==	Tenglik	a==b
!=	Tengsizlik	a!=b
<	Kichik	a<b
<=	Kichik yoki teng	a<=b
>	Katta	a>b
>=	Katta yoki teng	a>=b

Taqqoslash operatorlari mantiqiy qiymat qaytaradi. Mantiqiy qiymatlar True (shart bajarildi) yoki False (shart bajarilmadi) bo'lishi mumkin.

① True ingliz tilidan "To'g'ri", False esa "Yolg'on" deb tarjima qilinadi.

Tushunarli bo'lishi uchun quyidagi misolni ko'ramiz:

```
>>> a=5
>>> b=6
>>> print(a==b) # a b ga teng(mi)?
False
```

Yuqorida biz a ga 5, b ga esa 6 qiymatini yukladik. 3-qator-da bu ikki o'zgaruvchining bir-biriga tengligini tekshirgani-

mizda (a==b) taqqoslash operatori False qiymatini qaytardi (ya'ni tenglik bajarilmadi).

Yana bir misol ko'ramiz:

```
>>> a=5
>>> b=6
>>> print(a==(b-1))
True
```

Bu safar taqqoslashning o'ng tarafga b-1 ifodasini yozdik, b-1=5 bo'lgani uchun tenglik bajarildi va taqqoslash operatori True qiymatini qaytardi. Demak, mana shu yo'sinda biz turli ifodalarni taqqoslashimiz mumkin ekan.

Taqqoslash, odatda, bir turdagi ma'lumotlar o'rtasida bo'ladi. Misol uchun, matnlarni matn bilan, sonlarni sonlar bilan va hokazo.

```
>>> print('anvar'=='Anvar') # matnlarni taqqoslash
False
>>> print(10**2<2**10) # sonlarni taqqoslash
True
```

Taqqoslash qanday ishlashini tushunish uchun quyidagi mashqlarni Pythonda bajarib ko'ring va natijasini tahlil qiling:

```
ism = 'ahad'
print(ism != 'Ahad')
print(ism.title() == 'Ahad')
nums1 = [1, 2, 3]
nums2 = [1, 2, 4]
print(nums1 == nums2)
print(9**2 >= 7*9+18)
x=10
print(x*x<'x**2')
print(x*x>float(f"{x**2}"))
```


4.2. if-else OPERATORI

Avvalgi bo'limda biz turli ifodalarni taqdoslashni o'rgandik, keling, endi taqdoslashning natijasiga ko'ra tarmoqlanishini ham ko'zdan kechiraylik. Buning uchun dasturlash tillarida maxsus if operatori mavjud. "if" so'zi ingliz tilidan "agar" deb tarjima qilinadi va ma'lum bir shart qanoatlantirishiga qarab shu shartga bog'langan kod ham bajariladi.

<https://olicoo/Da-if-else>



```
1. son = int(input("Istalgan son kiriting: "))
2. if son>0:
3.     print(son, "musbat son")
```

Yuqoridagi dasturni tahlil qilamiz:

1-qatorda biz foydalanuvchidan istalgan son kiritishni so'raymiz.

2-qatorda if operatori yordamida kiritilgan sonning 0 dan katta ekanini tekshiramiz. Bu qatorni "agar son 0 dan katta bo'lsa" deb o'qish mumkin.

3-qator if operatorining badani hisoblanadi va faqatgina yuqoridagi shart to'g'ri (True) bo'lsa, bajariladi.

Keling, endi dasturni bajaramiz:

```
Istalgan son kiriting: 5
5 musbat son
```

Biz 5 sonini kiritdik. 5>0 bo'lgani uchun kodning 3-qatori bajarildi va "5 musbat son" degan yozuv konsulga chiqdi.

Dasturni qayta bajaramiz, bu safar endi manfiy son kiritib ko'ramiz:

```
Istalgan son kiriting: -8
```

Ko'rib turganingizdek, son>0 ifodasi False qaytargani uchun 3-qatordagi kod bajarilmadi.

Keling, yana bir misolga e'tibor qaratamiz va yangi else operatori bilan tanishamiz. "else" so'zi ingliz tilidan "aks holda" deb tarjima qilinadi va if operatorida berilgan shart qanoatlantirilmasa, else operatoridan keyingi kod bajariladi.

```
yosh = int(input("Yoshingizni kiriting: "))
if yosh<=7: # agar yosh 7 dan kichik yoki teng bo'lsa
    print("Sizga avtobus bepul")
else: # aks holda ya'ni, yosh>7
    print("Chipta narxi 5000 so'm")
```

Yuqoridagi dasturda foydalanuvchi yoshini so'raymiz. Agar foydalanuvchi 7 yoshdan kichik bo'lsa, "Sizga avtobus bepul", aks holda (else), "Chipta narxi 5000 so'm" degan yozuvni konsulga chiqaramiz.

△ Shart "badani" shartdan biroz o'ngga surib yoziladi (xuddi for sikli kabi). if/else dan keyin kelgan va o'ngga surib yozilgan har bir qator if/else shartining badani hisoblanadi.

Dasturni bajaramiz:

```
Yoshingizni kiriting: 6
Sizga avtobus bepul
```

Yana bir bor:

```
Yoshingizni kiriting: 18
Chipta narxi 5000 so'm
```

BIR QATOR if-else

Qisqa kodlar uchun shart va uning badanini 1 qatorga jamlab yozishimiz ham mumkin. Agar shartimiz faqat if dan iborat bo'lsa, uning badanini keyingi qatordan emas, ikki nuqtadan keyin yozish kifoya:

```
yosh = int(input("Yoshingiz nechida?>>>"))
if yosh>65: print("Siz COVID-19 risk guruhida ekan siz")
```

Natija:

```
Yoshingiz nechida?>>>70
Siz COVID-19 risk guruhida ekan siz
```

Agar kodimizda if-else birgalikda qatnashsa, if ning badani undan oldin, else ning badani esa keyin yoziladi:

```
x, y = 25, 50
# x=25 va y=50
print("x>y") if x>y else print("x<y")
```

Natija: x<y

MATNLARNI SOLISHTIRISH

Aksar tizimlar foydalanuvchi kiritgan matni ma'lum bir ko'rinishga keltirib oladi. Sababi, kompyuter uchun 'Ali', 'ALI' va 'ali' – uchta turli xil ism. Ularni solishtirish uchun esa bir ko'rinishga keltirib olish kerak.

Tasavvur qiling, siz yangi elektron pochta ochmoqchisiz va o'zingizga yangi foydalanuvchi ismini tanlashingiz kerak. Kompyuter siz kiritgan foydalanuvchi ismini tizimdagi mavjud foydalanuvchilar bilan solishtiradi, agar ism band bo'lsa, sizga boshqa ism tanlashni tavsiya etadi. Solishtirish jarayonida esa siz tanlagan ismni kichik harflarga o'tkazib, boshqa ismlar bilan solishtiradi.

Registratsiya

Ism	
Anvar	✓
Familiya	
Narzullaev	✓
Login yaratish	
Anvar	
Yangi parol kiritish	

Login band etilgan

Rasmda ko'rsatilgan misolda kimdir anvar@yandex.ru manzili band qilgan, agarda men 'Anvar', 'ANVAR' yoki 'ANVAR' deb login tanlasam ham, anvar@yandex.ru band bo'lgani sababi men so'ragan loginlar rad etilaveradi.

Xo'sh, turli ko'rinishda yozilgan matnlarni qanday qilib solishtiramiz?

Juda oddiy. Matnlarni solishtirishdan avval .lower() metodi yordamida kichik harflar ko'rinishiga keltirib olamiz:

```
ism = 'Ali'
ism.lower() == 'ali'
Natija: True
```

AMALIYOT

Quyidagi dasturlarni Pythonida bajarib ko'ring:

```
avtolar = ['audi', 'bmw', 'volvo', 'kia', 'hyundai']
for avto in avtolar:
    if avto == 'bmw':
        print(avto.upper())
    else:
        print(avto.title())
```



```
javob = float(input("12x6 nechiga teng?>>>"))
if javob!=72:
    print("Javob xato!")
```

```
yosh = int(input("Yoshingiz nechida?>>>"))
if yosh>=18:
    print('Xush kelibsiz!')
else:
    print('Kirish mumkin emas!')
```

```
yil = int(input("Tug'ilgan yilingizni kiriting:"))
if 2021-yil<18:
    print(f"Yoshingiz {2021-yil}da ekan.")
    print("Kirish mumkin emas!")
else:
    print("Xush kelibsiz!")
```

```
login = input("Yangi login tanlang:")
if len(login)<=5: # login uzunligini tekshiramiz
    print("Login 5 harfdan ko'proq bo'lishi shart!")
```

Quyidagi vazifalarni bajaring:

- Yangi cars = ['toyota', 'mazda', 'hyundai', 'gm', 'kia'] degan ro'yxat tuzing, ro'yxat elementlarining birinchi harfini katta qilib konsolga chiqaring. GM uchun ikkala harfni katta qiling.

- Yuqoridagi mashqni teng emas (!=) operatori yordamida bajaring.

- Foydalanuvchi login ismini so'rang. Agar login admin bo'lsa, "Xush kelibsiz, Admin. Foydalanuvchilar ro'yxatini ko'rasizmi?" xabarini konsolga chiqaring. Aks holda, "Xush kelibsiz, {foydalanuvchi_ismi}!" matnini konsolga chiqaring.



<https://click/Da-elf>

- Foydalanuvchidan 2 ta son kiritishni so'rang. Agar ikki son bir-biriga teng bo'lsa, "Sonlar teng" ekan degan yozuvni konsolga chiqaring.
- Foydalanuvchidan istalgan son kiritishni so'rang. Agar son manfiy bo'lsa, konsolga "Manfiy son", agar musbat bo'lsa, "Musbat son" degan xabarni chiqaring.
- Foydalanuvchidan son kiritishni so'rang, agar son musbat bo'lsa, uning ildizini hisoblab konsolga chiqaring. Agar son manfiy bo'lsa, "Musbat son kiriting" degan xabarni chiqaring.
- Kiritilgan sonning toq yoki juftligini konsolga chiqaruvchi dastur yozing.

4.3. BIR NECHTA SHARTLARNI TEKSHIRISH

if yordamida biz faqatgina bitta shartni tekshira olamiz va uning natijasiga ko'ra (True/False) dasturimiz ma'lum bir amallarni bajaradi. Agar dastur davomida bir nechta shartlarni tekshirish talab qilinsa, if-elif-else ketma-ketligidan foydalanamiz.

① **elif – else** va **if** so'zlarining jamlanmasi bo'lib, "aks holda, agar" deb tarjima qilinadi.

if bilan boshlangan ketma-ketlik bir nechta elif lardan iborat bo'lishi mumkin.

```
son = int(input("Son kiriting: "))
if son>0: # agar son 0 dan katta bo'lsa,
    print('Son musbat')
elif son<0: # aks holda, agar son 0 dan kichik bo'lsa,
    print('Son manfiy')
else: # aks holda
    print('Son 0 ga teng')
```

if-elif-else ketma-ketligida Python avval if shartini tekshiradi, shart bajarilmasa, birinchi elif ga o'tadi, birinchi elif sharti bajarilmasa, keyingi elif ga o'tadi va hokazo davom etaveradi.

△ if-elif-else ketma-ketlikda biror shart bajarilishi bilan Python qolgan shartlarni tekshirmaydi.

Keling, bir misolni ko'rib chiqamiz. Deylik, hayvonot bog'iga kirish narxлари quyidagicha belgilangan:

- 4 yoshdan kichik bolalarga kirish bepul
- 4 yoshdan 12 yoshgacha kirish 5000 so'm
- 12 yoshdan kattalarga 10000 so'm

Foydalanuvchining yoshini so'rab, hayvonot bog'iga kirish chiptasi narxini chiqaruvchi dastur yozamiz.

```
yosh = int(input('Yoshingiz nechida? '))
if yosh<4:
    print('Sizga kirish bepul.')
elif yosh<=12:
    print('Sizga kirish 5000 so\'m')
else:
    print('Sizga kirish 10000 so\'m')
```

Dasturimiz avval foydalanuvchining yoshini so'raydi. Birinchi shart yosh 4 dan kichik ekanligini tekshiradi. Agar bu shart bajarilsa, shartlarni tekshirish shu yerdayog to'xtaydi va keyingi shartlar tekshirilmaydi:

```
yoshingiz nechida? 3
Sizga kirish bepul.
```

Agar yosh<4 sharti bajarilmasa, keyingi elif yosh<=12 shartini tekshiriladi, agar shart bajarilsa, quyidagi natija chiqadi:

```
Yoshingiz nechida? 8
Sizga kirish 5000 so'm
```

Agar yuqoridagi ikki shart ham bajarilmasa, navbat o'z-o'zidan else ichidagi kodga keladi:

```
Yoshingiz nechida? 40
Sizga kirish 10000 so'm
```

if-elif-else ketma-ketligida shartlarning tartibi juda muhim. Agar yuqoridagi kodimizda biz dastavval foydalanuvchi 12 dan kichikligini tekshiradigan bo'lsak, dasturimiz biz kutgan natijani bermaydi:

```
yosh = int(input('Yoshingiz nechida? '))
if yosh<12:
    print('Sizga kirish 5000 so\'m')
elif yosh<4:
    print('Sizga kirish bepul.')
else:
    print('Sizga kirish 10000 so\'m')
```

Natija:

```
Yoshingiz nechida? 2
Sizga kirish 5000 so'm
```

Kod yozishda yaxshi amaliyotlardan biri kodlarni qisqa yozish va buyruqlarni qayta-qayta takrorlamaslikdir. Bu kela-jakda kodni o'zgartirishda ham juda qo'l keladi.

Avvalgi dasturda ham print() funksiyasini bir necha bor takrorlamalik uchun kodimizni quyidagicha o'zgartiramiz.

```
yosh = int(input('Yoshingiz nechida? '))
if yosh<=4:
    price = 0
elif yosh<=12:
    price = 5000
else:
    price = 10000
print(f"Sizga kirish {price} so'm")
```


Bu kelajakda kodimizga o'zgartirish kiritishni ham oson qiladi. Misol uchun, hayvonot bog'i 65 dan katta bo'lgan gariyalar uchun chegirma e'lon qilsa, kodimizni quyidagicha yangilab qo'yamiz:

```
yosh = int(input('Yoshingiz nechida? '))
if yosh<=4:
    price = 0
elif yosh<=12:
    price = 5000
elif yosh<65:
    price = 10000
else:
    price = 8000
print(f"Sizga kirish {price} so'm")
```

Natija:

```
Yoshingiz nechida? 70
Sizga kirish 8000 so'm
```

if-elif-else zanjirida ham else qismi majburiy emas:

```
yosh = int(input('Yoshingiz nechida? '))
if yosh<=4:
    price = 0
elif yosh<=12:
    price = 5000
elif yosh<65:
    price = 10000
elif yosh>=65:
    price = 8000
print(f"Sizga kirish {price} so'm")
```

4.4. AND VA OR OPERATORLARI

Yuqoridagi misollarda if yoki elif dan so'ng biz faqatgina bitta shartni tekshirdik. Agar bir vaqtning o'zida bir nechta shartlarni tekshirish talab qilinsa, buning uchun maxsus AND va OR operatorlari mavjud.

OR OPERATORI

OR ingliz tilidan "yoki" deb tarjima qilinadi va ikki va undan ko'p shartlardan **biri** bajarilishini tekshirishda ishlatiladi. Demak, dasturning biror qismi bajarilishi uchun bir nechta shartlardan birining to'g'ri (True) bo'lishi kifoya bo'lsa OR operatoridan foydalanamiz.

Quyidagi misolga qaraylik, foydalanuvchidan hafta kunini so'raymiz, agar kun shanba yoki yakshanba bo'lsa, bugun dam olish kuni degan xabarni chiqaramiz, aks holda, bugun ish kuni degan xabarni chiqaramiz:

```
kun = input("Bugun nima kun?>>>")
if kun.lower()=='shanba' or kun.lower()=='yakshanba':
    print('Bugun dam olish kuni.')
else:
    print('Bugun ish kuni.')
```

2-qatordagi or operatoriga e'tibor qiling, bu operator kun.lower()=='shanba' yoki kun.lower()=='yakshanba' shartlaridan biri bajarilsa, TRUE qiymatini qaytaradi.

```
Bugun nima kun?>>>shanba
Bugun dam olish kuni.
```

Yana ham tushunarli bo'lishi uchun quyidagi kodlarni bajarib ko'ring:

```
>>> print(True or False)
>>> print(True or True)
>>> print(False or False)
```

AND OPERATORI

AND ingliz tilidan "va" deb tarjima qilinadi va ikki va undan ko'p shartlarning **barchasi** bajarilishini tekshirishda ishlatiladi. AND operatori bilan yozilgan shartlarning **barchasi** bajarilgandagina TRUE qiymati qaytadi, shartlardan biri bajarilmay qolsa ham, FALSE qiymati qaytadi.

Quyidagi misolni ko'ramiz:

```
kun = input("Bugun nima kun? ")
harorat = float(input("Havo harorati qanday? "))
if kun.lower()=='yakshanba' and harorat>=30:
    print("Cho'milgani ketdiki!")
elif kun.lower()=='yakshanba' and harorat<30:
    print("Uyda dam olamiz!")
```

3-qatordagi and operatori kun.lower()=='yakshanba' va harorat>=30 shartlarining **ikkisi ham** bajarilgandagina TRUE qiymatini qaytaradi, aks holda, qiymat FALSE bo'ladi.

```
Bugun nima kun? yakshanba
Havo harorati qanday? 29
Uyda dam olamiz!
```

BIR NECHTA SHARTLARNI KETMA-KET YOZISH

Shartlarni yozishda bir nechta and va or operatorlarini aralashtirib ham yozish mumkin.

```
if (yosh<7 or yosh>65) and kun=='chorshanba':
    print('Bugun siz uchun Muzeyga kirish bepul')
```

Yuqorida biz yosh bolalar va qariyalarga har chorshanba kunlari muzeyga kirish bepul ekanligini ko'rsatuvchi soddakod yozdik.

4.5. BOOLEAN MA'LUMOTLAR TURI

Yuqorida taqqoslash operatorlari yordamida turli ifodalarni solishtirishda TRUE yoki FALSE qiymatlari qaytishini ko'rik. Bu qiymatlar boolean (mantiqiy) qiymatlar deb ataladi va das-turlashda juda keng qo'llaniladi.

Boolean ma'lumot turini ham xuddi son yoki matn kabi o'zgaruvchilarda saqlash mumkin.

```
a = True
b = False
```

Quyidagi dasturga e'tibor bering. Restoranimizga kelgan mijoz 15000 so'mlik taom oldi, biz mijoz qo'shimcha choy va salat ham olganiga (olmaganiga) qarab ularning narxini ham yakuniy narxga qo'shishimiz kerak. Mijozning choy yoki salat olganini (olmaganini) biz TRUE va FALSE qiymatlari bilan belgilaymiz:

```
narx = 15000 # mijoz 15000 so'mga taom oldi.
choy = True # mijoz choy ham oldi
salat = False # mijoz salat olmadi

if choy and salat: # agar mijoz choy va salat olgan bo'lsa,
    narx = narx + 10000 # narxga 10000 so'm qo'shamiz
elif choy or salat: # agar choy yoki salat olgan bo'lsa,
    narx = narx + 5000 # narxga 5000 so'm qo'shamiz

print(f"Jami {narx} so'm") # yakuniy narxni chiqaramiz
```

Natija: Jami 20000 so'm
E'tibor bering, choy va salat boolean (mantiqiy) qiymatlar bo'lgani uchun if va elif shartlarida biz choy==True yoki salat==True deb yozib o'tirishimiz shart emas.

Yuqoridagi misolda choy = True (choy oldi) va salat = False (salat olmadi) bo'lgani uchun yakuniy narx narx+5000=20000 chiqdi.

① Pythonda True va False qiymatlari o'rniga 1 va 0 sonlarini ham ishlatish mumkin.

4.6. SHARTLARNI KETMA-KET TEKSHIRISH

if-elif-else zanjirining kamchiliklaridan biri – shartlardan biri bajarilishi bilan qolgan shartlar tekshirilmaydi. Shung uchun shartlarni ketma-ket tekshirish uchun har bir shartni alohida if bilan ajratish talab qilinadi.

Avvalgi bo'limda ko'rgan misolni kengaytiraylik:

```
narx = 15000 # mijoz 15 so'mga ovqat oldi
choy = True
salat = False
non = True
kompot = True
assorti = False

if choy: # agar choy olsa,
    print("Mijoz choy oldi.")
    narx = narx + 3000
if salat: # agar salat olsa,
    print("Mijoz salat oldi.")
    narx = narx + 5000
if non: # agar non olsa,
    print("Mijoz non oldi.")
    narx = narx + 2000
if kompot: # agar kompot olsa,
    print("Mijoz kompot oldi.")
    narx = narx + 5000
if assorti: # agar assorti olsa,
    print("Mijoz assorti oldi.")
    narx = narx + 15000

print(f"Jami {narx} so'm")
```

Natija:

```
Mijoz choy oldi.
Mijoz non oldi.
Mijoz kompot oldi.
Jami 25000 so'm
```

Yuqoridagi dasturda har bir if alohida tekshiriladi, avvalgi yoki keyingi if ga bog'liq emas.

4.7. RO'YXAT TARKIBINI TEKSHIRISH

in OPERATORI

in operatori yordamida biz ro'yxatning tarkibida ma'lum bir element borligini tekshirishimiz mumkin.

```
menu = ['osh', 'qozonkabob', 'shashlik', 'norin', 'somsa']
'manti' in menu # menu da manti bormi?
```

Natija: False

```
menu = ['osh', 'qozonkabob', 'shashlik', 'norin', 'somsa']
'osh' in menu # menu da osh bormi?
```

Natija: True

```
menu = ['osh', 'qozonkabob', 'shashlik', 'norin', 'somsa']
ovqat = input('Nima ovqat yeysiz?>>> ')
if ovqat.lower() in menu:
    print('Buyurtma qabul qilindi.')
else:
    print('Afsuski, bizda bunday ovqat yo\'q')
```

Natija:

```
Nima ovqat yeysiz?>>> somsa
Buyurtma qabul qilindi.
```

Yana bir bor:

```
Nima ovqat yaysiz?>>> dimlama
Afsuski, bizda bunday ovqat yo'q
```

not in OPERATORI

not in operatori yordamida esa biror element ro'yxatda yo'qligini tekshirishimiz mumkin.

```
menu = ['osh', 'qozonkabob', 'shashlik', 'norin', 'somsa']
'manti' not in menu # menu da manti yo'qmi?
```

Natija: True

```
menu = ['osh', 'qozonkabob', 'shashlik', 'norin', 'somsa']
'osh' not in menu # menu da osh yo'qmi?
```

Natija: False

```
menu = ['osh', 'qozonkabob', 'shashlik', 'norin', 'somsa']
ovqat = input('Nima ovqat yaysiz?>>>')
if ovqat.lower() not in menu:
    print('Afsuski, bizda bunday ovqat yo\'q')
else:
    print('Buyurtma qabul qilindi.')
```

not operatorini boshqa shartlarning oldidan ham qo'yishimiz mumkin. Misol uchun, not a==5 ifodasi a!=5 ifodasi bilan bir xil natija qaytaradi.

```
>>> a = 6
>>> a!=5
True
>>> not a==5
True
>>> not False
True
>>> not True
False
```

IKKI RO'YXATNI SOLISHTIRISH

Ikki ro'yxatning tarkibini tekshirish uchun for sikli va yuqoridagi in operatoridan foydalanamiz:

```
menu = ['osh', 'qozonkabob', 'shashlik', 'norin', 'somsa']
buyurtmalar = ["osh", "somsa", "manti", "shashlik"]

for taom in buyurtmalar:
    if taom in menu:
        print(f"Menyuda {taom} bor")
    else:
        print(f"Kechirasiz, menyuda {taom} yo'q")
```

Natija:

```
Menyuda osh bor
Menyuda somsa bor
Kechirasiz, menyuda manti yo'q
Menyuda shashlik bor
```

RO'YXAT BO'SH EMASLIGINI TEKSHIRISH

Yuqoridagi dasturimizda biz foydalanuvchi buyurtma berdi deb tasavvur qilyapmiz. Lekin foydalanuvchidan bo'sh ro'yxat kelisa-chi? Demak, for siklini bajarishdan avval ro'yxat bo'sh emasligiga amin bo'lishimiz kerak. Buning uchun ro'yxat uzunligini tekshirib ko'ramiz. Agar ro'yxat uzunligi 0 ga teng bo'lsa, bu bo'sh ro'yxatdir.

```
>>> list1 = [1,2,3]
>>> len(list1)>0
True
>>> list2 = [] # bo'sh ro'yxat
>>> len(list2)>0
False
```


Lekin Pythonda ro'yxat bo'sh yoki bo'sh emasligini tekshirishning bundan ham oson yo'li bor. Buning uchun if operatoridan so'ng ro'yxat nomini yozish kifoya. Agar ro'yxatda bir dona element bo'lsa ham, bu ifoda TRUE qiymatini, aks holda, FALSE qiymatini qaytaradi.

```
list1 = [1,2,3]
if list1: # Bu ifoda True qaytaradi, sababi, list1
    bo'sh emas
    print("Ro'yxatda elementlar bor")
```

Yuqoridagi usuldan foydalanib avvalgi bo'limda ko'rgan dasturga o'zgartirish kiritamiz:

```
menu = ['osh', 'qozonkabob', 'shashlik', 'norin', 'somsa']
buyurtmalar = ['osh', 'somsa', 'manti', 'shashlik']

if buyurtmalar:
    for taom in buyurtmalar:
        if taom in menu:
            print(f"Menyuda {taom} bor")
        else:
            print(f"Kechirasiz, menyuda {taom} yo'q")
    else: # agar ro'yxat bo'sh bo'lsa,
        print("Savatchangiz bo'sh!")
```

AMALIYOT

Quyidagi dasturlarni alohida fayllarga yozing va bajaring:

- Foydalanuvchidan juft son kiritishni so'rang. Agar foydalanuvchi juft son kiritisa, "Rahmat!", agar toq son kiritisa, "Bu son juft emas" degan xabarni chiqaring.
- Foydalanuvchi yoshini so'rang va muzeyga kirish uchun chipta narxini quyidagicha chiqaring:
 - Agar foydalanuvchi 4 yoshdan kichkina yoki 60 dan katta bo'lsa, bepul.

- Agar foydalanuvchi 18 dan kichik bo'lsa, 10000 so'm.
 - Agar foydalanuvchi 18 dan katta bo'lsa, 20000 so'm.
- Foydalanuvchidan ikkita son kiritishni so'rang, sonlarni solishtiring va ularning teng yoki katta/kichikligi haqida xabarni chiqaring.

- mahsulotlar degan ro'yxat yarating va kamida 10 ta turli mahsulotni kiriting. Yangi, savat degan bo'sh ro'yxat yaratish va foydalanuvchidan savatga kamida 5 ta mahsulot kiritishni so'rang. Savatdagi elementlarni mahsulotlar ro'yxati bilan solishtiring va qaysi biri ro'yxatda bo'lsa, "Mahsulot do'konimizda bor", aks holda, "Mahsulot do'konimizda yo'q" degan xabarlarini chiqaring.

- Yuqoridagi dasturni quyidagicha o'zgartiring: foydalanuvchidan 5 ta mahsulot kiritishni so'rang. Foydalanuvchi so'ragan va do'konda bor mahsulotlarni yangi, bor mahsulotlar degan ro'yxatga, do'konda yo'q mahsulotlarni esa mavjud_emas degan ro'yxatga qo'shing. Agar mavjud_emas ro'yxati bo'sh bo'lsa, "Siz so'ragan barcha mahsulotlar do'konimizda bor" degan xabarni, aks holda, "Quyidagi mahsulotlar do'konimizda yo'q:" degan xabarni chiqaring.

- foydalanuvchilar degan ro'yxat tuzing va kamida 5 ta login qo'shing. Foydalanuvchidan yangi login tanlashni so'rang va foydalanuvchi kiritgan loginni foydalanuvchilar degan ro'yxatning tarkibi bilan solishtiring. Agar ro'yxatda bunday login mavjud bo'lsa, "login band, yangi login tanlang!", aks holda, "Xush kelibsiz, {foydalanuvchi}!" xabarini chiqaring.
- Foydalanuvchidan biror butun son kiritishni so'rang. Foydalanuvchi kiritgan sonning 2 dan 10 gacha bo'lgan sonlardan qay biriga goldiqsiz bo'linishini konsolga chiqaring.

5-BOB

LUG'AT VA TO'PLAM

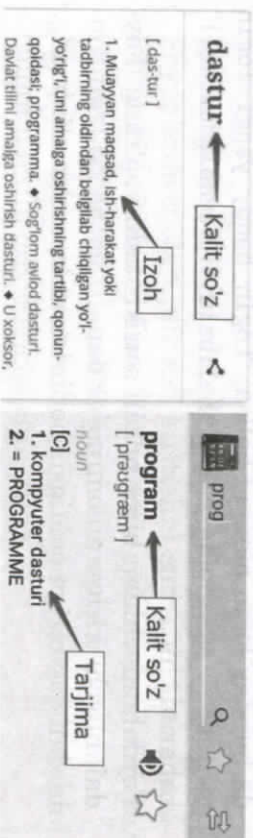
Ushbu bobda yangi ma'lumot turlari, lug'at (dictionary) va to'plam (set) bilan tanishamiz. Dars davomida lug'at yaratish, unga ma'lumot qo'shish, lug'atning ichida ro'yxat yoki, aksincha, ro'yxatning ichida lug'at saqlash kabi mavzular bilan tanishamiz. Darsimiz so'ngida esa to'plamlar haqida qisqacha to'xtalib o'tamiz.



<https://elli.co/Da-dict>

5.1. LUG'AT NIMA?

Lug'at bir obyektga bog'liq ma'lumotlarni kalit so'z va qiymat juftligi ko'rinishida saqlash imkoniyatini beradi. Bu esa, o'z navbatida, lug'at ichidagi ma'lumotlarni tushunishni osonlashtiradi. Misol uchun, biz biror avtomobilga oid lug'at yaratishimiz va lug'atda shu avtomobilga tegishli barcha ma'lumotlarni saqlashimiz mumkin (avtomobilning nomi, rangi, yili, motori, narxi va hokazo).



14-RASM. Izohli Lug'at (savodxon.uz) / English-Uzbek Dictionary (SmarBox) dasturlari

Nima uchun bu ma'lumot turi lug'at deyilishini tushunish uchun oddiy lug'atga qaraylik. Odatda, lug'atdagi ma'lumotlar ikki qismdan iborat bo'ladi: kalit so'z va izoh (yoki taryjima). Xuddi oddiy lug'atlardagi kabi Python lug'atdagi ma'lumotlar ham ikki qismdan iborat bo'ladi: kalit so'z va qiymat (ingliz tilida bu "key-value pair", ya'ni "kalit-qiymat juftligi" deyiladi).

① Dasturlashda ko'p ishlatiladigan atamalarni ingliz tilida yodlab qolish juda muhim! Bu sizga kelajakda yangi ma'lumotlar izlashda, xatolar ustida ishlashda, umuman, ish faoliyatingizda ko'p asqatadi. Shuning uchun variable, integer, float, string, list, tuple, dictionary, function, loop, key, value va boshqa atamalarning ma'nosini yaxshilab o'zlashtirib oling.

Pythonda lug'at yaratish uchun katta (jungalak) qavsdan foydalanamiz. Qavs ichida har bir element uchun kalit so'z va qiymat beramiz. Ularning orasi ikki nuqta (:) bilan, har bir element (juftlik) esa vergul (,) bilan ajratiladi:

```
car = {'model': 'ferrari', 'rang': 'qizil'}
```

Yuqorida car degan lug'at yaratdik. Lug'atda 2 ta element bor: mashinaning modeli (ferrari) va rangi (qizil). Bu yerda 'model' va 'rang' kalit so'zlar, 'ferrari' va 'qizil' esa mos keluvchi kalit so'zlarning qiymatlaridir.

△ Lug'at ichidagi kalitlar takrorlanmasligi kerak. Agar lug'at yaratishda kalit so'zlar takrorlansa, ulardan faqat oxirgisining qiymati saqlanib qoladi.

5.2. LUG'AT BILAN ISHLASH

Demak, Pytonda lug'at kalit-qiymat juftliklarining yig'indisi ekan. Lug'atdagi biror elementning qiymatini ko'rish uchun unga kalit so'z orqali murojaat etamiz:

```
>>> car = {'model': 'ferrari', 'rang': 'qizil'}
>>> print(car['model'])
ferrari
>>> print(car['rang'])
qizil
```

Lug'atdagi qiymatlar son (int, float), matn (string), ro'yxat (list, tuple), hatto boshqa lug'at ham bo'lishi mumkin.

```
talaba = {'ism': 'murod olimov', 'yosh': 20, 't_yil': 2000}
print(f"{talaba['ism'].title()}, \
{talaba['t_yil']}-yilda tug'ilgan, \
{talaba['yosh']} yoshda")
```

Natija: Murod Olimov, 2000-yilda tug'ilgan, 20 yoshda
Uzun lug'atlarni bir necha qatorga bo'lib yozishimiz ham mumkin. Bunda har bir element alohida qatordan yoziladi va qator oxirida vergul qo'yilib, yangi qatorga o'tiladi:

```
car = {
    'make': 'GM',
    'model': 'Malibu',
    'color': 'Black',
    'gear': 'Automatic',
    'year': 2020,
    'price': 40000
}
```

get() METODI

Yuqorida biz lug'at elementiga kalit so'z orqali murojaat etishni o'rgandik.

```
>>> print(car['model'])
Malibu
```

Bu usulning kamchiligi shundaki, agar lug'atda siz so'ragan kalit topilmasa, dastur **KeyError** xatoligi bilan to'xtab qoladi.

```
>>> print(car['narx'])
KeyError: 'narx'
```

Lug'atda 'narx' kalit so'zi bo'lmagani uchun yuqoridagi kod **KeyError** degan xatoni qaytardi.

Biz kelgusi darslarimizda Pythondagi xatolarni dastur bajarilishi jarayonida "tutib olish"ni o'rganamiz. Hozircha esa get() metodi yordamida lug'atga murojaat etish va mavjud bo'lmagan kalitning o'rniga biror xabar qaytarishni ko'rib chiqaylik.

```
narx = car.get('narx', 'Bunday kalit mavjud emas')
```

Yuqorida lug'at nomidan so'ng .get() metodini yozdik, argumentlar sifatida kalit so'z ('narx') va kalit mavjud bo'lmaganda chiqadigan xabarni yozdik ('Bunday kalit mavjud emas').

```
>>> print(narx)
Bunday ism mavjud emas
```

Agar .get() metodida ikkinchi argumentni tashlab ketсан-giz va siz so'ragan kalit mavjud bo'lmasa, .get() metodi None qiymatini qaytaradi.

```
>>> narx = car.get('narx')
>>> print(narx)
None
```

❶ None qiymat mavjud emas degan ma'noni beradi.

YANGI JUFTLIK QO'SHISH

Lug'atga yangi element (kalit so'z va qiymatlar) qo'shishimiz ham mumkin. Keling, yuqoridagi talaba nomli lug'atga yana 2 ta yangi – kurs va fakultet nomi kalit so'zlar va qiymatlar qo'shamiz:

```
talabal['kurs'] = 4  
talabal['fakultet'] = 'informatika'
```

Lug'atni konsolga chiqarib ko'ramiz:

```
>>> print(talaba)  
{ 'ism': 'abdulloh', 'yosh': 20, 't_yil': 2000, 'kurs':  
4, 'fakultet': 'informatika' }
```

BO'SH LUG'AT

Ba'zida dastur boshida bo'sh lug'at yaratib, dastur davomida lug'atga yangi ma'lumotlar kiritib borish talab qilinishi mumkin. Bunday holatda bo'sh lug'at quyidagicha yaratiladi:

```
car = {}
```

Dastur davomida esa lug'atga qiymatlar kiritib borilishi mumkin:

```
car['model'] = 'Mazda 6'  
car['color'] = 'Red'  
car['price'] = 40000  
print(f"{car['color']} {car['model']}, {car['price']}")
```

Natija:

```
Red Mazda 6, 40000$
```

Lug'atga kalit so'zlar qanday ketma-ketlikda kiritilsa, shu tartib saqlanib qoladi.

QIYMATNI O'ZGARTIRISH

Biror kalit so'zga tegishli qiymatni o'zgartirish esa quyidagicha amalga oshiriladi:

```
car['price'] = 38000  
print(f"{car['color']} {car['model']}, {car['price']}")
```

Natija:

```
Red Mazda 6, 38000$
```

KALIT SO'Z-QIYMAT JUFTLIGINI O'CHIRISH

Lug'atdagi elementlarni del operatori yordamida o'chiramiz:

```
>>> car = {'model': 'Malibu', 'color': 'Black', 'price': 40000}  
>>> print(car)  
{ 'model': 'Malibu', 'color': 'Black', 'price': 40000}  
>>> del car['color']  
{ 'model': 'Malibu', 'price': 40000}
```

AMALIYOT

- otam (onam, akam, ukam va hokazo) degan lug'at yaratib va lug'atga shu inson haqida kamida 3 ta ma'lumot kiritib (ismi, tug'ilgan yili, shahri, manzili va hokazo). Lug'atdagi ma'lumotni matn shaklida konsolga chiqaring.
- Oila a'zolariningizning sevimli taomlari lug'atini tuzing. Lug'atda kamida 5 ta ism-taom juftligi bo'lsin. Kamida uch kishining sevimli taomini konsolga chiqaring.
- Python izohli lug'ati tuzing: Lug'atga shu kunga qadar o'rgangan 10 ta so'zni (atamani) kiritib (masalan, integer, float, string, if, else va hokazo) va qiymat sifatida har birining qisqacha tarjimasini yozing.

- Foydalanuvchidan biroz so'z kiritishni so'rang va so'zning tarjimasini yuqoridagi lug'atdan chiqarib bering. Agar so'z lug'atda mavjud bo'lmasa, "Bunday so'z mavjud emas" degan xabarni chiqaring.
- Yuqoridagi vazifani if-else yordamida bajaring va natijani foydalanuvchiga tushunarli ko'rinishda chiqaring.

5.2. LUG'AT ELEMENTLARI BILAN ISHLASH

Avvalgi bo'limda lug'at elementlarga kalit so'z orqali murojaat etishni o'rgandik. Lug'atlar katta yoki kichik bo'lishi mumkin. Ba'zida lug'atdagi barcha kalitlarni yoki qiymatlarni bilmasligimiz mumkin. Bunday holatda qanday yo'l tutamiz?

Ushbu darsimda lug'at elementlarini turli usullar yordamida chiqarishni o'rgatamiz.



<https://olicoo/0a-dic2>

.items() METODI

Ushbu metod yordamida lug'at ichidagi barcha kalit-qiymat juftliklarini ko'rishimiz mumkin.

```
talaba = {
    'ism': 'alijon',
    'familiya': 'shamshiyev',
    'yosh': 22,
    'fakultet': 'matematika',
    'kurs': 4
}

print(talaba.items())
```

Natija:

```
dict_items([('ism', 'alijon'), ('familiya', 'shamshiyev'),
('yosh', 22), ('fakultet', 'matematika'), ('kurs', 4)])
```

Bu metodni to'g'ridan to'g'ri emas, for sikli yordamida chiqarish orqali lug'atdagi barcha elementlarni tushunarliroq shaklda ko'rishimiz mumkin.

```
for kalit, qiymat in talaba.items():
    print(f"Kalit: {kalit}")
    print(f"Qiymat: {qiymat}")
```

Natija:

```
Kalit: ism
Qiymat: alijon
Kalit: familiya
Qiymat: shamshiyev
Kalit: yosh
Qiymat: 22
Kalit: fakultet
Qiymat: matematika
Kalit: kurs
Qiymat: 4
```

Yuqoridagi dasturda talaba lug'atidagi har bir kalit va qiymat juftligini konsolga chiqardik. E'tibor bering, for siklida biz bir emas, ikkita o'zgaruvchi yaratib oldik (kalit va qiymat). Bu usul lug'atlardagi ma'lumotlarni chiroyli ko'rinishda chiqarish uchun juda qulay.

```
telefonlar = {
    'ali': 'iphone x',
    'vali': 'galaxy s9',
    'olim': 'mi 10 pro',
    'orif': 'nokla 3310'
}

for k, q in telefonlar.items():
    print(f"{k.title()}ning telefoni {q}")
```

Natija:

```
Alining telefoni iphone x
Valining telefoni galaxy s9
Olimning telefoni mi 10 pro
Orifning telefoni nokia 3310
```

.keys() METODI

Agar lug'atdagi kalit so'zlarni topish talab qilinsa, .keys() metodidan foydalanamiz:

```
mahsulotlar = { # Do'konidagi mahsulotlar
    'olma':10000,
    'anor':20000,
    'uzum':40000,
    'anjir':25000,
    'shaftoli':30000
}
print(mahsulotlar.keys())
```

Natija:

```
dict_keys(['olma', 'anor', 'uzum', 'anjir', 'shaftoli'])
```

Bu yerda ham for siklidan foydalanish mumkin:

```
print('Do\'konidagi mahsulotlar:')
for mahsulot in mahsulotlar.keys():
    print(mahsulot.title())
```

Natija:

```
Do'konidagi mahsulotlar:
Olma
Anor
Uzum
Anjir
Shaftoli
```

① Yuqoridagi kodimizda for siklida .keys() metodini ishlatmasak ham, xuddi shu natija chiqadi.

RO'YXAT VA LUG'AT

for sikli va if sharti yordamida lug'atdagi biror qiymatlarni alohida chiqarishimiz ham mumkin:

```
bozorlik = ['anor', 'uzum', 'non', 'baliq']
for m in mahsulotlar:
    if m in bozorlik:
        print(f'{m.title()} {mahsulotlar[m]} so'm")
```

Natija:

```
Anor 20000 so'm
Uzum 40000 so'm
```

Yuqoridagi kodga e'tibor bering. Biz avval bozorlik degan ro'yxat yaratdik (uyga bozor qilyapimiz), keyin esa mahsulotlar lug'atidagi har bir mahsulotni bizdagi bozorlik ro'yxati bilan solishtirdik. Agar mahsulot bizning bozorlik ro'yxatimizda bo'lsa, uning narxini konsolga chiqardik.

Quyidagi misolda esa, aksincha, bozorlik ro'yxatidagi har bir elementni do'konidagi mahsulotlar bilan solishtiramiz va mahsulot do'konda yo'q bo'lsa, do'konga murojaat qoldiramiz:

```
for buyum in bozorlik:
    if buyum not in mahsulotlar:
        print(f"Kechirasiz, bizda {buyum} yo'q")
```

Natija:

```
Kechirasiz, bizda non yo'q
Kechirasiz, bizda baliq yo'q
```


LUG'AT ELEMENTLARINI TARTIB BILAN CHIQARISH

Avval aytganimizdek, lug'at elementlari biz kiritgan tartibda saqlanadi. Agar elementlarini alifbo tartibida chiqarish talab qilinsa, sorted() funksiyasidan foydalanamiz.

```
print("Do'konimizdagi mahsulotlar:")
for mahsulot in sorted(mahsulotlar):
    print(mahsulot.title())
```

Natija:

```
Anjir
Anor
Olma
Shaftoli
Uzum
```

.values() METODI

Yuqorida biz .key() metodi yordamida lug'atdagi elementlarning kalitlarini ko'rishni o'rgandik. Agar element qiymatlarini topish talab qilinsa, .values() metodidan foydalanamiz:

```
telefonlar = {
    'ali': 'iphone x',
    'vali': 'galaxy s9',
    'olim': 'mi 10',
    'orif': 'nokia 3310'
}
```

```
print(telefonlar.values())
```

Natija:

```
dict_values(['iphone x', 'galaxy s9', 'mi 10', 'nokia 3310'])
```

Quyidagi kodni ham bajarib ko'ramiz:

```
print('Foydalanuvchilarning telefonlari:')
for tel in telefonlar.values():
    print(tel)
```

Natija:

```
Foydalanuvchilarning telefonlari:
iphone x
galaxy s9
mi 10
nokia 3310
```

Yuqoridagi usul bilan qiymatlarni chiqarganimizda lug'atdagi barcha qiymatlar chiqib keladi. Agar biror qiymat ko'p marta qaytarilsa, konsolga ham ko'p marta chiqadi.

Quyidagi misolni ko'ramiz:

```
telefonlar = {
    'ali': 'iphone x',
    'vali': 'galaxy s9',
    'olim': 'mi 10 pro',
    'orif': 'nokia 3310',
    'hamida': 'galaxy s9',
    'maryam': 'huawei p30',
    'tohir': 'iphone x',
    'umar': 'iphone x'
}
```

```
print('Foydalanuvchilarning telefonlari:')
for tel in telefonlar.values():
    print(tel)
```

Natija:

```
Foydalanuvchilarning telefonlari:
iphone x
galaxy s9
mi 10 pro
nokia 3310
galaxy s9
huawei p30
iphone x
iphone x
```

Yuqoridagi natijaga e'tibor bersangiz, bir nechta foydalanuvchilar iPhone x va Galaxy S9 telefonidan foydalanar ekan va bu modellar qayta-qayta konsolga chiqdi.

Buning oldini olish uchun set() funksiyasidan foydalanishimiz mumkin. Bu funksiya to'plam yaratish uchun ishlatiladi:

```
print(Foydalanuvchilarning telefonlari: ' ')\nfor tel in set(telefonlar.values()):\n    print(tel)
```

Natija:

Foydalanuvchilarning telefonlari:

```
iphone x\nGalaxy S9\nNokia 3310\nmi 10 pro\nhuawei P30
```

AMALIYOT

- Python izohli lug'atini yarating va lug'atga kamida 10 ta so'z qo'shing. Lug'atdagi har bir kalit va qiymatni for sikli yordamida, alifbo ketma-ketligida chiroyli qilib konsolga chiqaring.
- Davlatlar va ularning poytaxtlari lug'atini tuzing. Avval lug'atdagi davlatlarni, keyin poytaxtlarni alohida-alohida, alifbo ketma-ketligida konsolga chiqaring.
- Foydalanuvchidan istalgan davlatni kiritishni so'rang va shu davlatning poytaxtini konsolga chiqaring. Agar foydalanuvchi lug'atda yo'q davlatni kiritrsa, "Bizda bunday ma'lumot yo'q" degan xabarni chiqaring.
- Restoran menyusi lug'atini tuzing (kamida 10 ta taom-narx juftligini kiriting). Foydalanuvchidan 3 ta ovqat buyurtma berishni so'rang. Foydalanuvchi kiritgan taomlarni menu bilan solishtiring, agar taom menyuda bo'lsa, narxini ko'rsating, aks holda, "bizda bunday taom yo'q" degan xabarni chiqaring.

5.3. TO'PLAMLAR

Pythonda to'plam (set) yana bir ma'lumot turi bo'lib, ro'yxat va lug'at kabi bir nechta elementlarni saqlashda qo'llaniladi.

To'plam yaratish uchun ham lug'atlardagi kabi katta (jinalgalak) qavsdan foydalanamiz:

```
>>> sonlar = {1,2,3}\n>>> print(sonlar)\n{1, 2, 3}\n>>> ismlar = {'alijon', 'valijon', 'boqijon'}\n>>> print(ismlar)\n{'alijon', 'valijon', 'boqijon'}
```

To'plam bir xil elementlarni saqlamaydi:

```
>>> sonlar = {1,2,3,3,3,4,4,5,6,5}\n>>> print(sonlar)\n{1, 2, 3, 4, 5, 6}
```

E'tibor qiling, yuqoridagi kodning 1-qatorida biz 3, 4 va 5 qiymatlarini 2 marta takrorlab yozganimizga qaramay, to'plam ichida bu sonlar bir marta saqlandi, xolos.

Bo'sh to'plam yaratish uchun set() funksiyasidan foydalanamiz:

```
a = set() # bo'sh to'plam
```

Ro'yxatdan farqli ravishda, set ichidagi elementlar biror tartibda saqlanmaydi va ularga indeks orqali murojaat etib bo'lmaydi.

```
sonlar = {1,2,3,3,4,4,5,6}\nprint(sonlar[0])
```

Natija:

```
TypeError: 'set' object is not subscriptable
```

Avvalgi bo'limda ko'rganimizdek, to'plamlar biror ro'yxatdan qaytarilgan elementlarni o'chirib tashlash uchun juda qulay.


```
mevalar = ['olma', 'anjir', 'olma', 'uzum', 'olma', 'uzum']
mevalar = set(mevalar)
print(mevalar)
```

Natija:

```
{'anjir', 'olma', 'uzum'}
```

Yuqoridagi misolda mevalar ro'yxatida olma va uzum bir necha bor takrorlangan edi, biz set() funksiyasi yordamida ro'yxatni to'plamga o'zgartirdik va ortiqcha elementlardan xalos bo'ldik.

Agar to'plamni yana ro'yxatga o'tkazish talab qilinsa, list() funksiyasiga murojaat etamiz:

```
mevalar = list(mevalar)
print(mevalar)
```

Natija:

```
['anjir', 'olma', 'uzum']
```

TO'PLAMGA ELEMENT QO'SHISH

To'plamga yagona element qo'shish uchun .add() metodidan, bir nechta element qo'shish uchun esa .update() metodidan foydalanamiz:

```
>>> mevalar = {'anjir', 'olma', 'uzum'}
>>> mevalar.add('banan') # bitta element qo'shamiz
>>> print(mevalar)
{'banan', 'anjir', 'olma', 'uzum'}
>>> mevalar.update(['anor', 'qovun'])
>>> print(mevalar)
{'anjir', 'olma', 'anor', 'qovun', 'banan', 'uzum'}
```

① .update() metodidan foydalanganda parametr sifatida ro'yxat yoki lug'at uzatishimiz mumkin.

TO'PLAMDAN ELEMENT O'CHIRISH

To'plamdan element o'chirish uchun .discard() va .remove() metodlari mavjud.

```
>>> mevalar = {'anjir', 'olma', 'uzum', 'banan', 'anor'}
>>> mevalar.discard('anjir')
>>> print(mevalar)
{'olma', 'anor', 'banan', 'uzum'}
>>> mevalar.remove('banan')
>>> print(mevalar)
{'olma', 'anor', 'uzum'}
```

Bu ikki metod bir vazifani bajaradi, ularning o'rtasidagi farq shundaki, agar biz o'chirmog'chi bo'lgan element to'plamda mavjud bo'lmasa, .remove() metodi xato qaytaradi, .discard() esa xato qaytarmaydi.

```
>>> sonlar = {1,2,3,4,5,6}
>>> sonlar.discard(7) # to'plamda 7 yo'q
>>> print(sonlar)
{1, 2, 3, 4, 5, 6}
>>> sonlar.remove(7)
KeyError: 7
```

To'plamdan element o'chirishning (sug'urib olishning) yana bir usuli .pop() metodidir. Lekin to'plam elementlari indeksi mavjud bo'lmagani sababli .pop() metodi to'plamdan tasodifiy elementni sug'urib oladi.

```
>>> sonlar = {1,2,3,4,5,6}
>>> son = sonlar.pop()
>>> print(son)
1
>>> print(sonlar)
{2, 3, 4, 5, 6}
```

TO'PLAMLAR USTIDA AMALLAR

To'plamlar matematikada ham keng qo'llanilgani uchun ularning ustida o'ziga xos matematik amallar bajarish mumkin. Misol uchun, ikki to'plamni birlashtirish uchun | operatori yoki .union() metodidan foydalanamiz.

```
>>> A = {1,2,3,4}
>>> B = {3,4,5,6}
>>> C = A|B
>>> print(C)
{1, 2, 3, 4, 5, 6}
>>> D = A.union(B)
>>> print(D)
{1, 2, 3, 4, 5, 6}
```

Ikki to'plamning kesishmasini (bir xil elementlarini) topish uchun esa & operatori yoki .intersection() metodidan foydalanamiz:

```
>>> A = {1,2,3,4}
>>> B = {3,4,5,6}
>>> print(A&B)
{3, 4} # A va B uchun umumiy elementlar
>>> print(A.intersection(B))
{3, 4}
```

Ikki to'plam o'rtasidagi farqni topish uchun esa ayirish (-) operatori yoki .difference() metodidan foydalanamiz. B to'plamning A to'plamdan farqi deganda (A-B) A to'plamga kiruvchi, lekin B to'plamda yo'q elementlar tushuniladi.

```
>>> A = {1,2,3,4}
>>> B = {3,4,5,6}
>>> print(A-B)
{1, 2} # 1 va 2 A da bor B da yo'q
>>> print(B.difference(A))
{5, 6} # 5 va 6 B da bor A da yo'q
```

To'plamlar o'rtasidagi simmetrik farqni topish uchun ^ operatori yoki .symmetric_difference() metodidan foydala-

namiz. Simmetrik farq deb A va B to'plamga kiradigan, lekin bir vaqtda ikkalasiga kirmaydigan elementlar tushuniladi. Quyidagi misolda {3, 4} elementlari ikkala to'plamda bo'lgani uchun simmetrik farqqa kirmaydi.

```
>>> A = {1,2,3,4}
>>> B = {3,4,5,6}
>>> print(A^B)
{1, 2, 5, 6}
>>> print(A.symmetric_difference(B))
{1, 2, 5, 6}
```

AMALIYOT

- Uch xil ranglar nomi saqlangan to'plam yarating.
- To'plamga .add() va .update() metodlari yordamida ranglar qo'shing.
- Quyidagi ikki to'plam uchun umumiy elementlarni ajratib olib, yangi to'plam yarating:

```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}
```

- Yuqoridagi ikki to'plam orasidagi farqlarni konsolga chiqaring.
- Ikki to'plam orasidagi simmetrik farqni toping.
- Sizda quyidagi ikki ro'yxat bor:

```
bozorlik = ['choy', 'non', 'kartoshka', 'tuxum', 'sut']
mahsulotlar = ['non', 'sut', 'tuxum', 'olma', 'un', 'tuz']
```

- bozorlik – siz sotib olishingiz kerak bo'lgan narsalar;
- mahsulotlar – do'kondagi mavjud mahsulotlar.
- Sotib olishingiz kerak bo'lgan mahsulotlarning qay biri do'konda bor ekanini alohida ro'yxatga (list) saqlang.
- Do'kon mavjud bo'lmagan mahsulotlar ro'yxatini yarating.
- Do'kon egasi siz so'ragan mahsulotlarni olib keldi. Mahsulotlar ro'yxatiga yangi mahsulotlarni qo'shing.

5.4. NESTING

Baʼzida dasturlash jarayonida lugʻatning ichida roʻyxatlar yo boshqa lugʻatni yoki, aksincha, roʻyxat ichida lugʻatni saqlash ham qoʻl kelishi mumkin. Bu ingliz tilida **Nesting** deyiladi. Nesting Pythonidagi foydali xususiyatlardan biridir. Keling, bunga bir nechta misollar koʻrib chiqamiz.

LUGʻATLAR ROʻYXATI

Biz avvalgi darsimizda talabalarining maʼlumotlarini lugʻat shaklida saqlashni koʻrgan edik. Shunday talabalardan yuzlab boʻlganda ularning har biriga alohida lugʻat qilishimiz tabiiy, lekin bu lugʻatlar bilan ishlash qiyinchilik tugʻdirishi mumkin. Shunday holatda barcha lugʻatlarni (talabalarni) bitta roʻyxatga joylab, ular ustida turli amallar bajarish mumkin.

Quyidagi misolda 3 ta alohida lugʻatda 3 ta avtomobil haqida maʼlumotlar saqlangan:

```
car0 = {
    'model': 'lacetti', 'rang': 'oq',
    'yil': 2018, 'narx': 13000,
    'km': 50000, 'korobka': 'avtomat'
}

car1 = {
    'model': 'nexia 3', 'rang': 'qora',
    'yil': 2015, 'narx': 9000,
    'km': 89000, 'korobka': 'mexanika'
}

car2 = {
    'model': 'gentra', 'rang': 'qizil',
    'yil': 2019, 'narx': 15000,
    'km': 20000, 'korobka': 'mexanika'
}
```



<https://click/Da-nesting>

Agar biz har bir avtomobil haqida maʼlumot olmogchi boʻlsak, har bir lugʻatga alohida murojaat etishimiz talab qilinadi:

```
car = car0
print(f"{car['model']}.title()), \
{car['rang']} rang, \
{car['yil']}-yil, {car['narx']}$")

car = car1
print(f"{car['model']}.title()), \
{car['rang']} rang, \
{car['yil']}-yil, {car['narx']}$")

car = car2
print(f"{car['model']}.title()), \
{car['rang']} rang, \
{car['yil']}-yil, {car['narx']}$")
```

Natija:

```
lacetti, oq rang, 2018-yil, 13000$
Nexia 3, qora rang, 2015-yil, 9000$
Gentra, qizil rang, 2019-yil, 15000$
```

Yuqoridagi natijani chiqarishning osonroq usuli barcha avtolarni bitta roʻyxatga joylab, for siklidan foydalanishdir:

```
cars = [car0, car1, car2] # lugʻatlar roʻyxati
for car in cars:
    print(f"{car['model']}.title()), "
        f"{car['rang']} rang, "
        f"{car['yil']}-yil, {car['narx']}$")
```

Natija:

```
lacetti, oq rang, 2018-yil, 13000$
Nexia 3, qora rang, 2015-yil, 9000$
Gentra, qizil rang, 2019-yil, 15000$
```

Ishimiz birmuncha yengillashdi, kodimiz ham qisqardi. Natija esa bir xil.

Ro'yxat ichidagi istalgan lug'atga indeks bo'yicha murojaat etishim ham mumkin (lug'at nomini yodlab qolish shart emas).

```
>>> print(cars[0])
{'model': 'lacetti', 'rang': 'oq', 'yil': 2018, 'narx': 13000, 'km': 50000, 'korobka': 'avtomat'}
```

Biror lug'atdagi elementga murojaat etish uchun esa quyidagi usuldan foydalanishimiz mumkin:

```
>>> print(cars[0]['model'])
lacetti
>>> print(f"{cars[2]['rang']}.title()}"
f"{cars[2]['model']}")
qizil gentra
```

for sikli yordamida biz bo'sh lug'atlar ro'yxatini ham yaratib olishimiz mumkin:

```
malibus=[] # Malibu mashinalari uchun bo'sh ro'yxat yaratdik
for n in range(10):
    new_car = { # har bir yangi avto uchun lug'at yaratamiz
        'model': 'malibu',
        'rang': None, # rangi noaniq
        'yil': 2020,
        'narx': None, # narxi belgilanmagan
        'km': 0,
        'korobka': 'avto'
    }
    malibus.append(new_car) # lug'atni ro'yxatga qo'shamiz
```

Yuqoridagi misolda biz 10 ta Malibu mashinasidan iborat ro'yxat tuzdik. E'tibor qiling, 'rang' kalitiga qiymat bermadik va None deb qoldirdik.

Endi ishlab chiqarish jarayonida mashinalarga turli ranglarni berishimiz mumkin. Misol uchun, birinchi 3 ta mashinaga qizil rang beramiz:

```
for malibu in malibus[:3]:
    malibu['rang']='qizil'
Keyingi 3 tasiga esa qora:
for malibu in malibus[3:6]:
    malibu['rang']='qora'
```

Oxirgi 4 ta Malibuni qora, korobkasini mexanika qilamiz:

```
for malibu in malibus[6:]: # oxirgi 4 ta mashinaning
    malibu['rang']='qora' # rangi qora
    malibu['korobka']='mexanika' # korobkasi mexanika
```

Keling, endi mashinalarning korobkasidan kelib chiqqan holda narx belgilaymiz:

```
for malibu in malibus:
    if malibu['korobka']=='avto':
        malibu['narx']=40000
    else:
        malibu['narx']=35000
```

Mashinalar ro'yxatini konsolga chiqaramiz:

```
for malibu in malibus:
    print(malibu.values())
```

Natija:

```
dict_values(['malibu', 'qizil', 2020, 40000, 0, 'avto'])
dict_values(['malibu', 'qizil', 2020, 40000, 0, 'avto'])
dict_values(['malibu', 'qizil', 2020, 40000, 0, 'avto'])
dict_values(['malibu', 'qora', 2020, 40000, 0, 'avto'])
dict_values(['malibu', 'qora', 2020, 40000, 0, 'avto'])
dict_values(['malibu', 'qora', 2020, 40000, 0, 'avto'])
dict_values(['malibu', 'qora', 2020, 35000, 0, 'mexanika'])
dict_values(['malibu', 'qora', 2020, 35000, 0, 'mexanika'])
dict_values(['malibu', 'qora', 2020, 35000, 0, 'mexanika'])
dict_values(['malibu', 'qora', 2020, 35000, 0, 'mexanika'])
```


LUG'AT ICHIDA RO'YXAT

Bir kaliga bir nechta qiymatlar berish talab etilganda qiymatlarni ro'yxat ko'rinishida yozish o'rinishidir. Misol uchun, bir tashkilotda bir nechta dasturchilar ishlaydi va har bir dasturchi turli dasturlash tillarini biladi. Keling, dasturchilar lug'atini tuzamiz va har bir dasturchi haqidagi ma'lumotni konsolga chiqaramiz:

```
dasturchilar = {
    'ali': ['python', 'c++'],
    'vali': ['html', 'css', 'js'],
    'hasan': ['php', 'sql'],
    'husan': ['python', 'php'],
    'maryam': ['c++', 'c#']
}

for ism, tillar in dasturchilar.items():
    print(f"\n{ism.title()}: ", end='')
    for til in tillar:
        print(f'{til.upper()} ', end='')
```

① print() funksiyasi har bir matndan so'ng yangi qator tashlaydi. Buning oldini olish uchun quyidagi usuldan foydalanish mumkin: print(string, end = ' ').

Natija:

```
Ali:PYTHON C++
Vali:HTML CSS JS
Hasan:PHP SQL
Husan:PYTHON PHP
Maryam:C++ C#
```

LUG'AT ICHIDA LUG'AT

Bunday qilish tavsiya etilmasa-da, istisno holatlarda lug'at ichidagi qiymatlarni lug'at ko'rinishida ham saqlash mumkin. Misol uchun, ish joyingizdagi hamkasblaringiz haqidagi

ma'lumotlarni saqlashda hamkasbingizning ismi kalit, u haqidagi ma'lumotlar esa lug'at ko'rinishida berilishi mumkin.

```
hamkasblar = {
    'ali': {'familiya': 'valiyev',
            'tyil': 1995,
            'malumot': 'oliy',
            'tillar': ['python', 'c++']
    },
    'vali': {'familiya': 'aliyev',
            'tyil': 2001,
            'malumot': 'o'rta maxsus',
            'tillar': ['html', 'css', 'js']},
    'hasan': {'familiya': 'husanov',
            'tyil': 1999,
            'malumot': 'maxsus',
            'tillar': ['python', 'php']}
}
```

Hamkasblar to'g'risidagi ma'lumotlarni esa quyidagicha ko'rish mumkin:

```
for ism, info in hamkasblar.items():
    print(f"\n{ism.title()} {info['familiya'].title()}, "
          f"{info['tyil']}-yilda tug'ilgan.\n"
          f"Ma'lumoti: {info['malumot']}. \n"
          "Quyidagi dasturlash tillarini biladi:")
    for til in info['tillar']:
        print(til.upper())
```

Natija:

```
Ali Valiyev, 1995-yilda tug'ilgan.
Ma'lumoti: oliy.
Quyidagi dasturlash tillarini biladi:
PYTHON C++
Vali Aliyev, 2001-yilda tug'ilgan.
Ma'lumoti: o'rta maxsus.
Quyidagi dasturlash tillarini biladi:
HTML CSS JS
```

Hasan Husanov, 1999-yilda tug'ilgan.
Ma'lumoti: maxsus.
Quyidagi dasturlash tillarini biladi:
PYTHON PHP

Lug'at ichidagi lug'atlar bir xil tuzilishga ega bo'lgani ishingizni ancha yengillashtiradi, aks holda, kodingiz murakkablashib ketishi mumkin.

AMALIYOT

- Adabiyot (ilm-fan, san'at, internet) olamidagi 4 ta mashhur shaxlar haqidagi ma'lumotlarni lug'at ko'rinishida saqlang. Lug'atlarni bitta ro'yxatga joylang va har bir shaxs haqidagi ma'lumotni konsolga chiqaring.
- Yuqoridagi lug'atlarga har bir shaxsning mashhur asarlari ro'yxatini ham qo'shing. for sikli yordamida muallifning ismi va uning asarlarini konsolga chiqaring.
- Oila a'zolaringizdan (do'stlaringiz) 3 ta sevimli kino-seriali haqida so'rang. Do'stingiz ismi kalit, uning sevimli kinolarini esa ro'yxat ko'rinishida lug'atga saqlang. Natijani konsolga chiqaring.
- Davlatlar degan lug'at yaratib, lug'at ichida bir nechta davlatlar haqida ma'lumotlarni lug'at ko'rinishida saqlang. Har bir davlat haqida ma'lumotni konsolga chiqaring.
- Yuqoridagi dasturga o'zgartirish kiritib: konsolga barcha davlatlar emas, foydalanuvchi so'ragan davlat haqida ma'lumot berib. Agar davlat sizning lug'atingizda mavjud bo'lmasa, "Bizda bu davlat haqida ma'lumot yo'q" degan xabarni chiqaring.

6-BOB

while SIKLI

Biz 3-bobda for sikli yordamida kodning ma'lum bir qismini bir necha bor takrorlashni o'rgangan edik. Ushbu bobda esa while sikli bilan tanishamiz. Bu ikkisinin o'rtasidagi farq shundaki, for siklining takrorlanishi biror ro'yxat (lug'at, to'plam) elementlari soniga bog'langan bo'lsa, while tsiklining takrorlanishi biror shartning qanoatlanishiga bog'langan bo'ladi.

<https://cllco/Da-while>



① "While" so'zi ingliz tilidan "toki" deb tarjima qilinadi.

6.1. while QANDAY ISHLAYDI?

while sikli badandagi kodning necha marta takrorlanishi biror shartga bog'langan bo'ladi. Tushunarli bo'lishi uchun quyidagi misolni ko'zdan kechiramiz:

```
son = 1
while son<=5: # to'g'ri son 5 dan kichik yoki teng ekan...
    print(son, end=' ') # sonni konsolga chiqaramiz va
    son = son+1 # songa 1 ni qo'shamiz.
```

Natija:

1 2 3 4 5

Yuqoridagi dasturni tahlil qilamiz:

- avval son degan o'zgaruvchi yaratdik va unga 1 qiymatini berdik.

- 2-qatorida esa "toki son 5 dan kichik yoki teng ekan" keyingi qatorlarni bajar dedik.
- 3-qatorida sonni konsolga chiqardik.
- 4-qatorida songa 1 qo'shдик.
- 4-qatoridan so'ng kod yana 2-qatorga qaytadi va son<=5 shartini tekshiradi, agar shart bajarilsa (True), 3 – 4-qatorlar takrorlanadi.
- 5-qadamdan so'ng son=5 bo'lganida while sikli to'xtaydi.

Pythonda += operatori bor. Bu operator o'ng tarafdagi qiymatni chap tarafdagi qiymatga qo'shadi. Misol uchun, yuqorida son = son + 1 ifodasini son += 1 deb yozdik.

while va input()

while sikli yordamida dasturni to'xtatish imkoniyatini foydalanuvchiga berishimiz mumkin.

```
print("Kiritilgan sonning kvadratini qaytaruvchi dastur.")
savol = "Istalgan son kiriting "
savol += "(to'xtatish uchun 'exit' deb yozing): "
qiymat = ''
while qiymat != 'exit': # toki qiymat exitga teng emas ekan
    qiymat = input(savol)
    if qiymat != 'exit':
        print(float(qiymat)**2)
```

Yuqoridagi dasturimiz toki foydalanuvchi exit deb yozguniga qadar takrorlanaveradi.

Natija:

```
Kiritilgan sonning kvadratini qaytaruvchi dastur.
Istalgan son kiriting (to'xtatish uchun 'exit' deb yozing): 4
16.0
Istalgan son kiriting (to'xtatish uchun 'exit' deb yozing): 3
9.0
Istalgan son kiriting (to'xtatish uchun 'exit' deb yozing): 2
4.0
Istalgan son kiriting (to'xtatish uchun 'exit' deb yozing): exit
```

ISHORA – FLAG

Yuqoridagi misolda dasturni to'xtatish uchun yagona shartni tekshiridik (qiymat != 'exit'). Katta dasturlarda bir emas, bir neecha shartlarni tekshirish va ulardan biri bajarilgan taqdirda dasturni to'xtatish talab qilinishi mumkin.

Bunday holatlarda biror o'zgaruvchidan ishora (flag) sifatida foydalanishimiz mumkin. Agar dastur bajarilishi davomida dasturni to'xtatish shartlaridan biri bajarilganda ishora o'zgaruvchining qiymatini o'zgartiramiz va dastur o'z-o'zidan to'xtaydi.

```
print("Kiritilgan sonning kvadratini qaytaruvchi dastur.")
savol = "Istalgan son kiriting "
savol += "(to'xtatish uchun 'exit' deb yozing): "
ishora = True
while ishora: # toki ishora=True ekan
    qiymat = input(savol)
    if qiymat == 'exit':
        ishora = False
    else:
        print(float(qiymat)**2)
```

break OPERATORI

break operatori yordamida sikl bajarilishini sikl badanidan to'xtatishimiz mumkin. Ya'ni sikl to'xtashi while dan so'ng yozilgan shartga emas, sikl ichidagi boshqa shartga bog'lanishi mumkin:

```
print("Kiritilgan sonning kvadratini qaytaruvchi dastur.")
savol = "Istalgan son kiriting "
savol += "(to'xtatish uchun 'exit' deb yozing): "
while True: # abadiy sikl
    qiymat = input(savol)
    if qiymat == 'exit':
        break # siklni to'xtatish
    else:
        print(float(qiymat)**2)
```

break operatori for siklini to'xtatish uchun ham ishlatiladi.

```
sonlar = list(range(1,11))
for son in sonlar:
    if son == 5: # son 5 ga teng bo'lsa, sikl to'xtaydi
        break
    print(f"{son} ning kvadrati {son**2} ga teng")
```

Natija:

```
1 ning kvadrati 1 ga teng
2 ning kvadrati 4 ga teng
3 ning kvadrati 9 ga teng
4 ning kvadrati 16 ga teng
```

① Sikl ichida bir nechta break operatorlari bo'lishi mumkin.

continue OPERATORI

continue operatori esa, aksincha, ma'lum bir shart bajari-
ganda qadam tashlab o'tish uchun mo'ljallangan.

```
sonlar = list(range(1,11))
for son in sonlar:
    if son == 5: # son 5 ga teng bo'lsa, sikl boshiga qaytadi
        continue
    print(f"{son} ning kvadrati {son**2} ga teng")
```

Natijaga e'tibor bering, son 5 ga teng bo'lganda siklimiz
1 qadam tashlab o'tib ketdi va 5 ning kvadratini konsolga chi-
qarmadi:

```
1 ning kvadrati 1 ga teng
2 ning kvadrati 4 ga teng
3 ning kvadrati 9 ga teng
4 ning kvadrati 16 ga teng
6 ning kvadrati 36 ga teng
7 ning kvadrati 49 ga teng
8 ning kvadrati 64 ga teng
9 ning kvadrati 81 ga teng
10 ning kvadrati 100 ga teng
```

while sikliga ham misol ko'ramiz. Quyidagi kod 0 dan 10
gacha bo'lgan juft sonlarni konsolga chiqaradi:

```
son = 0
while son < 10:
    son += 1
    if son % 2 != 0: # agar son toq bo'lsa
        continue
    else: # aks holda (juft bo'lsa)
        print(son, end=' ')
```

Natija: 2 4 6 8 10

① Sikl ichida bir nechta continue operatorlari bo'lishi mumkin.

ABADIY SIKL TUZOG'I

Sikllar bilan ishlashda abadiy sikl yaratib qo'yishdan eh-
tiyot bo'lishimiz kerak. Abadiy siklga turli mantiqiy xatolar
sabab bo'lishi mumkin: noto'g'ri shart, o'zgarmas qiymat,
kodlar ketma-ketligida xatolik va hokazo.

① Dasturingiz abadiy siklga tushib qolganda dasturni to'xtatish uchun
konsolda (terminalda) Ctrl+C tugmalarini bosing.

Keling ba'zi misollarni ko'ramiz. **Diqqat, quyidagi kodni
ehtiyotkorlik bilan bajaring.**

```
son = 0
while son < 10:
    if son % 2 != 0:
        continue
    else:
        print(son)
```

Yuqoridagi kod abadiy davom etadi, sababi, biz sonning qiy-
matini o'zgartirishni esdan chiqardik.

Yana bir misol ko'ramiz:

```
son = 0
while son<10:
    if son%2!=0:
        continue
    else:
        print(son)
    son += 1
```

Bu kod ham abadiy davom etadi, sababini topa olasizmi?

```
son = 1
while son>0:
    son += 1
    if son%2!=0:
        continue
    else:
        print(son)
```

Yuqoridagi kodda xato shart tufayli ($son > 0$) sikl abadiy aylanadi.

AMALIYOT

- Foydalanuvchidan yaxshi ko'rgan kitoblarini kiritishni so'rang. Foydalanuvchi stop so'zini yozishi bilan dasturni to'xtating.
- Muzeyga chipta narxi foydalanuvchining yoshiga bog'liq: 7 dan yoshlarga 2000 so'm, 7 – 18 gacha 3000 so'm, 18 – 65 gacha 10000 so'm, 65 dan kattalarga bepul. Shunday while sikl yozingki, dastur foydalanuvchi yoshini so'rasin va chipta narxini chiqarsin. Foydalanuvchi exit yoki quit deb yozganda dastur to'xtasin (ikkita shartni ham tekshiring).
- Yuqoridagi dasturni turli usullarda yozib ko'ring (break, ishora yoki shart tekshirish).
- Quyidagi dasturda bir nechta mantiqiy xatolar bor. Jumladan, xususiy holatlarda sikl abadiy qaytarilib qolmoqda.

Xatolarni to'g'rilang.

```
savol = "Kiritilgan sonning ildizini qaytaruvchi dastur.\n"
savol += "Musbat son kiriting "
savol += "(dasturni to'xtatish uchun 'exit' deb yozing): "

while True:
    qiymat = input(savol)
    if qiymat<0:
        continue
    elif qiymat=='Exit':
        break
    else:
        ildiz = float(qiymat)**(0.5)
        print(f'{qiymat} ning ildizi {ildiz} ga teng")
```

6.2. while, RO'YXATLAR VA LUG'ATLAR

Ro'yxatlar (lug'atlar) bilan ishlashda while siklining foydalari juda ko'p. Misol uchun, foydalanuvchidan bir nechta ma'lumotlarni qabul qilib olish, ro'yxatdan takrorlanuvchi qiymatlarni o'chirib tashlash yoki bir ro'yxatni ikkinchi ro'yxatga ko'chirishda while siklidan foydalanish mumkin.



<https://click/Da-while2>

while YORDAMIDA RO'YXATNI TO'LDIRISH

Quyidagi dasturga e'tibor berib, avval ismlar degan bo'sh ro'yxat yaratib oldik. Keyin esa while sikli yordamida foydalanuvchidan ro'yxatga ism qo'shishni so'rayapmiz. So'ngra foydalanuvchidan yana ism qo'shmoqchi yoki yo'q ekanini so'raymiz va foydalanuvchining javobiga ko'ra yo sikl boshiga qaytamiz, yo siklni to'xtatamiz.

```

ismlar = []
print("Yaqin do'stlaringiz ro'yxatini tuzamiz.")
n=1 # ismlarni sanash uchun o'zgaruvchi
while True:
    savol = f"{n}-do'stingiz ismini kiriting:"
    ism = input(savol)
    ismlar.append(ism)
    javob = input("Yana ism qo'shasizmi? (ha/yo'q)")
    if javob == 'ha':
        n+=1
        continue
    else:
        break
print("Ro'yxat tuzildi")

```

Natija:

```

1-do'stingiz ismini kiriting:alijon
Yana ism qo'shasizmi? (ha/yo'q)ha
2-do'stingiz ismini kiriting:valijon
Yana ism qo'shasizmi? (ha/yo'q)yo'q
Ro'yxat tuzildi

```

Ro'yxat tarkibini ko'ramiz:

```

print("Do'stlaringiz ro'yxati:")
for ism in ismlar:
    print(ism.title())

```

Natija:

```

Do'stlaringiz ro'yxati:
Alijon
Valijon

```

while YORDAMIDA LUG'ATNI TO'LDIRISH

Yuqoridagi usul bilan while sikli yordamida lug'atlarni ham shakllantirishimiz mumkin. Quyidagi kodda ism bu kalit, yosh esa kalitga mos keluvchi qiymat. while siklining davomiyligi esa ishora ning qiymatiga bog'liq.

```

print("Do'stlaringiz yoshini saqlaymiz.")
dostlar = {}
ishora = True
while ishora:
    ism = input("Do'stingiz ismini kiriting: ")
    yosh = input(f"{ism.title()}ning yoshini kiriting: ")
    dostlar[ism] = int(yosh) # ism kalit, yosh qiymat
    javob = input("Yana ma'lumot qo'shasizmi? (ha/yo'q)")
    if javob == "yo'q":
        ishora = False

```

Dasturni bajaramiz:

```

Do'stlaringiz yoshini saqlaymiz.

Do'stingiz ismini kiriting: alijon
Alijonning yoshini kiriting: 30
Yana ma'lumot qo'shasizmi? (ha/yo'q)ha

Do'stingiz ismini kiriting: valijon
Valijonning yoshini kiriting: 37
Yana ma'lumot qo'shasizmi? (ha/yo'q)yo'q

```

Lug'atdagi qiymatlarni chiqaramiz:

```

for ism, yosh in dostlar.items():
    print(f"{ism.title()} {yosh} yoshda")

```

Natija:

```

Alijon 30 yoshda
Valijon 37 yoshda

```

RO'YXAT ELEMENTLARINI O'CHIRISH

Avvalgi darslarimizning birida ro'yxat elementini o'chirish uchun .remove(qiymat) metodi bilan tanishgan edik. Esingiz-da bo'lsa, bu metod ro'yxatdan eng birinchi uchragan qiymat-

ni o'chiradi. Agar ro'yxatimizda ma'lum bir qiymat bir necha bor takrorlangan bo'lsa, ularning barchasini o'chirish uchun while siklidan foydalanishimiz mumkin:

```
cars = ['lacetti', 'nexia', 'toyota', 'nexia', 'malibu', 'nexia']
while 'nexia' in cars:
    cars.remove('nexia')
print(cars)
```

Natija: ['lacetti', 'toyota', 'malibu']

Yuqoridagi sikl toki cars ro'yxatida nexia qiymati tugagunga qadar takrorlanaveradi.

RO'YXATDAN RO'YXATGA ELEMENT KO'CHIRISH

Tasavvur qiling, bizda ma'lum bir ro'yxat bor, biz ro'yxatdagi har bir element ustida biror amalni bajarib, uni birinchi ro'yxatdan ikkinchi ro'yxatga ko'chirib olmoqchimiz. Shunday holatlarda while sikli juda qo'l keladi.

Quyidagi misolni ko'raylik. Bizda talabalar ro'yxati bor. while sikli, toki ro'yxatda talabalar bor ekan, aylanaveradi. Sikl ichida biz .pop() metodi yordamida talabaning ismini ro'yxatdan sug'urib oldik va foydalanuvchidan talabani baholashni so'radik. Talabaning ismi va bahosini lug'at elementi ko'rinishida saqlab qo'ydik (talaba – kalit, baho – qiymat).

```
talabalar = ['hasan', 'husan', 'olim', 'botir']
baholangan_talabalar = {}
while talabalar:
    talaba = talabalar.pop()
    baho = input(f"{talaba.title()}ning bahosini kiriting: ")
    print(f"{talaba.title()} baholandi")
    baholangan_talabalar[talaba] = baho
```

Natija:

```
Botirning bahosini kiriting: 5
Botir baholandi
Olimning bahosini kiriting: 4
Olim baholandi
Husanning bahosini kiriting: 5
Husan baholandi
Hasanning bahosini kiriting: 3
Hasan baholandi
```

Yuqorida biz while sikli yordamida ro'yxat va lug'atlar ustida bajarish mumkin bo'lgan ba'zi misollarni ko'rdik. Albatta, dasturlash davomida bundan boshqa holatlar ham uchrashi tabiiy.

AMALIYOT

- Foydalanuvchidan buyurtma qabul qiluvchi dastur yozing. Mahsulotlar nomini birma-bir qabul qilib, yangi ro'yxatga joylang.
- e-bozor uchun mahsulotlar va ularning narxlari lug'atini shakllantiruvchi dastur yozing. Foydalanuvchidan lug'atga bir nechta elementlar (mahsulot va uning narxi) kiritishni so'rang.
- Yuqoridagi ikki dasturni jamlaymiz. Foydalanuvchi buyurtmasi ro'yxatidagi har bir mahsulotni e-bozordagi mahsulotlar bilan solishtiring (tayyor ro'yxat ishlatishingiz mumkin). Agar mahsulot e-bozorda mavjud bo'lsa, mahsulot narxini chiqaring, aks holda, "Bizda bu mahsulot yo'q" degan xabarni ko'rsating.

7-BOB

XATOLAR BILAN ISHLASH

Har qanday dasturchi kod yozishda xatoga yo'l qo'yadi. Ko'p yozgan odam ko'p xato qiladi va bu tabiiy. Ba'zi xatolarimiz Python tomonidan dastur bajarilishdan avval aniqlanadi. Ba'zilar esa dastur bajarish jarayonida aniqlanib, dasturimiz to'xtab qoladi.

Ushbu bobda dastavval dasturchilar ko'p yo'l qo'yadigan xatolar bilan tanishamiz. Keyin esa dastur bajarilishi davomida xatolik yuz berganda dastur to'xtab qolishining oldini olishni o'rganamiz.

<https://ol.co/da-xatolar>



7.1. XATOLARNING TURLARI

SyntaxError – SINTAKSIS XATOLIK

Biz bu xatolik turi bilan 2-bobda tanishgan edik. Syntax Error eng ko'p uchraydigan xato bo'lib, odatda, dasturlash tili qoidalariga amal qilmaslik natijasida keilib chiqadi. Aksar dasturlash muhitlari bunday xatolikni dastur bajarilishidan avval aniqlab, dasturchiga ishora beradi. Sintaksis xatolik bor dasturni Python bajarmaydi.

```
>>> print "Hello World!"
SyntaxError: Missing parentheses in call to 'print'.
Did you mean print("Hello World!")?
```

① Odatda, dasturlash muhiti xatoning turi bilan birga (SyntaxError) xato haqida qo'shimcha ma'lumot ham beradi: Missing parentheses in call to 'print'. Did you mean print("Hello World!")?

Xatolar bilan ishlashda xatolik matnini sinchkovlik bilan o'qish va tahlil qilish juda muhim. Agar ingliz tilini bilmasangiz, Google Translate yoki Yandex Tarjimon kabi onlayn xizmatlar yordamida xatolik matnini o'zingizga tushunarli (O'zbek yoki Rus) tilga tarjima qilib xatoni to'g'rilashingiz mumkin.

EOl va EOF xatolik

EOL (End of Line – qator yakuni) xatoligi sintaksis xatolikning bir turi bo'lib, odatda, qator oxirida qo'shtirnoqni (bir-tirnoq) yopish esdan chiqqanda yuzaga keladi.

```
>>> print("Hello World!
SyntaxError: EOL while scanning string literal
```

EOF (End of function – funksiya yakuni) xatoligi esa funksiya oxirida qavsni yopish esdan chiqqanda yuzaga keladi.

```
print("Hello World!"
SyntaxError: unexpected EOF while parsing
```

EOF xatoligining muammoli tarafi – Python aynan gaysi funksiya yopilmay qolganini ko'rsata olmaydi va dastur yakuniga ishora qiladi. Dasturingiz uzun bo'lsa, kodingizni sinchiklab ko'zdan kechirib chiqish talab etiladi.

IndentationError – JOY TASHLASHDA XATOLIK

Python tilida vaziyatga qarab kodni qator boshidan joy tashlab yoki tashlamadan yozish muhim ahamiyatga ega. Qator boshidan asossiz joy qoldirish IndentationError ga olib keladi.

Quyidagi kodga e'tibor berib, qator boshida 1 dona bo'sh joy qolgani uchun dasturlash muhiti xatolikni aniqlab, **X** bilan belgilab qo'ydi.

15 print("Hello World!")

15-RASM. IndentationError

Agar rasmdagi kodni bajaradigan bo'lsak, quyidagi natijani olamiz:

IndentationError: unexpected indent

Ba'zi joylarda esa, aksincha, bo'sh joy tashlab yozish talab qilinadi. Masalan, for siklida yoki if-elif-else shartlarining ichida va hokazo.

```
print("O'ngacha sanaymiz")
for n in range(10):
    print(n+1) # bu qator joy tashlab yozilishi kerak edi
```

Natija:

IndentationError: expected an indented block

Yana bir misol ko'ramiz:

```
son = 50
if son>=0:
    print("Musbat son")
else:
    print("Manfiy son")
```

Natija:

IndentationError: expected an indented block

Qancha joy tashlaymiz?

Yuqoridagi misollarda IndentationError oldini olish uchun joy tashlash talab qilindi. Xo'sh, qancha joy tashlash kerak va qanday qilib?



16-RASM. Space va Tab tugmalari

Aslida, hech bo'lmaganda 1 harfli bo'sh joy qoldirish ham bizni IndentationError dan xalos etadi. LEKIN biz dastur davomida bir xil joy tashlashga odatlanishimiz kerak.

Qoida sifatida kamida 4 harfli joy yoki 1 ta TAB (klaviatura) radagi tab tugmasi joy tashlashga odatlanishimiz zarur. Eng muhimi, bir dastur (loyiha) davomida ikkalasini aralashtirmaslik lozim. Ya'ni agar joy tashlash uchun Space (probel) tugmasini ishlatsangiz, oxirigacha shunday qiling, agar Tab ishlatsangiz, oxirigacha tab ishlatib. Ikkalasini aralashtirmang!

RUN TIME ERROR – DASTURNI BAJARISHDA XATOLIK

Run time error dastur bajarish jarayonida kelib chiqadi va dasturning ishlashini to'xtatadi. Sintaksis xatolikdan farqli ravishda, Python bunday xatolarni dasturni bajarish-

dan avval aniqlay olmaydi. Bu xatolikning bir necha turi bor. Ulardan ba'zilari bilan tanishamiz.

TypeError

Biror funksiya yoki metodga noto'g'ri ma'lumot turini yuborish:

```
son = input("Istalgan son kiriting: ")
print(f"{son} ning kvadrati {son**2} ga teng")
```

Natija:

```
TypeError: unsupported operand type(s) for ** or pow():
'str' and 'int'
```

Yuqoridagi kodda biz foydalanuvchi kiritgan qiymatni matndan songa o'tkazib olishni unuttik, natijada sonning kvadratini hisoblashda Python xato berdi.

NameError

O'zgaruvchi, funksiya, obyekt nomini noto'g'ri yozish natijasida kelib chiquvchi xatolik.

```
prit("Hello World!")
```

Natija:

```
NameError: name 'prit' is not defined
```

Yuqorida print() funksiyasini prit() deb xato yozganimiz sababli Python bunday funksiya yo'q deb xato qaytardi.

Yana bir misol ko'ramiz:

```
mevalar = ['olma', 'uzum', 'nok', 'anor', 'anjir']
for meva in mvealar:
    print(meva)
```

Natija:

```
NameError: name 'mvealar' is not defined
```

Xatoni topdingizmi?

ValueError

Funksiyaga noto'g'ri qiymatni yuborish natijasidagi xatolik

```
son = int(input("Istalgan son kiriting: "))
if son >= 0:
    print("Musbat son")
else:
    print("Manfiy son")
```

Kodni bajaramiz:

```
Istalgan son kiriting: 5.5
ValueError: invalid literal for int() with base 10: '5.5'
```

Yuqoridagi dasturning 1-qatorida foydalanuvchidan istalgan son kiritishni so'radik va foydalanuvchi kiritgan qiymatni int, ya'ni butun songa o'tkazdik. Kodning o'zida xato yo'q, lekin dastur bajarish jarayonida foydalanuvchi butun son emas, o'nlik son kiritgani uchun ValueError xatosi chiqdi. Sababi, int() funksiyasi faqatgina butun sonlar ko'rinishidagi matn bilan ishlaydi.

Dastur xato bermasligi uchun yoki int() funksiyasini float() funksiyasiga almashtirish yoki foydalanuvchidan butun son kiritishni talab qilishimiz kerak.

IndexError

Yangi dasturchilar yo'l qo'yadigan yana bir xato indeks xatolikdir. Ya'ni ro'yxat elementlariga murojaat etishda in-

deksni noto'g'ri kiritish (ro'yxatda mavjud bo'lmagan elementga murojaat etish):

```
mevalar = ['olma', 'anor', 'uzum']  
print(mevalar[3])
```

Natija:

```
IndexError: list index out of range
```

Natijani tahlil qilamiz. Bizda uchta elementdan iborat mevalar ro'yxati mavjud. Biz 3-elementni konsolga chiqarmoqchimiz va `print(mevalar[3])` deb yozdik va `IndexError` natijasini oldik. Sababi, dasturlashda indeks 0 dan boshlanadi va 3-elementga murojaat etish uchun 2-indeksni tanlaymiz. Demak, to'g'ri kod:

```
>>> print(mevalar[2])  
uzum
```

ZeroDivisionError

Dastur jarayonida 0 ga bo'lish natijasida yuzaga keladigan xatolik

```
>>> x, y = 50, 50  
>>> z = 250/(x-y) # x-y=0  
ZeroDivisionError: division by zero
```

MANTIQUIY XATOLAR

Mantiqiy xatolar – dasturchi tomonidan dastur algoritmini yozishda yo'l qo'yilgan xatolar. Bunday xatolar dastur bajarilishiga to'sqinlik qilmasa-da, dastur to'g'ri ishlashiga xalaqit qiladi va kutilgan natijani bermaydi.

Mantiqiy xatolar eng ko'p uchraydigan va aniqlash qiyin bo'lgan xatolar hisoblanadi. Aksar holatlarda Python mantiqiy xatolarini aniqlamaydi va dastur bajarilaveradi (lekin kutilgan natija chiqmaydi). Bunday xatolarning oldini olish uchun dasturimizning har bir qadamida chiqayotgan natijalarni tekshirib borish kerak. Katta dasturlar o'rnlab, balki yuzlab funksiyalardan iborat bo'ladi, agar mantiqiy xatolar vaqtida topilmasa, minglab qator kodlarni ko'zdan kechirib chiqish talab etilishi mumkin.

Mantiqiy xatolar turli ko'rinishda bo'lishi mumkin, masalan, sonlar bilan ishlashda:

```
radius = 5  
pi = 4.14  
aylana_yuzi = pi*radius**2  
print(aylana_yuzi)
```

Natija:

```
103.49999999999999
```

Kod bajarildi, natija ham chiqdi. Lekin natija xato. Nima uchun? Sababi, biz **$\pi=4.14$** deb xato yozib ketdik (to'g'ri qiymat: $\pi=3.14$).

Avval aytganimizdek, mantiqiy xatolarning oldini olish uchun dasturimizning har bir qadamida chiqayotgan natijalarni tahlil qilib borish muhim. Sodda dasturlar qo'lida bajariladi, murakkabroq funksiyalar uchun esa maxsus test dasturlar yoziladi. Test dasturlar yozish haqida kelgusi boblarda alohida gaplashamiz. Hozir esa qo'lida tekshirishni ko'raylik:

```
son = float(input("Istalgan son kiriting: "))  
ildiz = son**1/2  
print(f"{son} ning ildizi {ildiz} ga teng")
```

Natija:

```
Istalgan son kiriting: 4
4.0 ning ildizi 2.0 ga teng
```

Dasturimiz biz kutgan natijani berdi. Lekin yaxshi dasturchi bitta test bilan chegaralanmaydi. Dasturni qayta bajarimiz:

```
Istalgan son kiriting: 9
9.0 ning ildizi 4.5 ga teng
```

Mana endi natija xato, 9 sonining ildizi 4.5 bo'lib chiqdi. Kodni tahlil qilamiz: 2-qatorida ildizni hisoblashda $1/2$ qavs ichida yozilmagani xatolikka olib kelyapti. Natijada foydalanuvchi kiritgan son avval 1-darajaga oshirildi va undan keyin 2 ga bo'lini. Kodni to'g'rilaymiz va dasturni 3-4 marta turli qiymatlar berib bajarib ko'ramiz:

```
son = float(input("Istalgan son kiriting: "))
ildiz = son**(1/2)
print(f"{son} ning ildizi {ildiz} ga teng")
```

Natija:

```
Istalgan son kiriting: 9
9.0 ning ildizi 3.0 ga teng
Istalgan son kiriting: 1024
1024.0 ning ildizi 32.0 ga teng
```

Ba'zida noo'rin bo'sh joy qoldirish (yoki goldirmaslik) ham mantiqiy xatoga olib kelishi mumkin:

```
mevalar = ['olma', 'uzum', 'nok', 'anor', 'anjir']
for meva in mevalar:
    print(meva)
print("Dastur tugadi")
```

Natija:

```
olma
Dastur tugadi
uzum
Dastur tugadi
nok
Dastur tugadi
anor
Dastur tugadi
anjir
Dastur tugadi
```

Yuqorida "Dastur tugadi" matni bir marta, dastur tugaganidan so'ng chiqishi kerak edi. Lekin o'ngga surilib qolgani uchun bir necha bor qaytarildi.

Bundan boshqa ham mantiqiy xatoliklar juda ko'p uchraydi. Mantiqiy xatoliklar mutlaqo topilmasdan ham qolib ketishi va dastur bozorga chiqqanidan so'ng aniqlanishi tabiiy hol. Shuning uchun ham aksar dasturlar tez-tez yangilanib turadi. Dasturlash jarayonida bundan boshqa xatoliklar ham ko'p uchraydi. Biz ulardan ba'zilari bilan tanishdik, xolos. Keyingi bo'limda Runtime xatoliklarni dastur davomida aniqlash va dastur to'xtab qolishining oldini olishni o'rganamiz.

AMALIYOT

Quyidagi dasturlarda bir nechta xatolar mavjud. Ularni to'g'rilang:

```
son = float(input("Juft son kiriting: "))
if son%2==0:
    print("Bu son juft emas.")
else:
    print("Rahmat!")
```



```

yosh = (input("Yoshingiz nechida?"))

if yosh<=4 or yosh>=60:
    narx = 0
elif yosh < 18
    narx = 10000
else:
    narx = 20000
print(f"Chipta {narx} so'm")

```

```

x = float(input("Birinchii sonni kiriting: "))
y = float(input("Ikkinchi sonni kiriting: "))
if x==y:
    print(f"x={y}")
elif x<y:
    print(f"x<y")
else
    print(f"x>y")

```

```

mahsulotlar = ['un', "yog'", "sovun", 'tuxum', 'piyoz',
'kartoshka', 'olma', 'banan', 'uzum', 'qovun'
for n in range(5):
    savat.append(input(f"{n+1}-mahsulotni qo'shing: "))
if savat:
    for mahsulot in savat
        if mahsulot in mahsulotlar:
            print(f"Do'konimizda {mahsulot} bor")
        else:
            print(f"Do'konimizda {mahsulot} yo'q")
else:
    print("Savatingiz bo'sh")

```

```

users = ['alisher1983', 'aziza', 'yasina', 'umar']
login = input("Yangi login tanlang: ")
if login in users:
    print('Login band, yangi login tanlang!')
else:
    print("Xush kelibsiz!")

```

7.2. EXCEPTIONS

Avvalgi bo'limda "Run time error" xatoliklari bilan tanishdik. Bunday xatolar dastur bajarish jarayonida kelib chiqadi va dasturning ishlashini to'xtatadi. Sintaksis xatolikdan farqli ravishda, Python bunday xatolarni dasturni bajarishdan avval aniqlay olmaydi. Ushbu darsimizda mana shunday xatoliklarni jilovlashni o'rganamiz. Maqsadimiz – xatolik yuz berganda dastur to'xtab qolishining oldini olish. Gap shundaki, dastur davomida xato yuz berganda Python maxsus exception (istisno) obyektini yaratadi. Agar bu obyekt "tutib" olinmasa, dastur bajarilishdan to'xtaydi.

try-except

Istisno obyektlarni tutib olish uchun Pythonda maxsus try-except operatorlari bor. Bu operatorlar quyidagicha ishlaydi, try operatori badanida bajarish kerak bo'lgan kod yoziladi, except operatori badanida esa xatolik yuz berganda bajarilishi kerak bo'lgan kod yoziladi. Ya'ni dasturimiz to'xtab qolmasdan bajarilaveradi.

Tushunarli bo'lishi uchun quyidagi misolni ko'ramiz.

```

yosh = input("Yoshingizni kiriting: ")
yosh = int(yosh)
print(f"Siz {2021-yosh} yilda tug'ilgansiz")

```

Yuqoridagi dasturning 1-qatorida biz foydalanuvchidan yoshini kiritishni so'radik. Navbatdagi qatorda esa foydalanuvchi kiritgan qiymatni int() funksiyasi yordamida butun songa o'tkazayapmiz. Agar foydalanuvchi yoshini kiritganda butun emas, o'nlik son kiritisa bu ValueError xatoligiga olib keladi va dastur bajarilishdan to'xtaydi.



<https://clic.co/Da-try>

```
Yoshingizni kiriting: 32.5
ValueError: invalid literal for int() with base 10: '32.5'
```

Mana shunday xatolarning oldini olish uchun xato yuz berishi mumkin bo'lgan qatorlarni try-except bloki yordamida yozamiz. Bunda try bloki ichida bevosita xato keltirib chiqarishi mumkin kodni, except bloki ichida esa xatolik yuz berganda bajarilishi kerak bo'lgan buyruqlarni yozamiz.

Tushunarli bo'lishi uchun yuqoridagi dasturni yangidan yozamiz:

```
yosh = input("Yoshingizni kiriting: ")
try:
    yosh = int(yosh) # xato qaytargan qator
    print(f"Siz {2021-yosh} yilda tug'ilgansiz")
except: # xato yuz bergan bajariluvchi kod
    print("Butun son kiritmadingiz")

print("Dastur Tugadi!")
```

Bu yerda ham dastavval foydalanuvchi yoshini so'radik. int() funksiyasini esa try badani ichida yozdik, agar foydalanuvchi to'g'ri qiymat kiritgan bo'lsa, kodimiz foydalanuvchining tug'ilgan yilini hisoblab ko'rsatadi, exception (istisno) yuz berganda esa "Butun son kiritmadingiz" xabarini konsolga chiqaradi. Lekin dastur bajarilishdan to'xtamaydi va try-except blokidan keyingi qatorlar ham bajarilaveradi (print("Dastur Tugadi!")).

Dasturni tekshiramiz. Boshlanishiga to'g'ri qiymat kiritaylik:

```
Yoshingizni kiriting: 25
Siz 1996 yilda tug'ilgansiz
Dastur Tugadi!
```

Yana bir bor, lekin bu safar o'nlik son kiritamiz:

```
Yoshingizni kiriting: 32.5
Butun son kiritmadingiz
Dastur Tugadi!
```

try-except operatorining afzalliklaridan biri – foydalanuvchiga Python ko'rsatadigan tushunarsiz xatolar o'rniga o'zimiz istagan, tushunarliroq matnni ko'rsatish imkonini beradi. Shuningdek, kompleks tizimlarda arzimagan xatoni deb dasturimiz to'xtab qolishining oldini oladi.

try-except-else

Yuqoridagi kodimizda biz try moduli ichida xato qaytarishi mumkin bo'lgan ifodani ham (tyil = int(tyil)), xato qaytmaganda bajarilishi kerak bo'lgan ifodani ham (print(f"Siz {2021-tyil} yoshdasiz")) birdan yozib ketyapmiz. Aslida, bunday qilishimiz to'g'ri emas.

To'g'ri usuli – avval xatoga tekshirish va xato yuz bergandanda bajariladigan ifodani alohida, else blokida yozish:

```
yosh = input("Yoshingizni kiriting: ")
try:
    yosh = int(yosh)
except:
    print("Butun son kiritmadingiz")
else:
    print(f"Siz {2021-yosh} yilda tug'ilgansiz")
```

Natija:

```
Yoshingizni kiriting: 20
Siz 2001 yilda tug'ilgansiz
```

Albatta, yuqoridagi usul har doim ham qo'l kelavermaydi.

MALUM TURDAGI XATOLARNI USHLASH

Xatolarning turlari ko'p, except operatori yordamida biz aynan qaysi xatolarni ushlamoqchi ekanimizni ham ko'rsatishimiz mumkin. Misol uchun, yuqoridagi misolda `int()` funksiyasi `ValueError` xatosini qaytardi. Agar biz faqatgina shu turdagi xatolarni ushlamoqchi bo'lsak, except operatoridan so'ng xatolik nomini ham ko'rsatamiz:

```
yosh = input("Yoshingizni kiriting: ")
try:
    yosh = int(yosh)
except ValueError:
    print("Butun son kiritmadingiz")
else:
    print(f"Siz {2021-yosh} yilda tug'ilgansiz")
```

Shu yo'sinda boshqa turdagi xatolarni ham tutib olish mumkin.

ZeroDivisionError

```
x, y = 5, 10
try:
    y/(x-5)
except ZeroDivisionError:
    print("0 ga bo'lib bo'lmaydi")
```

Natija: 0 ga bo'lib bo'lmaydi

IndexError

```
mevalar = ['olma', 'anor', 'anjir', 'uzum']
try:
    print(mevalar[7])
except IndexError:
    print(f"Ro'yxatda {len(mevalar)} ta meva bor xolos")
```

Natija: Ro'yxatda 4 ta meva bor xolos

KeyError

Bu xatolik lug'atda mavjud bo'lmagan kalitga murojaat etishda kelib chiqadi:

```
user = {"username": "sariqdev",
        "status": "admin",
        "email": "admin@sariq.dev",
        "phone": "99897123456"}
key = "tel"
try:
    print(f'Foydalanuvchi: {user[key]}')
except KeyError:
    print("Bunday kalit mavjud emas")
```

Natija: Bunday kalit mavjud emas

FileNotFoundError

Bu xatolik Pythonda fayllar bilan ishlashda mavjud bo'lmagan faylga murojaat etish ortidan kelib chiqadi. Biz fayllar bilan ishlash haqida kelgusi boblarda to'xtalamiz, bu bo'limda esa shunday xatolikni ushlashni ko'ramiz, xolos:

```
fayl = "data.txt" # bunday fayl mavjud emas
try:
    # faylni ochishga harakat qilamiz
    f = open(fayl)
except FileNotFoundError:
    print(f"{fayl} fayli mavjud emas.")
```

Natija: data.txt fayli mavjud emas.

BIR NECHTA XATOLARNI USHLASH

`try-except` ketma-ketligida bir nechta `except` operatorlari ham bo'lishi mumkin. Ularning har biri ma'lum turdagi xatolik uchun javobgar bo'ladi:

```

n = input("Butun son kiriting: ")
try:
    n = int(n)
    x=20/n
except ValueError: # agar foydalanuvchi butun son kiritmasa,
    print("Butun son kiritmadingiz")
except ZeroDivisionError: # agar foydalanuvchi 0 kiritrsa,
    print("0 ga bo'lib bo'lmaydi")
else:
    print(f"x={x}")

```

XATOLARNI KO'RSATMAY O'TISH

Yuqoridagi misollarda kodimiz xato qaytarganida dasturimiz foydalanuvchiga xatolik haqida ma'lumot ko'rsatyapti. Agar dastur hech qanday ma'lumot ko'rsatmay, dasturni davom etishi talab qilinsa, except badanida pass operatorini yozamiz.

```

user = {"username": "sariqdev",
        "status": "admin",
        "email": "admin@sariq.dev",
        "phone": "99897123456"}

key="tel"
try:
    print(f'Foydalanuvchi: {user[key]}')
except KeyError:
    pass

```

Umuman olganda, pass hech narsa bajarmasa-da, juda foydali operator. Bu operator yordamida biz funksiyalar, sikllar yoki if-else shartlarining badanini vaqtincha to'ldirib turishimiz mumkin. Misol uchun, siz dasturning skeletini o'ylab qo'yingiz, qanday sikllar yoki shartlar bo'lishini reja qildingiz, lekin bevosita sikl yoki funksiya badanini yozishga yetib kelmagan bo'lsangiz, IndentationError xatoligining oldini olish uchun pass operatoridan foydalanasiz.

```

if yosh<20:
    pass
else:
    pass

```

Yuqoridagi kodni pass operatorisiz yozib ko'ring.

7.3. XATOLARNING OLDINI OLISH

Yuqorida biz xatolar ro'y berganda ularni ushlash va dastur to'xtashining oldini olishni ko'rib chiqdik. Lekin try-except ketma-ketligi xatolarning oldini olishga yordam bera olmaydi. Xatolarning oldini olish uchun if-else va while sikllaridan foydalanganimiz afzal.

Avvalgi bo'limdagi misolimizga qaytsak:

```

yosh = input("Yoshingizni kiriting: ")
try:
    yosh = int(yosh) # xato qaytargan qator
    print(f"Siz {2021-yosh} yilda tug'ilgansiz")
except: # xato yuz bergan bajariluvchi kod
    print("Butun son kiritmadingiz")

```

Biz foydalanuvchi yoshini so'radik va foydalanuvchi butun son kiritmagani sababli dasturni to'xtatdik. Aslida, bunday holatda while sikli yordamida foydalanuvchi to'g'ri qiymat kiritgunga qadar uning yoshini qayta so'rash to'g'riroq yechim bo'ladi:

```

while True:
    yosh = input("Yoshingizni kiriting: ")
    if yosh.isdigit():
        yosh = int(yosh)
        break
    print(f"Siz {2021-yosh} yilda tug'ilgansiz")

```

E'tibor bering, foydalanuvchi kiritgan matn faqatgina sonlardan iborat ekanligini tekshirish uchun .isdigit() metodi-dan foydalandik.

Dasturni bajaramiz va boshlanishiga turli "noto'g'ri" qiymatlar berib ko'ramiz:

```
Yoshingizni kiriting: 12.5
Yoshingizni kiriting: 23.5
Yoshingizni kiriting: 22.5
Yoshingizni kiriting: 24
Siz 1997-yilda tug'ilgansiz
```

Ko'rib turganingizdek, biz dastavval 12.5 qiymatini kiritdik va matn tarkibida nuqta belgisi bo'lgani uchun `.isdigit("12.5")` metodi `False` qaytardi va siklimiz boshiga qaytdi. Biz yana 2 marta xato qiymat kiritdik va har gal sikl boshidan takrorlandi. 4-urinishda 24 qiymatini kiritganimiz sababli `.isdigit("24")` metodi `True` qaytardi, siklimiz to'xtadi va "Siz 1997-yilda tug'ilgansiz" degan natija konsolga chiqdi.

Albatta, yuqoridagi usul barcha xatolar uchun ishlamaydi. Shunday xatolar bo'lishi mumkin, biz ularni oldindan ko'ra olmasligimiz, oldindan to'g'ri olmay olmasligimiz mumkin yoki xato foydalanuvchiga bog'liq bo'lmashi mumkin. Shunday holatlarda `try-except` operatorlari bizning xaloskorimizga aylanadi.

AMALIYOT

- Quyidagi kod bajarilishida yuzaga kelishi mumkin bo'lgan xatolarni ushlab, xatoga mos matn konsolga chiqaring:

```
x = int( input( "son kiriting: " ) )
y = int( input( "yana bir son kiriting: " ) )
print( x, '/', y, '=', x/y )
```

- Yuqoridagi kodni while sikli yordamida xato qiymat kiritilganda takrorlanadigan qiling.

8-BOB

FUNKSIYA

Funksiya ma'lum bir vazifani bajarishga mo'ljallangan kodlar yig'indisidir. Funksiyalar, odatda, ma'lumotlarni qabul qilib oladi, ularni qayta ishlaydi va biron natija qaytaradi. Biz darslarimiz davomida bir nechta tayyor funksiyalardan foydalalanib keldik. Misol uchun, `print()` funksiyasi matn yoki ifoda qabul qiladi va uni konsolga chiqaradi, `range()` funksiyasi esa ma'lum oralikdagi sonlarni yaratish uchun ishlatiladi.

Aslida, har qanday funksiyaning ortida ham bir nechta qatordan iborat kod bo'ladi, lekin biz funksiyaga murojaat etganda uning nomini yozamiz, xolos. Funksiya ortidagi kod esa biz uchun yashirin bo'lib qolaveradi. Funksiyalarning qulayligi ham shunda. Dastur davomida ma'lum bir kodlarni qayta-qayta yozmaslik uchun biz ularni jamlab, bitta funksiya ichiga joylashimiz va dastur davomida bu kodlarga funksiya nomi orqali murojaat etishimiz mumkin.

Funksiyalar turlicha bo'ladi, ba'zi funksiyalar sizdan qiymat qabul qilib, konsolga biror ma'lumot chiqaradi, ba'zilari esa sizdan qabul qilgan qiymat ustida turli amallar bajarib yangi qiymat qaytaradi. Foydalanuvchidan mutlaqo qiymat qabul qilmaydigan funksiyalar ham mavjud.

Ushbu bobda biz Python'da yangi funksiya yaratish, unga murojaat etish, tekshirish va to'g'rilashni o'rganamiz. Shuningdek, darsimiz yakunida dasturimizni bir nechta fayllarga



<https://olco/Da-tunc>

Funksiya yaratishning asl maqsadlaridan biri – biz unga qayta-qayta, yangi qiymatlar bilan murojaat etishimiz mumkin.

```
>>> salom_ber('hasan')
Assalomu alaykum, hurmatli Hasani
>>> salom_ber('olim')
Assalomu alaykum, hurmatli Olimi
```

DOCSTRING

Funksiyani yaratishda funksiya qanday ishlashi haqida qisqacha ma'lumot berib ketish o'zimiz uchun ham, kelajakda bizning funksiyamizni ishlatadigan boshqa dasturchilar uchun ham juda foydali bo'ladi. Bunday ma'lumotlar **docstring** deb ataladi va funksiya badanining ilk qatorida uchta qo'shtirnoq ichida yoziladi:

```
def salom_ber():
    """Salom beruvchi funksiya""" # <-Docstring
    print("Assalomu alaykum!")
```

Murakkab funksiyalar uchun docstringni bir necha qatorga bo'lib yozish mumkin:

```
def salom_ber(ism):
    """Foydalanuvchi ismini qabul qilib,
    unga salom beruvchi funksiya"""
    print(f"Assalomu alaykum, hurmatli {ism.title()}!")
```

Docstringni konsolga chiqarish uchun quyidagi buyruqni yozamiz: `print(funksiya_nomi.__doc__)`

```
>>> print(salom_ber.__doc__)
Foydalanuvchi ismini qabul qilib,
unga salom beruvchi funksiya
```

Odatda, dasturlash muhitlari funksiya nomini yozishingiz bilan docstring ichidagi matnни ko'rsatadi:

```
19
20 salom_ber()
```

```
salom_ber(ism)
Foydalanuvchi ismini qabul qilib,
unga salom beruvchi funksiya
```

17-RASM. DOCSTRING

8.2. ARGUMENT VA PARAMETR

Funksiya yaratishda gavs ichida berilgan, funksiya to'g'ri ishlashi uchun uzatiladigan qiymat **parametr** deb ataladi. Sodda bir misol keltiramiz:

```
def salom_ber(ism):
    """Foydalanuvchi ismini qabul qilib,
    unga salom beruvchi funksiya"""
    print(f"Assalomu alaykum, hurmatli {ism.title()}!")
```

Yuqoridagi misolda **ism** – salom ber funksiyasining parametri. Parametrlarga mazmunli nom berishni odat qiling:

- ✗ `def salom_ber(n)`
- ✓ `def salom_ber(name)`
- ✓ `def salom_ber(ism)`

Foydalanuvchi funksiyaга murojaat etishda funksiyaга uzatgan qiymat **argument** deb ataladi. `salom_ber('hasan')` kodida **'hasan'** – argument.

Demak, parametr va argument bitta narsaga ikki xil qarash ekan, xolos.

① Ba'zi manbalarda argument va parametr so'zlari almashitirib ishlatilishi ham kuzatiladi.

FUNKSIYAGA BIR NECHTA ARGUMENT UZATISH

Ba'zi funksiyalar bir emas, bir nechta parametr talab qilishi mumkin, foydalanuvchi esa, o'z navbatida, bir nechta argumentlar taqdim qilishi kerak. Funksiyaga argument uzatishning bir necha usuli bor, keling, ular bilan birma-bir tanishamiz.

To'g'ri tartibda uzatish

Bu usulda funksiya parametrlari qaysi tartibda yozilgan bo'lsa, argumentlar ham aynan shu ketma-ketlikda uzatilishi shart. Bitta misol keltiramiz. Quyidagi funksiya foydalanuvchining ismi va familiyasini parametr sifatida qabul qilib, ularni jamlab xabar chiqaradi.

```
def toliq_ism(ism, familiya):  
    """ism va familiyani jamlab chiqaruvchi funksiya"""  
    print(f"Foydalanuvchi ismi: {ism.title()}\n"  
          f"Foydalanuvchi familiyasi: {familiya.title()}")
```

Yuqoridagi funksiya to'g'ri natija chiqarishi uchun argumentlarni ism va familiya ketma-ketligida kiritishimiz lozim.

```
>>> toliq_ism('olim', 'hakimov')  
Foydalanuvchi ismi: Olim  
Foydalanuvchi familiyasi: Hakimov
```

Agar argumentlarni noto'g'ri ketma-ketlikda bersak, natija ham biz kutganday chiqmaydi:

```
toliq_ism('hakimov', 'olim')  
Foydalanuvchi ismi: Hakimov  
Foydalanuvchi familiyasi: Olim
```

Ko'p holatlarda esa argumentlarni noto'g'ri tartibda uzatish xatolikka ham olib kelishi mumkin.

```
def yosh_hisobla(ism, tugilgan_yil):  
    """Foydalanuvchi yoshini hisoblaydigan funksiya"""  
    print(f"{ism.title()} {2020-tugilgan_yil} yoshda")
```

Funksiyani chaqiramiz:

```
>>> yosh_hisobla('olim', 1997)  
Olim 23 yoshda
```

Endi argumentlarni o'rinni almashtirib yozib ko'ramiz:

```
>>> yosh_hisobla(1997, 'olim')  
AttributeError: 'int' object has no attribute 'title'
```

Parametr nomi bilan uzatish

Yuqoridagi kabi holatlarning oldini olish uchun argumentlarni parametr nomi bilan qo'shib uzatishimiz mumkin. Buning uchun funksiya o'zgartirish kiritish talab qilinmaydi.

```
>>> yosh_hisobla(tugilgan_yil=1997, ism='olim')  
Olim 23 yoshda
```

Yuqoridagi misolda funksiyaning chaqirishida biz parametrlar ketma-ketligiga rioya qilmagan bo'lsa-da, argumentlarni parametr=qiyamat ko'rinishida yozganimiz sababli funksiya to'g'ri ishladi.

Xuddi shu kabi yuqoridagi toliq_ism funksiyasiga murojaat etishimiz mumkin:

```
>>> toliq_ism(familiya='hakimov', ism='olim')  
Foydalanuvchi ismi: Olim  
Foydalanuvchi familiyasi: Hakimov
```

△ Bu usuldan foydalanganda parametr nomi to'g'ri yozilishiga e'tibor bering!

STANDART QIYMAT

Funksiya yaratishda istalgan parametr uchun standart qiymat ko'rsatib ketishimiz mumkin. Agar foydalanuvchi shu parametr uchun qiymat (argument) kiritmasa, funksiya bajarilishi jarayonida standart qiymat ishlatiladi. Standart qiymat funksiya yaratish vaqtida parametr = qiymat ko'rinishida beriladi.

Quyidagi misolda yosh_hisobla funksiyasida joriy_yil parametriга standart qiymat beramiz:

```
def yosh_hisobla(tugilgan_yil, joriy_yil=2021):  
    print(f"Siz {joriy_yil-tugilgan_yil} yoshdasiz")
```

① Funksiya yaratishda standart qiymatga ega parametrlar doim oxirida yozilishi kerak. Aks holda, xatolik yuzaga keladi.

Keling, avval funksiyani ikkala argument bilan chaqiramiz:

```
>>> yosh_hisobla(1995, 2020)  
Siz 25 yoshdasiz
```

Endi esa faqat bitta argument (tugilgan_yil) bilan chaqiramiz:

```
>>> yosh_hisobla(1993)  
Siz 28 yoshdasiz
```

Bu safar foydalanuvchi joriy_yil ni kiritmagani sababli standart qiymat – 2021 ishlatildi.

FUNKSIYAGA MUHOJAAT ETISHDA XATOLIKLAR

Funksiyalarga murojaat etishda turli xatoliklarga yo'l qo'yishimiz tabiiy. Bunday holatlarda Python qaytargan xatoni sinchiklab o'qib, xato qayrataligini topishimiz va uni to'g'rilashimiz zarur. Quyida men turli funksiyalar yaratib,

ularga xato usullar bilan murojaat etaman. Xatolar nimada ekanini topa olasizmi?

1-misol

```
def yosh_hisobla(tugilgan_yil, joriy_yil=2020):  
    """Tug'ilgan yildan yoshini hisoblaymiz"""  
    print(f"Siz {joriy_yil-tugilgan_yil} yoshdasiz")  
  
tyil = input("Tug'ilgan yilingizni kiritting: ")  
yosh_hisobla(tyil)
```

Natija:

```
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

2-misol

```
def yosh_hisobla(tugilgan_yil, joriy_yil):  
    print(f"Siz {joriy_yil-tugilgan_yil} yoshdasiz")  
  
yosh_hisobla(1993)
```

Natija:

```
TypeError: yosh_hisobla() missing 1 required positional argument: 'joriy_yil'
```

3-misol

```
def salom_ber():  
    """Salom beruvchi funksiya"""  
    print("Assalomu alaykum!")  
  
salom_ber('hasan')
```

Natija:

```
TypeError: salom_ber() takes 0 positional arguments  
but 1 was given
```


4-misol

```
def toliq_ism(ism, familiya):  
    print(f"Foydalanuvchi ismi: {ism.title()}\n"  
          f"Foydalanuvchi familiyasi: {familiya.title()}")  
    toliq_ism('olim hakimov')
```

Natija:

```
TypeError: toliq_ism() missing 1 required positional  
argument: 'familiya'
```

AMALIYOT

- Foydalanuvchi ismi va yoshini so'rab, uning tug'ilgan yilini hisoblaydigan funksiya yozing.
- Foydalanuvchidan son olib, uning kvadrati va kubini konsolga chiqaruvchi funksiya yozing.
- Foydalanuvchidan son olib, son juft yoki toqligini konsolga chiqaruvchi funksiya yozing.
- Foydalanuvchidan ikkita son olib, ulardan kattasini konsolga chiqaruvchi funksiya yozing. Agar sonlar teng bo'lsa, "Sonlar teng" degan xabarni chiqaring.
- Foydalanuvchidan x va n sonlarini olib, x ning n-darajasini konsolga chiqaruvchi funksiya yozing.
- Yuqoridagi funksiya n uchun 2 standart qiymatini bering.
- Foydalanuvchidan son qabul qilib, sonning 2 dan 10 gacha bo'lgan sonlarga qoldiqsiz bo'linishini tekshiruvchi funksiya yozing. Natijalarni konsolga chiqaring.



<https://cil.co/DA-func2>

8.3. QIYMAT QAYTARUVCHI FUNKSIYA

Avvalgi darsimizda yaratgan barcha funksiyalarimiz natijalarni konsolga chiqarayotgan edi. Aslida, aksar holatlarda bu g'ayrioddiy. Sababi, dasturchi sifatida biz konsolga chiqqan ma'lumotdan unumli foydalana olmaymiz. Konsoldagi qiymatni o'zgaruvchiga yuklab, undan kelajakda foydalanib ham bo'lmaydi. Funksiyadan unumli foydalanish uchun undan biror qiymatni qaytarish maqsadga muvofiq bo'ladi.

FUNKSIYADAN YAGONA QIYMAT QAYTARISH

Avvalgi bo'limdagi toliq_ism funksiyamizga o'zgartirish kiritamiz. Bu safar funksiyamiz natijani konsolga chiqarmasdan, qiymat sifatida qaytaradi. Buning uchun maxsus return operatoridan foydalanamiz.

```
def toliq_ism_yasa(ism, familiya):  
    """Toliq isma qaytaruvchi funksiya"""  
    toliq_ism = f"{ism} {familiya}"  
    return toliq_ism
```

Yuqoridagi funksiyamizga ahamiyat bersangiz, uning badanida endi print() funksiyasi yo'q. Buning o'rniiga funksiyamiz return operatori yordamida toliq_ism degan o'zgaruvchining qiymatini qaytaradi.

△ Funksiya badani to return operatoriga qadar bajariladi. return dan so'ng yozilgan kodlar funksiya badanida bo'lsa ham, bajarilmaydi.

Endi funksiyadan to'g'ri foydalanish uchun u qaytargan qiymatni biror o'zgaruvchiga yuklashimiz kerak:

```
talabal = toliq_ism_yasa('olim', 'hakimov')  
talabaz = toliq_ism_yasa('hakim', 'olimov')
```

Yuqoridagi kodlarni bajarganimizda konsolga hech narsa chiqmaydi. talabal va talabaz o'zgaruvchilarining qiymatini ko'rish uchun esa print() funksiyasidan foydalanamiz.


```
print(f"Darsga kelmagan talabalar: {talaba1} va {talaba2}")
Darsga kelmagan talabalar: Olim Hakimov va Hakim Olimov
```

Demak, qiymat qaytaradigan funksiyaning afzalligi shundaki, biz bu qiymatlardan keyin ham bemaol foydalanishimiz mumkin.

① Funksiya ichidagi o'zgaruvchilar mahalliy yoki ichki o'zgaruvchilar deyiladi (local variables). Ichki o'zgaruvchilar faqatgina funksiya ichida mavjud bo'ladi, ularga tashqaridan murojaat etib bo'lmaydi. Shuning uchun ham funksiya o'zgaruvchi emas qiymat qaytaradi.

IXTIYORIY ARGUMENTLAR

Avvalgi darsimizda funksiyalarga standart parametr berishni ko'rib chiqqan edik. Xuddi shu usul bilan ba'zi argumentlarni ixtiyoriy qilishimiz mumkin. Ya'ni funksiya ishlashi uchun bu argumentlarni kiritish majburiy emas, ixtiyoriy bo'ladi.

Keling, avvalgi funksiyamizni o'zgartiramiz va unga yana bitta otasining_ismi degan parametr qo'shamiz, lekin bu parametr ixtiyoriy bo'ladi. Buning uchun funksiya yaratishda otasining_ismi='' deb yozib ketamiz.

```
def toliq_ism_yasa(ism, familiya, otasining_ismi=''):
    """Toliq ism qaytaruvchi funksiya"""
    if otasining_ismi: # otasining_ismi mavjud bo'lsa,
        toliq_ism = f"{ism} {otasining_ismi} {familiya}"
    else:
        toliq_ism = f"{ism} {familiya}"
    return toliq_ism.title()
```

Yuqoridagi funksiyani tahlil etadigan bo'lsak, 3-qatorda biz otasining_ismi parametri bo'sh yoki yo'qligini tekshirdik. Pythonda if dan so'ng bo'sh bo'lmagan matn (string) yozsak, bu shart True qaytaradi. Demak, bu ixtiyoriy parametr kiribligani yoki yo'qligiga qarab funksiyamiz turlicha qiymat qaytaradi.

```
talaba1 = toliq_ism_yasa('olim', 'hakimov')
talaba2 = toliq_ism_yasa('hakim', 'olimov', 'abrorovich')
print(f"Darsga kelmagan talabalar: {talaba1} va {talaba2}")
```

Natija:

Darsga kelmagan talabalar: Olim Hakimov va Hakim Abrorovich Olimov

FUNKSIYADAN LUG'AT QAYTARISH

Funksiyadan soddagina qiymat emas, ro'yxat, lug'at va boshqa ma'lumot turlarini ham qaytarishimiz mumkin. Quyidagi funksiya mashina haqidagi ma'lumotlarni jamlab, ularni lug'at ko'rinishida qaytaradi:

```
def avto_info(make, model, rangi, korobka, yili, narxi=None):
    avto = {'kompaniya': make,
            'model': model,
            'rangi': rangi,
            'korobka': korobka,
            'yil': yili,
            'narxi': narxi}
    return avto
```

E'tibor bering, narxi nomli parametrga None standart qiymatini berib ketdik. None Pythonda *mavjud emas* ma'nosini beradi va if yordamida tekshirganda False mantiqiy qiymatini qaytaradi.

```
avto1 = avto_info('GM', 'Malibu', 'Qora', 'Avtomat', 2018)
avto2 = avto_info('GM', 'Gentra', 'Oq', 'Mexanika', 2016, 15000)
avto1ar = [avto1, avto2]
print('Onlayn bozordagi mavjud avtomashinalar:')
for avto in avto1ar:
    if avto['narxi']:
        narx = avto['narxi']
    else:
        narx = "Noma'lum"
    print(f"{avto['rangi']} {avto['model']}. Narxi: {narx}")
```


Dastur natijasini tahlil qilishni sizga vazifa sifatida qoldiramiz:

Onlayn bozordagi mavjud avtomashinalar:
qora Malibu. Narxi: Noma'lum
Oq Gentra. Narxi: 15000

FUNKSIYADAN RO'YXAT QAYTARAMIZ

Biz avvalroq `range()` funksiyasi bilan tanishgan edik. Bu funksiya 2 ta son qabul qilib, shu ikki son oraliqidagi sonlarni qaytaradi. Keling biz `oraliq()` degan yangi funksiya yaratamiz. `range()` dan farqli ravishda, funksiyamiz 2 son oraliqidagi sonlarni `ro'yxat` ko'rinishida qaytarasin.

```
def oraliq(min,max):  
    sonlar = [] # bo'sh ro'yxat  
    while min<max:  
        sonlar.append(min)  
        min += 1  
    return sonlar
```

Funksiyani tekshiramiz:

```
print(oraliq(0,10))  
print(oraliq(10,21))
```

Natija:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

Yuqoridagi funksiyaga uchinchi, qadam deb nomlangan ixtiyoriy parametрни qo'sha olasizmi?

```
>>> print(oraliq(0,21,2)) # 0 dan 21 gacha 2 qadam bilan  
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

FUNKSIYALARNI SIKLDA ISHLATISH

Funksiyalarni takrorlash uchun sikldan foydalanishimiz mumkin. Quyidagi misolda biz `while` yordamida avvalroq yaratgan `avto_info` funksiyamizni bir necha bor chaqiramiz va salondagi avtomashinalar ro'yxatini shakllantiramiz. Bunda ro'yxatning har bir elementi `avto_info` funksiyasidan qaytgan lug'at bo'ladi.

```
print("Saytimizdagi avtolar ro'yxatini shakllantiramiz.")  
avtolar=[] # salondagi avtolar uchun bo'sh ro'yxat  
while True:  
    print("\nQuyidagi ma'lumotlarni kiriting",end='')  
    kompaniya=input("Ishlab chiqaruvchi: ")  
    model=input("Modeli: ")  
    rangi=input("Rangi: ")  
    korobka=input("Korobka: ")  
    yili=input("Ishlab chiqarilgan yili: ")  
    narxi=input("Narxi: ")  
    #Kiritilgan ma'lumotlardan avto_info yordamida  
    #lug'at shakllantirib, ro'yxatga qo'shamiz:  
    avtolar.append(avto_info(kompaniya, model, rangi,\  
                             korobka, yili, narxi))  
    # Yana avto qo'shish-qo'shaslikni so'raymiz  
    javob = input("Yana avto qo'shasizmi? (yes/no): ")  
    if javob=='no':  
        break
```

Yuqoridagi funksiyani Pythonida bajarib ko'ring. Ro'yxatga bir necha qiymatlar qo'shing. Natijalarni konsolga chiroyli qilib chiqaring.

AMALIYOT

- Foydalanuvchidan ismi, familiyasi, tug'ilgan yili, tug'ilgan joyi, elektron manzili va telefon raqamini qabul qilib, lug'at ko'rinishida qaytaruvchi funksiya yozing. Lug'atda foyda-

- lanuvchi yoshi ham bo'lsin. Ba'zi argumentlarni kiritishni ixtiyoriy qiling (masalan, tel.raqam, el.manzil)
- Yuqoridagi funksiyani while yordamida bir necha bor chiqiring va mijozlar degan ro'yxatni shakllantiring. Ro'yxatdagi mijozlar haqidagi ma'lumotni konsolga chiqaring.
 - Uchta son qabul qilib, ulardan eng kattasini qaytaruvchi funksiya yozing.
 - Foydalanuvchidan aylananing radiusini qabul qilib olib, uning radiusini, diametrini, perimetri va yuzini lug'at ko'rinishida qaytaruvchi funksiya yozing.
 - Berilgan oralikdagi tub sonlar ro'yxatini qaytaruvchi funksiya yozing (tub sonlar – faqat birga va o'ziga qoldiqsiz bo'linuvchi, 1 dan katta musbat sonlar).
 - Foydalanuvchidan son qabul qilib, shu son miqdoricha Fibonacci ketma-ketligidagi sonlar ro'yxatini qaytaruvchi funksiya yozing.

① Har bir hadi o'zidan oldingi ikkita hadning yig'indisiga teng bo'lgan ketma-ketlik Fibonacci ketma-ketligi deyiladi. Bunda boshlang'ich had ko'pincha 1 deb olinadi. 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,...

8.4. FUNKSIYA VA RO'YXAT

Biz avvalgi darslarimizda funksiya parametr sifatida yagona qiymat berayotgan edik. Aslida, bu bilan cheklanmasdan, funksiya ro'yxat (list) ham berishimiz mumkin. Bunda funksiya ro'yxat qiymatlariga to'g'ridan to'g'ri murojaat eta oladi.

Keling, talabalarni baholaydigan funksiya yozamiz. Funksiyamiz talabalar ro'yxatini qabul qilib oladi, ro'yxatdan har bir talabani sug'urib olib, bahosini kiritishni so'raydi. Talaba ismi va bahosini lug'atga joylab, yakuniy lug'atni foydalanuvchiga qaytaradi.



<https://olico/DA-func3>

```
def bahola(ismilar):
    baholar = {}
    while ismlar:
        ism = ismlar.pop()
        baho=input(f"{ism.title()}ning bahosini kiriting: ")
        baholar[ism]=baho
    return baholar

talabalar = ['ali', 'vali', 'hasan', 'husan']
baholar = bahola(talabalar)
print(baholar)
```

Natija:

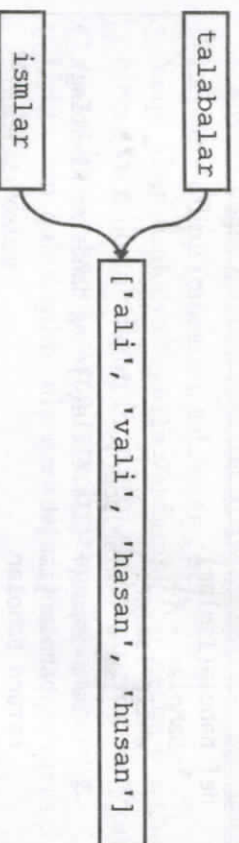
```
Husanning bahosini kiriting: 5
Hasanning bahosini kiriting: 5
Valining bahosini kiriting: 4
Alining bahosini kiriting: 3
{'husan': '5', 'hasan': '5', 'vali': '4', 'ali': '3'}
```

RO'YXATGA O'ZGARTIRISH KIRITISH

Funksiya ro'yxat uzatganimizda funksiya ro'yxat elementlariga to'g'ridan to'g'ri murojaat eta oladi. Ro'yxatga funksiya ichida kiritilgan o'zgartirishlar funksiya tashqarisidagi asl ro'yxatga ham ta'sir qiladi. Avvalgi misolimizga qaytaylik:

```
talabalar = ['ali', 'vali', 'hasan', 'husan']
baholar = bahola(talabalar)
print(talabalar)
Natija: []
```

Yuqoridagi funksiya unga uzatilgan ro'yxat ichidagi talabalarining ismini .pop() yordamida sug'urib olgani uchun bizning asl ro'yxatimiz ham bo'shab qoldi. E'tibor berib, funksiya tashqarisidagi va ichidagi ro'yxatlar ikki xil nomlangan bo'lsa-da (talabalar va ismlar), ikkalasi ham xotiradagi bitta ro'yxatga bog'langan.



18-RASM. Ikki o'zgaruvchi ham bitta ro'yxatga bog'langan

ASL RO'YXATGA O'ZGARTIRISH KIRITISHNING OLDINI OLISH

Agar funksiya asl ro'yxatga o'zgartirish kiritishini istamasangiz, funksiyaga ro'yxatning o'zini emas, uning nusxasini uzatish mumkin. Buning uchun funksiya parametrini ro'yxat_nomi[:] ko'rinishida yozish kifoya. Bunda [:] operatori ro'yxatdan nusxa olishni anglatadi:

```
talabalar = ['ali', 'vali', 'hasan', 'husan']
baholar = bahola(talabalar[:])
print(talabalar)
```

Natija:

```
['ali', 'vali', 'hasan', 'husan']
```

AMALIYOT

Matnlardan iborat ro'yxat qabul qilib, ro'yxatdagi har bir matning birinchi harfini bosh harfga o'zgartiruvchi funksiya yozing.

Yuqoridagi funksiyaning asl ro'yxatni o'zgartirmaydigan va yangi ro'yxat qaytaradigan qilib o'zgartiring.

Darsimiz davomida yozgan bahola funksiyasini .pop() metodidan foydalanmasdan va asl ro'yxatga o'zgartirish kiritmasdan faqat lug'at qaytaradigan qilib yozing.

8.5. MOSLASHUVCHAN FUNKSIYA (*args, **kwargs)

Agar funksiya bir nechta argument qabul qilishi kerak bo'lsa-yu, lekin siz argumentlar sonini aniq bilmasangiz, Pythonida istalgancha qiymat qabul qiluvchi funksiya yaratish imkoniyati bor.

*args USULI

Agar funksiya qabul qiladigan parametrlar soni noaniq bo'lsa va parametrlar yagona qiymatlar ko'rinishida uzatilsa, funksiya yaratishda argumentdan avval yulduzcha qo'yiladi (*arguments).

Quyidagi misolni ko'raylik. summa() nomli funksiyamiz istalgancha sonlarni qabul qilib oladi va ularning yig'indisini hisoblaydi:

```
def summa(*sonlar):
    """Kiritilgan sonlar yig'indisini qaytaruvchi funksiya"""
    yig'indi = 0
    for son in sonlar:
        yig'indi += son
    return yig'indi
```

Bu funksiyaning istalgancha parametr bilan chaqirish mumkin:

```
>>> print(summa(1,2))
3
>>> print(summa(1,2,3,4,5))
15
```

*args usulida barcha uzatilgan parametrlar (bir dona bo'lsa ham) funksiya ichida o'zgarmas ro'yxatga (tuple) joylanadi. Bundan kelib chiqib yuqoridagi funksiyamizni yanada soddalashtirib yozishimiz mumkin:

```
def summa(*sonlar):
    """Kiritilgan sonlar yig'indisini qaytaruvchi funksiya"""
    return sum(sonlar)
```



<https://clico/Da-func4>

Funksiyani ishlatib ko'ramiz:

```
>>> print(summa(4,5,6,7))
22
```

Agar funksiya bir nechta argument qabul qilsa, *args argumenti doim oxirida yoziladi:

```
def summa(x,y,*sonlar):
    """Kiritilgan sonlar yig'indisini qaytaruvchi funksiya"""
    return x+y+sum(sonlar)
```

Yuqoridagi funksiyamiz 2 ta majburiy parameter qabul qiladi (x va y), undan keyingi qiymatlar esa ixtiyoriy bo'ladi:

```
>>> print(summa(2))
TypeError: summa() missing 1 required positional argument: 'y'
>>> print(summa(9,10,21)
40
>>> print(summa(-10,10)
0
```

**kwargs USULI

Agar funksiya **kalit-qiymat** ko'rinishidagi argumentlarni uzatish talab qilinsa va bunday parametrlar soni noma'lum bo'lsa, argument oldidan ikkita yulduzcha qo'yiladi (**kwargs).

❶ **kwargs – keyword arguments (kalit so'zli argumentlar)

```
def avto_info(kompaniya,model,**ma'lumotlar):
    """Avto haqidagi ma'lumotlarni lug'at
    ko'rinishida qaytaruvchi funksiya"""
    ma'lumotlar['kompaniya']=kompaniya
    ma'lumotlar['model']=model
    return ma'lumotlar
```

Yuqoridagi funksiyamiz kompaniya va model degan ikki qiymatni qabul qiladi, undan keyin esa funksiyaga istalgan cha parameter uzatish mumkin. Bunday funksiya parameterlar kalit=qiymat ko'rinishida uzatiladi.

Funksiya ichida avval foydalanuvchi kiritgan **qo'shimcha** qiymatlardan iborat ma'lumotlar deb nomlangan lug'at shakl-lantiriladi. Undan keyin esa majburiy parametrlarni lug'atga qo'shamiz.

```
>>> avto1 = avto_info("GM", "malibu", rang='qora', yil=2018)
>>> avto2 = avto_info("kia", "K5", rang='qizil', narx=35000)
>>> print(avto2)
{'rang': 'oq', 'narx': 35000, 'kompaniya': 'kia', 'model': 'K5'}
```

AMALIYOT

- Istalgan cha sonlarni qabul qilib, ularning ko'paytmasini qaytaruvchi funksiya yozing.
- Talabalar haqidagi ma'lumotlarni lug'at ko'rinishida qaytaruvchi funksiya yozing. Talabaning ismi va familiyasi majburiy argument, qolgan ma'lumotlar esa ixtiyoriy ko'rinishda istalgan cha berilishi mumkin bo'lsin.

8.6. MODULLAR

Funksiyaning qulayliklaridan biri ko'p takrorlanadigan kodlarni funksiya ichida yashirishimiz va kerak bo'lganda funksiya nomi orqali murojaat etishimiz mumkinligida. Maqsadimiz dasturimizni ixcham va tushunarli qilib, kelajakda o'zimiz yoki boshqalar uchun ham "toza" kod qoldirishdir. Bu yo'nalishda yana bir qadam qo'yib, dasturimizni modullarga ajratishimiz mumkin.



<https://click/Da-func5>

MODUL NIMA?

Modul loyihamiz ichidagi alohida fayl bo'lib, dasturimiz davomida ishlatiladigan funksiyalarni (va o'zgaruvchilarni) mana shu faylga joylab, ko'zdan yashirib qo'yishimiz mumkin. Bu bizga asosiy dasturimizdan chalg'imasdan kod yozish imkoniyatini beradi.

Modul va uning ichidagi funksiyalarni istalgan payt asosiy dasturimizga yuklab olishimiz, modularni boshqa dasturchilar bilan ulashishimiz yoki kelajakda o'zimizning boshqa loyihalarimizda foydalanishimiz mumkin.

Umuman olganda, katta dasturlar bir necha o'nlab modul-lardan iborat bo'lishi tabiiy hol.

MODUL YARATAMIZ

Modul yaratish uchun asosiy dasturimizdagi funksiyalarni yangi faylga ko'chiramiz xolos. Modulga oson murojaat etishimiz uchun faylimiz asosiy dasturimiz bilan bitta papkada bo'lgani afzal. Bunda adashib ketmaslik uchun loyihamizning (dasturining) asosiy faylini main.py deb nomlash o'rinni.

Keling, biz ham avto_info_mod.py degan fayl yaratamiz va ichiga quyidagi 3 ta funksiyalarni joylaymiz:

```
def avto_info(make, model, rangi, korobka, yili, narxi=None):
    avto = {'kompaniya': kompaniya,
            'model': model,
            'rangi': rangi,
            'korobka': korobka,
            'yil': yili,
            'narx': narxi}
    return avto

def avto_kirit():
    """Foydalanuvchiga avto_info funksiyasi yordamida
    bir nechta avtolar haqida ma'lumotlarni bitta
    ro'yxatga joylash imkonini beruvchi funksiya"""
```

```
avtolar=[] # salondagi avtolar uchun bo'sh ro'yxat
while True:
    print("\nQuyidagi ma'lumotlarni kiriting", end='')
    kompaniya=input("Ishlab chiqaruvchi: ")
    model=input("Modeli: ")
    rangi=input("Rangi: ")
    korobka=input("Korobka: ")
    yili=input("Ishlab chiqarilgan yili: ")
    narxi=input("Narxi: ")
    avtolar.append(avto_info(kompaniya, model, rangi,
                             korobka, yili, narxi))
```

```
javob = input("Yana avto qo'shasizmi? (yes/no): ")
if javob=='no':
    break
return avtolar
```

```
def info_print(avto_info):
    """Avtomobillar haqida ma'lumotlar
    saqlangan lug'atni konsolga chiqaruvchi funksiya"""
    print(f"{avto_info['rang'].title()} "
          f"{avto_info['kompaniya'].upper()} "
          f"{avto_info['model'].upper()} ",
          f"{avto_info['korobka']} korobka, "
          f"{avto_info['yil']}-yil, {avto_info['narx']}$")
```

Yuqoridagi funksiyalarga asosiy dasturdan murojaat etishning bir necha usuli bor.

MODULNI CHAQIRIB OLIISH

Modul ichidagi istalgan funktsiyaga murojaat etish uchun import modul_nomi buyrug'idan foydalanamiz. Bunda modul ichidagi istalgan funktsiyaga modul_nomi.funksiya_nomi() ko'rinishida murojaat etishimiz mumkin. Ya'ni avval modul nomi, undan keyin esa nuqta qo'yilib, modul ichidagi funksiya nomi yoziladi.

Keling, yuqoridagi modulimizdagi `avto_info()` va `info_print()` funksiyalariga murojaat etamiz. Buning uchun asosiy dasturimizni (`main.py`) quyidagicha yozamiz:

```
import avto_info_mod # avto_info_mod modulini chaqiramiz
avto1 = avto_info_mod.avto_info("GM", "Malibu", "Qora", \
                                "avtomat", 2020, 40000)
avto_info_mod.info_print(avto1)
```

Natija:

```
Qora GM MALIBU, avtomat korobka, 2020-yil, 40000$
```

Ko'rib turganingizdek, `main.py` dasturimiz qisqa, tushunarli, muhim, toza bo'ldi. 3 qator kod orqida 20 qatordan ortiq kodni alohida faylda (modulda) yashirdik.

① `import modul_nomi` komandasi bir marta, dastur boshida yoziladi.

Modulga qisqa nom berish

Yuqoridagi usul bilan modulni chaqirib olishda fayl nomi uzun bo'lsa, bu o'ziga yarasha noqulayliklar tug'dirishi mumkin. Buning oldini olish uchun esa modulni chaqirganda unga as operatori yordamida qisqa nom berish va modulga qisqa nom orqali murojaat etish mumkin. Quyidagi misolda `avto_info_mod` ni qisqa qilib `aim` deb nomlab oldik va 3-4-qatorlarda modulga murojaat etishda qisqa nomidan foydalandik.

```
import avto_info_mod as aim
avto1 = aim.avto_info("GM", "Malibu", "Qora", "avto-
mat", 2020, 40000)
aim.info_print(avto1)
```

Modulga qisqa nom berganingizda bunday nomli boshqa o'zgaruvchi yoki funksiya yo'qligiga amin bo'ling. Shuningdek, dastur davomida bu nomni boshqa o'zgaruvchilarga yoki funksiyalarga berib yubormang.

MODUL ICHIDAN MA'LUM FUNKSIYALARNI CHAQIRIB OLISH

Agar asosiy dasturimizda modul ichidagi muayyan funksiyalarga murojaat etish kerak bo'lsa, quyidagi ko'rinishdagi koddan foydalanamiz: **from** modul_nomi **import** funksiya2

Bu usulning qulayligi shundaki, chaqirilgan funksiyalarga to'g'ridan to'g'ri, modul nomini yozmasdan murojaat etish mumkin. Misol uchun, avvalgi kodimizda biz faqatgina `avto_info` va `info_print` funksiyalaridan foydalandik. Shu funksiyalarni asosiy dasturimizga chaqirib olamiz:

```
from avto_info_mod import avto_info, info_print
avto1 = avto_info("GM", "Malibu", "Qora", "avtomat", 2020, 40000)
info_print(avto1)
```

Funksiyalarga qisqa nom berish

as operatori yordamida chaqirib olgan funksiyamizga ham qisqa nom berishimiz mumkin:

```
from avto_info_mod import avto_info as ainfo
from avto_info_mod import info_print as iprint
avto1 = ainfo("GM", "Malibu", "Qora", "avtomat", 2020, 40000)
iprint(avto1)
```

MODUL ICHIDAGI BARCHA FUNKSIYALARNI CHAQIRIB OLISH

Modul ichidagi barcha funksiyalarni asosiy dasturga ko'chirib olish uchun `from modul_nomi import *` komandasidan foydalanamiz.

```
from avto_info_mod import *
avto1 = avto_info("GM", "Malibu", "Qora", "avtomat", 2020, 40000)
info_print(avto1)
```


△ Bir necha sabablarga ko'ra bu usuldan foydalanish tavsiya etilmaydi. Katta modullarda yuzlab funksiyalar bo'lishi mumkin va funksiya nomi dasturimizdagi boshqa funksiya yoki o'zgaruvchi bilan bir xil nomga ega bo'lsa, dastur xato ishlashiga olib keladi.

MODULDA O'ZGARUVCHI SAQLASH

Modullarning ichida nafaqat funksiyalar, balki turli o'zgaruvchilarni ham saqlash mumkin. Modul ichidagi o'zgaruvchilarga ham xuddi yuqoridagi usullar bilan murojaat etamiz.

PYTHON MODULLARI

Python dasturlash tili tayyor modullar bilan keladi, ular-dan ba'zilari bilan tanishamiz.

math MODULI

Bu modulda matematik hisob-kitoblarni bajaruvchi funksiyalar va o'zgaruvchilar joylashgan, jumladan:

`sqrt()` – qavs ichida berilgan qiymatning kvadrat ildizini qaytaradi

```
>>> import math
>>> x=400
>>> print(math.sqrt(x))
20.0
```

`pow(x,n)` – x ning n-darajasini qaytaruvchi funksiya

```
>>> print(math.pow(5,5))
3125.0
```

`pi` – π ning qiymatini saqlovchi o'zgaruvchi

```
>>> from math import pi
>>> print(pi)
3.141592653589793
```

`log2(x)` va `log10(x)` – x ning 2 lik va 10 lik logarifmini qaytaruvchi funksiyalar.

```
>>> print(math.log2(8))
3.0
>>> print(math.log10(100))
2.0
```

`math` modulga kelgusida yana qaytamiz.

random MODULI

`random` moduli tasodifiy sonlar bilan ishlash uchun qulay funksiyalarga boy. Ulardan ayrimlari bilan tanishamiz.

`randint(a,b)` funksiyasi a va b oralig'idagi tasodifiy butun sonni qaytaradi.

```
>>> import random as r # random modulini r deb chaqiramiz
>>> son = r.randint(0,100)
>>> print(son)
77
```

`choice(x)` – berilgan x argumentning ichidan tasodifiy elementni qaytaruvchi funksiya. Bunda x bir necha elementdan iborat o'zgaruvchi (`matn`, `ro'yxat`, `to'plam` va `hokazo`) bo'lishi kerak.

```
>>> ismlar = ['olim', 'anvar', 'hasan', 'husan']
>>> ism = r.choice(ismlar) # tasodifiy ism tanlaymiz
>>> print(ism)
husan
```

Yana bir misol:

```
>>> x = list(range(0,51,5)) # ro'yxat yaratamiz
>>> print(x)
[0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
>>> print(r.choice(x)) # tasodifiy element tanlaymiz
20
```

`shuffle(x)` – `x` ichidagi elementlarni tasodifiy tartibda qaytaruvchi funksiya. Bunda `x` bir necha elementdan iborat o'zgaruvchi (matn, ro'yxat) bo'lishi kerak.

```
>>> x = list(range(11)) # ro'yxat yaratamiz
>>> print(x)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> r.shuffle(x) # ro'yxatni aralashtirib tashlaymiz
[2, 8, 9, 6, 1, 7, 10, 0, 4, 3, 5]
```

`sample(x, k)` – `x` ro'yxat ichidan tasodifiy `k` ta element ajratib olish:

```
from random import sample
x = list(range(0,100)) # 0 dan 100 gacha sonlar ro'yxati
y = sample(x,k=10) # ro'yxatdan 10 ta element ajratish
print(y)
```

Natija:

```
[81, 0, 22, 83, 87, 16, 26, 6, 8, 68]
```

`math` va `random` modullari ichidagi boshqa funksiyalar haqida Python rasmiy sahifasidan (docs.python.org) ma'lumot olishingiz mumkin.

8.7. lambda – NOMSIZ FUNKSIYA

Pythonning o'ziga xos xususiyatlaridan biri nomsiz, vaqtinchalik funksiyalar yaratish imkoniyatidir. Bunday funksiyalarni yaratishda `def` operatori o'rniga `lambda` operatori ishlatilgani uchun ham `lambda` funksiyalar deb ataladi.

Nomsiz funksiyalar quyidagicha yaratiladi:

`lambda argument:ifoda`



<https://cli.co/Da-lambda>

`Lambda` funksiyalari istalgan miqdordagi argumentlarga ega bo'lishi mumkin, ammo funksiya badanida faqat bitta ifoda mavjud bo'ladi. Ifoda bajariladi va qaytariladi (`return` operatori yozilmaydi).

Nomsiz funksiyalar biror ifodani tezda hisoblab olishda juda qulay. Misol uchun, quyidagi `lambda` funksiya ikkita argument qabul qiladi (`pi, r`) va aylana uzunligini qaytaradi:

```
import math
uzunlik = lambda pi, r : 2*pi*r
print(uzunlik(math.pi,10))
```

Natija: 62.83185307179586

Kodni tahlil qilamiz:

1-qatorda `math` modulini chaqirib oldik.

2-qatorda `lambda` funksiyani yaratdik. Funksiyamiz `pi` va `r` argumentlarini qabul qilib, `2*pi*r` qiymatni qaytaradi. Funksiyaga nom bermadik, lekin unga uzunlik identifikatori orqali murojaat etishimiz mumkin.

3-qatorda funksiyamizga murojaat etdik va natijani konsolga chiqardik.

Yana bir misol, toping-chi, quyidagi funksiyaning vazifasi nima?

```
product = lambda x, y : x ** y
print(product(3, 2))
```

Shu yerda so'rashingiz mumkin, nima uchun `lambda` nomsiz deb ataladi, axir, unga hozirgina `product` nomi bilan murojaat etdik-ku.

Gap shundaki, `lambda` funksiyalarning asl mohiyati boshqa funksiyalar bilan birga ishlaganda ko'rinadi. Keling, tushunarli bo'lishi uchun soddaroq misolni ko'rib chiqamiz.

Quyidagi dasturda biz avval daraja degan funksiya yasadik, bu funksiyamiz `n` degan o'zgaruvchi qabul qilib oladi va funksiya ichidagi noma'lum `x` ning `n`-darajasini qaytaradi.

Ya'ni daraja funksiya yasaydigan funksiya bo'ldi. Xo'sh, undan qanday foydalanamiz? 4 – 5-qatorlarga e'tibor bering, biz daraja funksiyasidan yana 2 ta funksiya yasadik: kvadrat kiritilgan sonning kvadratini hisoblaydi, kub kiritilgan sonning kubini hisoblaydi.

```
def daraja(n):
    return lambda x : x**n

kvadrat = daraja(2)
kub = daraja(3)
print(f"3-ning kvadrati {kvadrat(3)} ga teng")
print(f"3-ning kubi {kub(3)} ga teng")
```

Natija:

```
3-ning kvadrati 9 ga teng
3-ning kubi 27 ga teng
```

Lambda funksiyalaridan argument sifatida boshqa funksiyani qabul qiluvchi funksiyalar bilan ishlashda ham keng foydalaniladi. Misol uchun, `map()` va `filter()` funksiyalari.

map() FUNKSIYASI

Bu funksiya argument sifatida ro'yxat (yoki lug'at) va boshqa bir funksiyani qabul qilib, ro'yxat elementlariga qabul qilingan funksiya yordamida ishlov beradi. Tushunarli bo'lishi uchun quyidagi misolni ko'zdan kechiramiz.

```
from math import sqrt
sonlar = list(range(11)) # 0 dan 10 gacha sonlar ro'yxati
ildizlar = list(map(sqrt, sonlar))
```

Yuqoridagi misolda avval 0 dan 10 gacha sonlar ro'yxatini tuzib oldik, keyin esa `map` funksiyasiga ro'yxat va `sqrt` funksiyasini uzatib, ro'yxatdagi barcha sonlarning ildizini hisoblab oldik. `map()` funksiyasi obyekt qaytargani sababli qaytgan obyektni ro'yxatga o'tkazib olish uchun `list()` funksiyasidan foydalandik (obyektlar haqida kelgusi bobda batafsil gaplashamiz).

Yana bir misol ko'ramiz:

```
sonlar = list(range(11)) # 0 dan 10 gacha sonlar ro'yxati
def daraja2(x):
    """Berilgan sonning kvadratini qaytaruvchi funksiya"""
    return x*x
print(list(map(daraja2, sonlar)))
```

Natija:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Yuqoridagi misolda biz avval berilgan sonning kvadratini hisoblovchi funksiya yaratdik, undan keyin esa `map` yordamida sonlar ro'yxatidagi elementlarning kvadratini ham hisoblab oldik.

Endi, keeling, xuddi shu misolni lambda yordamida yozamiz:

```
kvadratlarni = list(map(lambda x:x*x, sonlar))
print(kvadratlarni)
```

Natija:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Bu safar daraja degan funksiyani yaratib o'tirmasdan, `map` funksiyasiga to'g'ridan to'g'ri darajani hisoblovchi lambda funksiya uzatdik.

`map()` funksiyasi bo'lmaganida, biz bunday dasturlarni for yordamida yozishimiz kerak bo'lar edi:

```
kvadratlarni = []
for son in sonlar:
    kvadratlarni.append(son*son)
```

`map()` funksiyasiga bir nechta ro'yxatlar ham uzatish mumkin:

```
a = [4, 5, 6]
b = [7, 8, 9]
a_plus_b = list(map(lambda x,y:x+y, a, b))
print(a_plus_b)
```

Natija:

[11, 13, 15]

map() istalgan ko'rinishdagi ma'lumot turlari bilan ishlaydi:

```
ismlar = ['hasan', 'husan', 'olim', 'umid']
print(list(map(lambda matn:matn.upper(), ismlar)))
```

Natija:

['HASAN', 'HUSAN', 'OLIM', 'UMID']

filter() FUNKSIYASI

Bu funksiya ham argument sifatida ro'yxat va boshqa funksiyani qabul qilib oladi va berilgan ro'yxat elementlarini berilgan funksiya yordamida saralaydi (filter). Bunda argument sifatida uzatilgan funksiya mantiqiy qiymat qaytarishi kerak (True yoki False).

Keling, bunga ham bir misol keltiramiz: tasodifiy sonlar ro'yxatidan juft sonlarni ajratib oluvchi dastur yozamiz. Dasturimiz 3 qismdan iborat:

1. Avvalo, random modulidagi sample() funksiyasi yordamida 0 - 99 oralig'idagi 10 ta tasodifiy sonlar ro'yxatini tuzamiz.
2. Berilgan son juft (True) yoki juft emas (False) ekanligini qaytaruvchi funksiya yozamiz.

3. filter() funksiyasiga yangi yaratilgan juftmi funksiyasi va tasodifiy sonlar ro'yxatini uzatib, yangi juft_sonlar ro'yxatini shakllantiramiz.

```
import random as r
sonlar = r.sample(range(100), 10)
def juftmi(x):
    """x juft bo'lsa, True, aks holda, False qaytarimiz"""
    return x%2==0
juft_sonlar = list(filter(juftmi, sonlar))
print(sonlar)
print(juft_sonlar)
```

Natija:

[97, 93, 67, 34, 35, 80, 70, 90, 58, 84]
[34, 80, 70, 90, 58, 84]

Keling, endi shu dasturni lambda yordamida yozamiz:

```
import random as r
sonlar = r.sample(range(100), 10)
juft_sonlar = list(filter(lambda son: son%2==0, sonlar))
print(sonlar)
print(juft_sonlar)
```

Ko'rib turganingizdek, lambda funksiya yordamida dastur birmuncha qisqaroq chiqadi. Agar juftmi funksiyasi kelajakda shart bo'lmasa, alohida funksiya yaratib o'tirmasdan, bir martalik lambda funksiyasidan foydalangan afzal.

Keling, endi filter() funksiyasi yordamida matnlarni saralashga ham misollar keltiraylik:

Quyidagi dastur mevalar ro'yxatidan b harfi bilan boshlanuvchi mevalarni ajratib oladi. Bu yerda biz matnlarga tegishli bo'lgan .startswith() metodidan foydalandik. Bu metod berilgan matn shu harfdan boshlanadimi yoki yo'qligini tekshiradi va True yoki False qiymat qaytaradi.

```
mevalar = ['olma', 'anor', 'anjir', 'o'rik', 'qovun', 'banan']
mevB = list(filter(lambda meva: meva.startswith('b'), mevalar))
print(mevB)
```

Natija:

['banan']

Quyidagi dastur esa mevalar ro'yxatidan nomi 4 yoki undan kam harfdan iborat mevalarni saralab oladi.


```
mevalar2 = list(filter(lambda meva:len(meva)<=4, mevalar))
print(mevalar2)
```

Natija:

```
['olma', 'anor']
```

Topping-chi, quyidagi kod qanday vazifani bajaradi?

```
list(filter(lambda meva:(meva.startswith('a') and \
meva.endswith('r')), mevalar))
```

AMALIYOT

- Berilgan sonni 10 ga ko'paytiruvchi lambda funksiya yozing.
- Ikki son qabul qilib, ularning yig'indisini qaytaruvchi lambda funksiya yozing.
- random moduli ichidagi sample funksiyasi yordamida 0 dan 1000 gacha sonlar oraliqidagi tasodifiy 10 ta sonlar ro'yxatini tuzing.
 - map() va lambda funksiya yordamida sonlarning kvadratini hisoblang.
 - filter() va lambda funksiya yordamida ro'yxatdan toq sonlarni ajratib oling.
- Berilgan son tub bo'lsa, True, aks holda, False qaytaruvchi funksiya yozing.
 - filter() va yuqoridagi funksiya yordamida 1 dan 10000 gacha oraliqdagi tub sonlar ro'yxatini tuzing.

9-BOB

OBJEKTGA YO'NALTIRILGAN DASTURLASH

Dasturlashni o'rganar ekanmiz, al-batta, object oriented programming (OOP), ya'ni obyektga yo'naltirilgan dasturlash tushunchasiga kelamiz. Ko'pchilik uchun bu bosqich biroz tushunarsiz va murakkabdek tuyuladi. Aslida, unday emas. Ushbu bobda object oriented programming va unga tegishli tamoyillar haqida gaplashamiz.



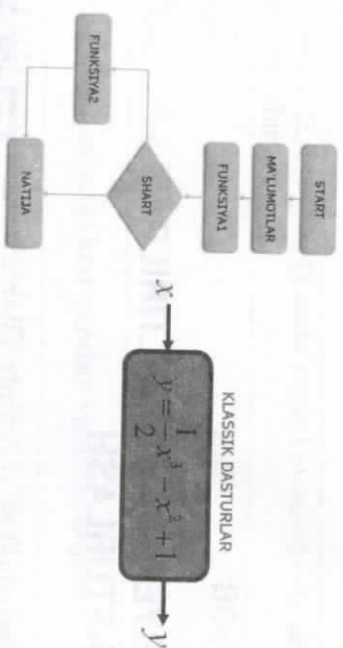
<https://elico/Da-OOP>

9.1. OOP NIMA?

Klassik yoxud chiziqli dasturlash

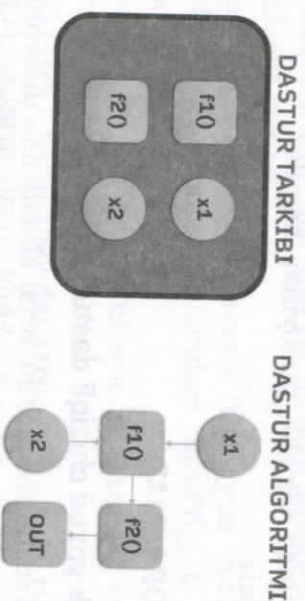
OOPni tushunish uchun avval klassik dasturlashni ko'zdan kechiraylik. Gap shundaki, ilk kompyuterlar va dasturlar matematik muammolarni hal etishga qaratilgan. Bunday dasturlar foydalanuvchidan biror ma'lumotni qabul qilib olgan va qat'iy ketma-ketlik, ya'ni tartibga amal qilgan holda turli arifmetik amallarni bajarib, dastur so'ngida foydalanuvchi kutgan natijani qaytargan. Shuning uchun ham bunday dasturlar chiziqli yoki tartibli dasturlar deb ataladi.

Siz ham dasturlashga ilk qadam qo'yganingizda mana shunday chiziqli dasturlarni yozishni o'rganishdan boshlaysiz. Sizing dasturingiz bir nechta o'zgaruvchilar va funk-



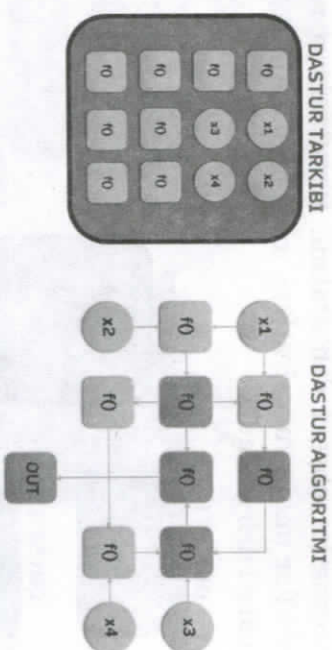
19-RASM. Chiziqli dasturlash

siyalardan iborat bo'ladi. Bu o'zgaruvchilar va funksiyalar ma'lum ketma-ketlikda bir-biri bilan munosabatga kiradi va dastur yakunida siz kutgan natijani beradi.



20-RASM. Chiziqli dastur tarkibi va algoritmi

Dastur kattalashgani sari o'zgaruvchilar va funksiyalar soni ortib boradi. Ular o'rtasidagi munosabatlar ham chi-gallashib, kodingiz murakkab va tushunishga qiyin bo'lib ketadi. Dasturlash jarayonida bitta funksiyaga o'zgartirish kiritishingiz esa unga bog'liq boshqa funksiyalarning ishdan chiqishiga va dasturingiz xato natija berishiga olib kelishi ham mumkin.



21-RASM. Chiziqli dastur murakkablashib ketishi mumkin

Chiziqli dasturlarning afzalliklari:

- Dasturlashni o'rganish uchun qulay
- Sodda va tushunarli kod
- Dastur algoritmini kuzatish oson
- Dastur xotirada kamroq joy egallaydi

Chiziqli dasturlarning kamchiliklari:

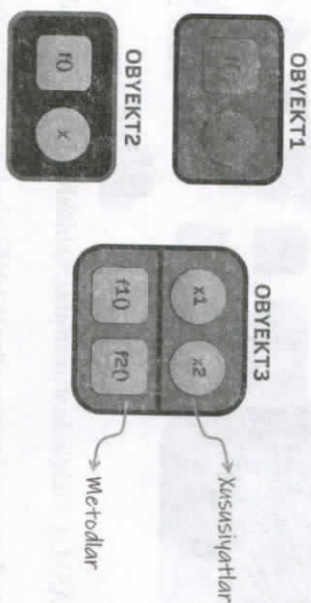
- Murakkab dasturlarni chiziqli usulda yozish qiyin (ilojsiz)
- Bir dastur uchun yozilgan koddan boshqa dasturda qayta foydalanib bo'lmaydi
- Dastur ichidagi ma'lumotlar (o'zgaruvchilar) barcha funksiyalar uchun ochiq
- ZAMONAVIY DASTURLAR CHIZIQLI EMAS

Vaqt o'tib dasturlarga qo'yilgan talablar murakkablashib borganini sababli chiziqli dasturlash tamoyili zamon talabiga javob bermay qo'ydi va 1970-yillarda OOP tamoyili olg'a suri-la boshlandi.

OBJEKT NIMA?

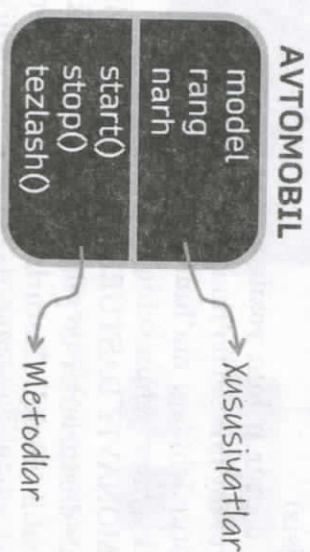
Object oriented dasturlashda o'zaro bog'liq bo'lgan o'z-garuvchilar va funksiyalar bitta konteynerga jamlanadi, bun-

day konteynerlar obyekt deb ataladi. Bir obyektga tegishli o'zgaruvchilar uning xususiyatlari, unga tegishli funksiyalar esa metodlari deb ataladi.



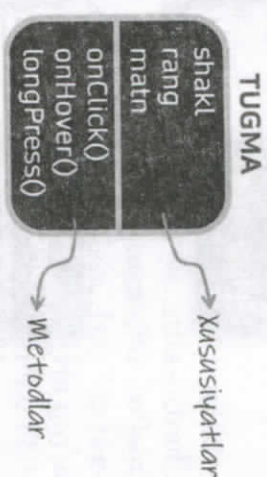
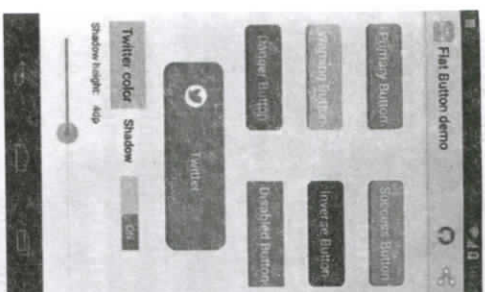
22-RASM. Obyekt

Keling, misol tariqasida avtomobil degan obyektni ko'rib chiqamiz. Avtomobilning modeli, rangi va narxi uning xususiyatlari. Avtomobilga tegishli bo'lgan start(), stop() va tezlashish() kabi amallar esa uning metodlari deyiladi.



23-RASM. AVTOMOBIL obyeksi

Agar real dasturdan misol keltiradigan bo'lsak, istalgan dastur ichidagi tugma – bu obyekt. Uning shakli, rangi va matni esa xususiyatlari bo'ladi. Tugma ustida bajariladigan amallar tugmaning metodlari deyiladi. Misol uchun, tugmani bosish, uzog bosish, ustiga sichqonchani olib borish va hokazo.



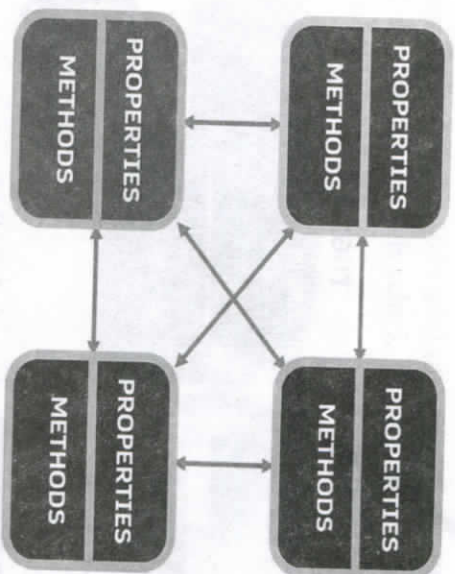
24-RASM. Dastur ichidagi tugma ham, aslida, obyekt

Object oriented dasturlar o'nlab, balki, yuzlab obyektlardan iborat bo'ladi va bunday dasturlar uchun dastur boshi yoki oxiri degan tushunchalar nisbiy.

Object oriented dasturlar bajarilishida qat'iy ketma-ketlikka amal qilmaydi. Foydalanuvchi istalgan obyektga murojaat etishi, uning ustida turli amallar bajarishi mumkin. O'z navbatida, bitta obyektga murojaat ortidan boshqa obyektlar ham faollashishi mumkin.

Misol uchun, mobil ilovalarda obyektlar dastur ichidagi tugmalar, matnlar, rasmlar va boshqa elementlardir. Foydalanuvchi istalgan tugmani bosishi, istalgan matnni ajratib olishi va boshqa amallarni istalgan tartibda bajarishi mumkin. Bunda bitta tugma (ya'ni obyekt) bosish bilan boshqa obyekt (masalan, rasm) o'zgarishi mumkin.

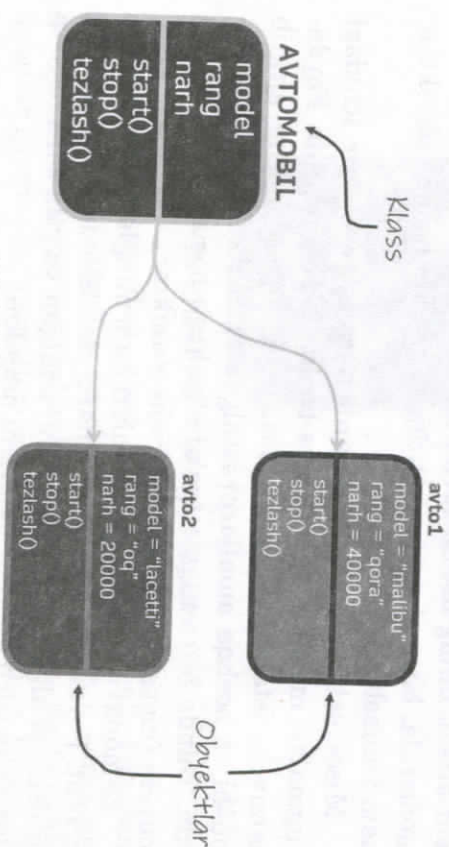
Zamonaviy kompyuter o'yinlari ham minglab obyektlardan iborat. Foydalanuvchi esa virtual o'yin olamida erkin harakat qilishi, istalgan tarafa yurishi, istalgan vaqtda turli obyektlar bilan turli amallar bajarishi mumkin.



25-RASM. Obyektlar o'rtasida munosabat

KLASS NIMA?

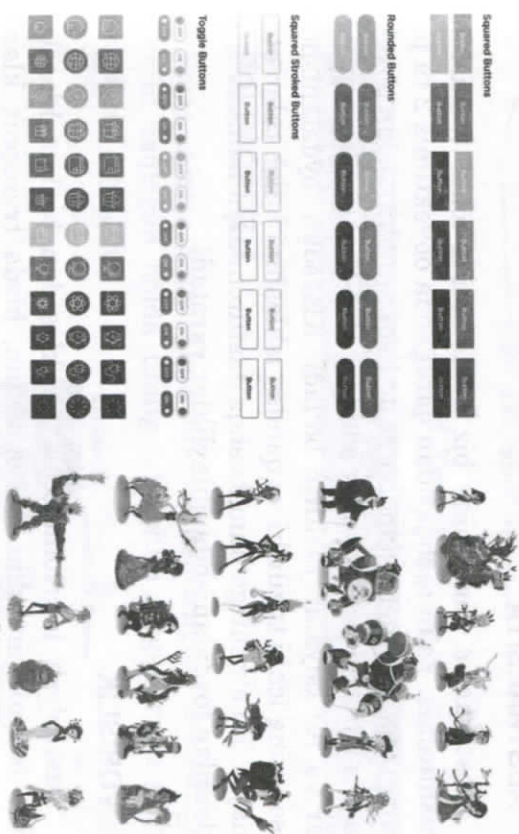
Object oriented programming haqida gaplashar ekanmiz, uning fundamental tushunchalaridan biri – klass haqida gapirib o'tmaslikning iloji yo'q. Klass obyekt yaratish uchun shablon yoki qolipdir. Bitta klassdan istalgancha nusxa oli-



26-RASM. Klass va obyektlar

shimiz va yangi obyektlar yaratishimiz mumkin. Demak, obyekt – bu biror klassning xususiy ko'rinishi. Odatda, klasslarning nomi o'zgarimas, undan yaratilgan obyektlar esa istalgancha nomlanishi mumkin.

Dasturimiz yuzlab obyektlardan iborat bo'lishi mumkin. Klasslar esa bizga obyektlarni yaratishni yengillashtiradi. Xoh dastur interfeysidagi o'nlab turli xil tugmalar bo'lsin, xoh kompyuter o'yinidagi qahramonlar bo'lsin. Har bir tugma yoki o'yin qahramoni va uning harakatlarini qayta-qayta yozmasdan, bir marta yaratilgan klassdan nusxa olib, o'nlab obyektlarni yaratishimiz mumkin.



27-RASM. Yagona klassdan yaratilgan obyektlar turli ko'rinishda bo'lishi mumkin

9.2. OOP TAMOYILLARI

INKAPSULYATSIYA

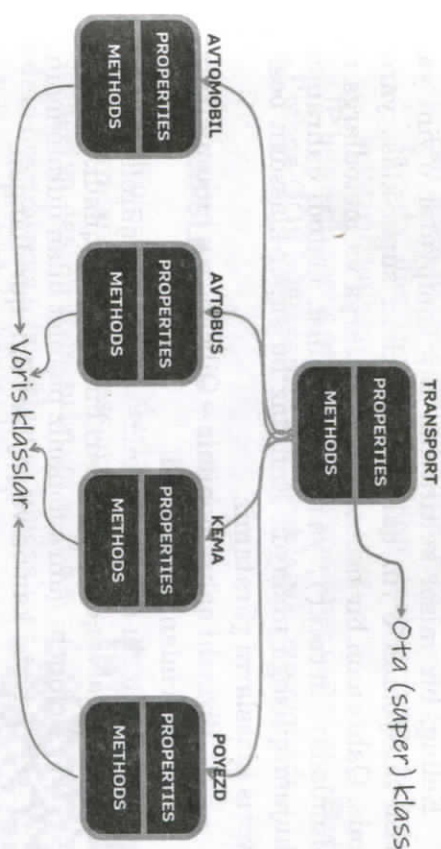
Biz object oriented dasturlash haqida gapira turib ma'lum bir obyektga tegishli bo'lgan xususiyatlar va metodlarni bit-ta konteynerga joylaymiz dedik. Bu jarayon inkapsulyatsi-ya (ya'ni kapsulaga solish) deb ataladi. Inkapsulyatsiya biz-ga klasslar yaratish va keyinchalik bu klasslardan boshqa obyektlarni yaratishga yordam beradi.

ABSTRAKSIYA

Abstraksiya yordamida biz kodimizning ichki tuzilishini yashiramiz. Ya'ni tashqaridan qaraganda obyektimiz 2 ta pa-rametr va 2 ta metoddan iborat bo'lishi mumkin, lekin obyekt to'g'ri ishlashi uchun uning ichida o'nlab boshqa o'zgaruvchi-lar va funksiyalar yashirin bo'ladi. Klassdan foydalanishda esa uning ichki tuzilishi va qanday ishlashini bilish talab etil-maydi. Bu o'zimizga ham, boshqa dasturchilarga ham mazkur klassdan foydalanishda qulayliklar yaratadi.

VORISLIK

Dasturlash jarayonida biz bir klassdan boshqa klasslar yaratishimiz mumkin. Misol uchun, bizda transport klassi bor, biz bu klassdan qo'shimcha avtomobil, avtobus, kema, poyezd kabi klasslarni yaratishimiz mumkin. Bunda bizning asl klassimiz ota yoki super-klass deb ataladi, undan yaratil-gan klasslar esa voris klasslar deyiladi.



28-RASM. Vorislik

① Voris klasslar ota klassning ba'zi yoki barcha xususiyatlari va metod-lariga ega bo'ladi.

POLIMORFIZM

Voris klass super-klassdan o'zlashtirilgan metodning nomini saqlagan holda uning ishlashini o'zgartirishi poli-morfizm deyiladi.



29-RASM. Super-klass va voris klasslar

Keling, bir misol keltiraylik. Biz kompyuter o'yini yaratish jarayonida o'yin qahramoni uchun super-klass yaratamiz. Qahramon bir nechta xususiyatlarga va metodlarga ega. Jumladan, attack(), ya'ni hujum qilish metodi qahramoni hujum qilishga undaydi. Endi biz bu super-klassdan boshqa voris klasslarni yaratamiz.

- Birinchi qahramonimiz – Qilichboz, u hujum qilganda qilichdan foydalanadi.
 - Ikkinchi qahramonimiz – Jangchi, u qurolsiz bo'lgani sababli qo'l va oyoqlari bilan hujum qiladi.
 - Uchinchi qahramonimiz pistolet bilan qurollangan.
 - Oxirgisi kamon va yoylar bilan qurollangan.
- To'rt qahramonimiz ham super-klassdan attack() metodini meros oladi, lekin bu metodni biz har bir qahramon uchun turli ko'rinishda yozishimiz va talqin etishimiz mumkin. Bu esa, o'z navbatida, bizni turli qahramonlar va turli hujum tur-lari uchun alohida metodlar yozishdan qutqaradi.
- Mana shular OOPning asosiy tamoyillari ekan.

OOP AFZALLIKLARI VA KAMCHILIKLARI

Keling, darsimiz yakunida OOPning afzalliklari va kamchiliklariga ham to'xtalib o'taylik.

Afzalliklari

- Parallel dasturlash – bir loyihaning turli qismlari bir vaqtda yaratilishi mumkin.
- Vorislik tamoyili klasslardan qayta foydalanish imkonini beradi.
- Polimorfizm tamoyili klasslarni moslashuvchan qiladi.
- Klasslardan boshqa dastur va loyihalarda qayta-qayta foydalanish mumkin.

Kamchiliklari

- Dasturlashga yangi qadam qo'yganlar uchun biroz tushunarsiz.

- Har doim ham samarali emas.
- Ba'zida dasturimizni haddan tashqari murakkablashtirib yuborishi mumkin.

OOP bilan qisqacha tanishuvimiz shundan iborat edi. Endi esa Python OOP bilan tanishuvni boshlasak ham bo'ladi.

9.3. KLASSLAR

Klasslar object oriented dasturlashning poydevorlaridan biridir. Klasslar bizga dasturlashga va dastur elementlariga real hayotdagi buyumlarga (obyektlarga) yondashgandek yondashish imkonini beradi.

Klasslar, obyektlar va ularning qanday ishlashini tushungan dasturchi mantiqiy fikrlashda ham kuchli bo'ladi. Mukammal va kompleks muammolarga ham yechimni ko'ra biladi.



<https://elicoo/Da-class>

PYTHONDA KLASSLAR

Klass tushunchasi siz uchun yangi bo'lishi mumkin, lekin biz shu vaqtgacha ulardan doimiy ravishda foydalanib keldik. Keling, x o'zgaruvchi yaratamiz, unga biror qiymat yuklaymiz va type() funksiyasi yordamida uning turini ko'ramiz:

```
>>> x = 10
>>> print(type(x))
<class 'int'>
>>> matn = "salom"
>>> print(type(matn))
<class 'str'>
```

Ko'ryapmizki, x int klassidagi, matn esa str klassidagi obyektlar ekan. Demak, biz o'zgaruvchi yaratganimizda, asli-

da, Python int yoki str klassidan foydalangan holda yangi obyektlar yaratib kelayotgan ekan.

Xuddi shu kabi, agar yangi funksiya yaratib, uning ham turini tekshirsak, funksiyamiz function klassiga tegishli obyekt bo'lib chiqadi.

```
def salom_ber():  
    print("Assalomu alaykum")  
    print(type(salom_ber))
```

Natija: <class 'function'>

Demak, Pythondagi har qanday o'zgaruvchi, funksiya va boshqa elementlar, aslida, obyektlar ekan.

METODLAR

Har bir obyekt uning ustida bajarish mumkin bo'lgan funksiyalar bilan keladi. Bu funksiyalar obyekt ichida yashirin bo'ladi, biz ularga nuqta va funksiya nomi orqali murojaat etishimiz mumkin. Bunday funksiyalar shu klassga (yoki obyektga) tegishli **metodlar** deyiladi.

Ba'zi metodlar bilan avvalgi darslarimizda tanishdik. Bir klassga tegishli metodlar boshqa klassdagi obyektlar uchun mavjud bo'lmashligi tabiiy. Masalan, matnlar uchun mavjud metodlarni butun yoki o'nlik sonlarga qo'llab bo'lmaydi.

```
>>> matn = "salom"  
>>> print(matn.upper())  
SALOM  
>>> son = 20  
>>> print(son.lower())  
AttributeError: 'int' object has no attribute 'lower'
```

KLASS YARATISH

Yangi klass yaratish uchun class operatoridan foydalanamiz va klassimizga tushunarli nom beramiz. Esingizda bo'lsin, klass hali obyekt emas, obyekt uchun shablonidir. Shuning uchun klass yaratishda mazkur klassdagi obyektlar uchun umumiy bo'lgan xususiyatlar va funksiyalarni o'ylashimiz kerak.

Keling, Talaba degan klass yaratamiz:

```
class Talaba:  
    """Talaba nomli klass yaratamiz"""  
    def __init__(self, ism, familiya, tyil):  
        """Talabaning xususiyatlari"""  
        self.ism = ism  
        self.familiya = familiya  
        self.tyil = tyil
```

Kodimizni tahlil qilamiz:

- class Talaba – Talaba nomli klass yaratdik. Klasslarga nom berishda uning birinchi harfini bosh harfdan boshlash tavsiya etiladi. Agar klass nomi 2 va undan ko'p so'zdan iborat bo'lsa, har bir so'zni bosh harf bilan boshlang.
- def __init__(self) – klassga tegishli xususiyatlarni saqlovchi maxsus metod (funksiya). self so'zi ingliz tilidan "o'zi" deb tarjima qilinadi va bu klassdan yaratilgan obyektning o'ziga ishora qiladi. Ya'ni keyinchalik biz obyekt ichidagi metodga murojaat etganimizda shu obyektning o'zi birinchi bo'lib funksiyaga argument sifatida uzatiladi, obyekt ustida turli amallar bajarish imkonini beradi.
- def __init__(self, ism, familiya, tyil) – yaratayotgan klassimizga xos xususiyatlarni def __init__(self) funksiyasiga argument sifatida uzatamiz. Bizning Talaba klassimiz ism, familiya va tug'ilgan yilga (tyil) ega bo'ladi.
- Keyingi qatorlarda esa self.xususiyat = argument komandasi yordamida uzatilgan argumentlarni klassning xususiyatlariga qo'llaymiz.

susiyatlari bilan bog'layapmiz. Bu yerda xususiyat nomi uzatilgan argument nomi bilan mos tushishi shart emas, xususiyatga istalgan nom berishimiz mumkin (masalan, self.name = ism)

△ def __init__ metodi yozilishida init so'zidan avval va keyin ikki pastki chiziq yoziladi.

KLASSDAN OBYEKT YARATISH

Klassimiz tayyor, keing, endi klassimizdan yangi obyekt yaratamiz.

```
talaba1 = Talaba("Alijon", "Valiyev", 2000)
```

Mana, talaba1 obyektimiz tayyor. Obyektni yaratish uchun Talaba klassiga murojaat etdik va talabaning ismi, familiyasi va tug'ilgan yilini parametr sifatida uzatdik.

Obyektning xususiyatlarini ko'rish

Obyektning xususiyatlarini ko'rish uchun nuqta orqali murojaat etishimiz mumkin.

```
>>> print(talaba1.ism)
Alijon
>>> print(talaba1.familiya)
Valiyev
```

Klassdan bir nechta obyektlar yaratish

Yuqoridagi klassdan istalganicha obyektlar yaratishimiz mumkin:

```
talaba2 = Talaba("Olim", "Olimov", 1995)
talaba3 = Talaba("Husan", "Akbarov", 2004)
talaba4 = Talaba("Hasan", "Akbarov", 2004)
```

Bunda har bir obyekt o'zining alohida xususiyatlariga ega bo'ladi.

```
>>> print(talaba2.ism)
Olim
>>> print(talaba4.familiya)
Akbarov
```

KLASSGA METODLAR QO'SHAMIZ

Obyektimizning xususiyatlarini aniqlab oldik, keing, endi obyekt bajarishi kerak bo'lgan metodlarni ham qo'shaylik.

```
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self, ism, familiya, tyil):
        """Talabaning xususiyatlari"""
        self.ism = ism
        self.familiya = familiya
        self.tyil = tyil

    def tanishtir(self):
        print(f"Ismim {self.ism} {self.familiya}."
              f"{self.tyil} yilda tug'ilganman")
```

Boshlanishiga klassimizga bitta, tanishtir metodini qo'shildik. Bu metodimiz, ko'rib turganingizdek, bitta self (ya'ni obyektning o'zini) argumentini qabul qiladi va talaba haqidagi ma'lumotlarni konsolga chiqaradi.

Obyektning metodlariga murojaat etamiz

Obyekt ichidagi funksiyaga, ya'ni obyektning metodiga murojaat etamiz:

```
talaba4 = Talaba("Hasan", "Akbarov", 2004)
talaba4.tanishtir()
```

Natija: Ismim Hasan Akbarov. 2004-yilda tug'ilganman

Klassimiz istalgancha metodlarga ega bo'lishi mumkin. Yuqoridagi klassga yana obyektlar qo'shamiz:

Kitobdagi kodlar cho'zilib ketmasligi uchun kodning qaytarilgan qismini uch yulduzcha *** bilan bildirib ketamiz.

```
class Talaba:
    """Talaba nomli klass yaratamiz"""
    ***
    def get_name(self):
        """Talabaning ismini qaytaradi"""
        return self.ism

    def get_lastname(self):
        """Talabaning familiyasini qaytaradi"""
        return self.familiya

    def get_fullname(self):
        """Talabaning ism-familiyasini qaytaradi"""
        return f"{self.ism} {self.familiya}"
```

Metodlarni tekshirib ko'ramiz:

```
>>> talabal = Talaba("Alijon", "Valiyev", 2000)
>>> print(talabal.get_fullname())
Alijon Valiyev
>>> print(talabal.get_lastname())
Valiyev
```

Argument qabul qiluvchi metodlar

Avvalgi misolimizda barcha metodlar faqatgina self parametrini qabul qilyapti. Aslida, boshqa funksiyalar kabi klass ichidagi metodlar ham argumentlar qabul qilishi mumkin. Yuqoridagi Talaba klassimizga yangi get_age() metodini qo'shamiz:

```
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self, ism, familiya, tyil):
        """Talabaning xususiyatlari"""
        self.ism = ism
        self.familiya = familiya
        self.tyil = tyil

    ***

    def get_age(self, yil):
        """Talabaning yoshini qaytaradi"""
        return yil - self.tyil
```

get_age() metodi obyektning o'zidan tashqari qo'shimcha yil argumentini ham qabul qiladi va talabaning yoshini qaytaradi.

```
>>> talabal = Talaba("Alijon", "Valiyev", 2000)
>>> print(talabal.get_age(2021))
21
```

pass OPERATORI

Pythonda hech qanday vazifani bajarmaydigan pass operatori mavjud. Bu operator dan bo'sh metodlar yaratishda foydalanish mumkin. Misol uchun, siz klassingiz uchun muhim metodlarni bilasiz, lekin metod badani hali tayyor emas. Agar metod badanini bo'sh qoldirsangiz, Python IndentationError xatosini qaytaradi. Shunday holatlarda funktsiya badaniga pass operatorini qo'yib ketishimiz mumkin:

```
class User:
    def __init__(self, name, username, email):
        self.name = name
        self.username = username
        self.mail = email
```

```
def describe():
    pass

def get_email():
    pass
```

Yuqoridagi klassimizda describe() va get_email() funksiyalar badani hali tayyor emas, bo'shliqni to'ldirish uchun esa pass operatoridan foydalanganmiz.

① pass operatori bilan if-else, for, while bloklarini ham vaqtinchalik to'ldirib turish mumkin

AMALIYOT

- Veb-sahifangiz uchun foydalanuvchi (user) klassini tuzing. Klassning xususiyatlari sifatida, odatda, ijtimoiy tarmoqlar talab qiladigan ma'lumotlarni kiriting (ism, foydalanuvchi ismi, email va hokazo).
- Klassga bir nechta metodlar qo'shing: jumladan, get_info() metodi foydalanuvchi haqida yig'ilgan ma'lumotlarni chiroyli qilib chiqarsin (masalan: Foydalanuvchi: alijon1994, ismi: Alijon Valiyev, email: alijon1994@gmail.com).
- Klassdan bir nechta obyektlar yaratib, uning xususiyatlari va metodlariga murojaat eting.

9.4. OBYEKTLAR BILAN ISHLASH

Xususiyatlarga standart qiymat berish

Avvalgi bo'limda biz klass yaratish, unga xususiyatlar va metodlar qo'shishni ko'rib chiqdik. Klassdan obyekt yaratganimizda esa uning xususiyatlarini parametr sifatida uzatishni o'rgandik.

Pythonda obyektning ba'zi xususiyatlarini parametr yordamida uzatmasdan, klass yaratishda unga standart qiymat

berib ketishimiz mumkin. Keling, Talaba klassimizga qaytamiz. Bu klassimiz 3 ta xususiyatga ega edi: ism, familiya, tyil. Biz yana bitta qo'shimcha bosqich nomli xususiyat qo'shamiz, va unga standart qiymat sifatida 1 beramiz, e'tibor qiling, bu xususiyat obyekt yaratilishida parametr sifatida uzatilmaydi:



<https://elico/Da-object>

```
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self, ism, familiya, tyil):
        """Talabaning xususiyatlari"""
        self.ism = ism
        self.familiya = familiya
        self.tyil = tyil
        self.bosqich = 1

    def get_info(self):
        return f"{self.ism} {self.familiya}. {self.bosqich}-bosqich talabasi."
```

Endi Talaba klassidan yangi obyekt yaratganimizda har bir yangi talabaning kursi 1 ga teng bo'ladi.

```
>>> talabal = Talaba("Alijon", "Valiyev", 2000)
>>> print(talabal.get_info())
Alijon Valiyev. 1-bosqich talabasi
```

STANDART QIYMATNI O'ZGARTIRISH

Obyektning standart qiymatiga ham boshqa xususiyatlar kabi nuqta orqali murojaat etishimiz va uning qiymatini almashtirishimiz mumkin:

```
>>> talabal.bosqich = 2
>>> print(talabal.bosqich)
2
```


Yana boshqa usuli – obyekt xususiyatini yangilovchi metod yozish.

```
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self,ism,familiya,tyil):
        """Talabaning xususiyatlari"""
        ***
    def get_info(self):
        ***
    def set_bosqich(self,bosqich):
        """Talabaning kursini yangilovchi metod"""
        self.bosqich = bosqich
```

Metodga murojaat etamiz:

```
>>> talabal.set_bosqich(3)
>>> print(talabal.get_info())
Alijon Valiyev. 3-bosqich talabasi
```

Umuman olganda, xususiyatlarni yangilashda turli usullardan foydalanish mumkin. Misol uchun, talabaning bosqichi, odatda, 1 tadan ko'payib boradi, shuning uchun quyidagicha metod ham yozishimiz mumkin:

```
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self,ism,familiya,tyil):
        ***
    def get_info(self):
        ***
    def set_bosqich(self,bosqich):
        ***
    def update_bosqich(self):
        """Talabaning bosqichini 1 taga ko'paytirish"""
        self.bosqich += 1
```

update_bosqich() metodiga murojaat etganimizda talabaning bosqichi oshib boradi:

```
>> talabal = Talaba("Alijon", "Valiyev", 2000)
>> print(talabal.get_info())
Alijon Valiyev. 1-bosqich talabasi
>>> talabal.update_bosqich() # 1 bosqichga oshiramiz
>>> print(talabal.get_info())
Alijon Valiyev. 2-bosqich talabasi
>>> talabal.update_bosqich() # 1 bosqichga oshiramiz
>>> print(talabal.get_info())
Alijon Valiyev. 3-bosqich talabasi
```

OBJEKTLAR O'RTASIDA MUNOSABAT

Obyektga yo'naltirilgan dasturlashning atfzalligi turli obyektlar o'rtasida o'zaro munosabatlarni oson yo'lga qo'yish mumkinligidadir. Keling, yangi Fan degan klass yaratamiz va fanimizga talabalar qo'shish uchun add_student() metodini yozamiz:

```
class Fan():
    def __init__(self,nomi):
        self.nomi = nomi
        self.talabalar_soni = 0
        self.talabalar = []
    def add_student(self,talaba):
        """Fanga talabalar qo'shish"""
        self.talabalar.append(talaba)
        self.talabalar_soni += 1
```

E'tibor bering, Fan klassi faqatgina yagona nomi degan parameterga ega. Qolgan xususiyatlariga esa standart qiymat berilgan: talabalar_soni=0, talabalar ro'yxati bo'sh.

Fanimizga talaba qo'shish uchun add_student() metodini chaqiramiz. Bu metod parameter sifatida Talaba klassiga oid

obyektni qabul qiladi va uni talabalar ro'yxatiga qo'shadi. Shuningdek, bu metod yangi talaba qo'shilganda talabalar_soni ni bittaga oshirib qo'yadi.

Keling, boshlanishiga yangi fan obyektini va bir nechta talabalarni yaratamiz:

```
matematika = Fan("Oliy Matematika")
talaba1 = Talaba("Alijon", "Valiyev", 2000)
talaba2 = Talaba("Hasan", "Alimov", 2001)
talaba3 = Talaba("Akrom", "Boriyev", 2001)
```

Talabalarni yangi fanimizga qo'shamiz:

```
matematika.add_student(talaba1)
matematika.add_student(talaba2)
matematika.add_student(talaba3)
```

Mana, fanimiz tayyor, talabalar qo'shildi. Keling, endi fan haqida ma'lumotlarni olamiz:

```
>>> print(matematika.talabalar_soni)
3
```

Fanimizga 3 ta talaba qo'shilibdi. Talabalar haqida ma'lumot olsak bo'ladimi?

```
>>> print(matematika.talabalar)
[<__main__.Talaba object at 0x000001AB08939588>, <__main__.Talaba object at 0x000001AB089397C8>, <__main__.Talaba object at 0x000001AB087D9F08>]
```

Talabalarning ism-familiyasi o'rniqa qandaydir tushunarsiz ma'lumotlar. Aslida, hammasi to'g'ri, yuqorida fanimizga yangi talabalarni obyekt sifatida qo'shgan edik, yuqoridagi natija esa matematika.talabalar ro'yxatida Talaba klassiga oid 3 ta obyekt borligini ko'rsatmoqda.

Fanimizga yozilgan talabalar haqida tushunarli ma'lumot olish uchun Fan klassiga yangi get_students() metodini qo'shamiz. Bu metod talabalar ichidagi har bir talaba obyektiga murojaat etib, get_info() metodi yordamida talabaning ma'lumotlarini oladi, ro'yxatga qo'shadi va shu ro'yxatni qaytaradi:

```
class Fan():
    ***
    def get_students(self):
        return [talaba.get_info() for talaba in self.talabalar]
```

Shu o'rinda Pythonda ro'yxatlar bilan ishlashning qulayliklaridan birini ham ko'rsatib o'tsak. get_students() metodiga e'tibor bersangiz, biz yangi ro'yxat shakllantirishda 1 qator koddan foydalandik:

```
[talaba.get_info() for talaba in self.talabalar]
```

Kodimizni tahlil qilsak, self.talabalar ichidagi har bir talaba uchun get_info() metodini bajar degan ma'no kelib chiqadi. Kodni kvadrat qavslar ichiga olganimiz uchun har bir sikl natijasi avtomatik ravishda ro'yxatga qo'shib boriladi.

Mana, endi metodga murojaat etib, talabalar haqida ma'lumot olishimiz mumkin:

```
>> mat_talabalar = matematika.get_students()
>>> print(mat_talabalar)
>>> ['Alijon Valiyev. 1-bosqich talabasi ',
      'Hasan Alimov. 1-bosqich talabasi ',
      'Akrom Boriyev. 1-bosqich talabasi ']
```

Shunday qilib, ikki bir-biriga bog'liq bo'lmagan obyektlar ustida turli munosabatlar o'rnatishimiz mumkin.

NUQTA YOKI METOD?

Pythondagi obyektlarning o'ziga xos xususiyatlaridan biri – obyektning xususiyatiga nuqta orqali to'g'ridan to'g'ri murojaat etish mumkin. Misol uchun, avval yaratilgan talaba obyektning ismini bilish uchun talaba1.ism deb yozish kifoya.

Bu ancha qulay bo'lsa-da, mazkur usuldan foydalanish tavsiya etilmaydi. Sababi, vaqt o'tib klassingiz takomillashishi, uning ba'zi xususiyatlari o'zgarishi, o'chirilishi yoki almasishi mumkin. Shunday holatlarda nuqta orqali murojaat etish siz kutgan natijani bermasligi va dastur xato ishlashiga olib kelishi tabiiy. Bunday holatlarning oldini olish uchun esa obyektning xususiyatlariga murojaat etish yoki ularni o'zgartirishni metod orqali yo'lga qo'yishni odat qilish kerak.

① Umuman olganda, obyektning xususiyatlariga to'g'ridan to'g'ri murojaat etishni taqiqlash uchun inkapsulyatsiyadan foydalanamiz. Bu haqida kelgusi bo'limda batafsil to'xtalamiz.

Dasturchilar orasida obyektning xususiyatlarini o'zgartiradigan metodlarni set (o'zgartirish) so'zi bilan, xususiyatlarni qaytaradigan metodlarni esa get (olish) so'zi bilan boshlash goida qilib olingan. Masalan: set_name() va get_name() kabi.

Agar yuqoridagi goidalarga amal qilgan holda Talaba klassimizni yangilaydigan bo'lsak, u shunday ko'rinishga ega bo'ladi:

```
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self,ism,familiya,tyil):
        """Talabaning xususiyatlari"""
        self.ism = ism
        self.familiya = familiya
        self.tyil = tyil
        self.bosqich = 1
```

```
def set_bosqich(self,bosqich):
    """Talabaning kursini yangilovchi metod"""
    self.bosqich = bosqich

def update_bosqich(self):
    """Talabaning bosqichini 1 taga ko'paytirish"""
    self.bosqich += 1

def get_info(self):
    """Talaba haqida ma'lumot"""
    return f"{self.ism} {self.familiya}. {self.bosqich}-bosqich talabasi "

def get_name(self):
    """Talabaning ismini qaytaradi"""
    return self.ism

def get_lastname(self):
    """Talabaning familiyasini qaytaradi"""
    return self.familiya

def get_fullname(self):
    """Talabaning ism-familiyasini qaytaradi"""
    return f"{self.ism} {self.familiya}"

def get_age(self,yil):
    """Talabaning yoshini qaytaradi"""
    return yil-self.tyil
```

Xuddi shu kabi, Fan klassimizning yakuniy ko'rinishi quyidagicha bo'ladi:

```
class Fan():
    """Fan nomli klass"""
    def __init__(self,nomi):
        self.nomi = nomi
        self.talabalar_soni = 0
        self.talabalar = []
```

```

def add_student(self, talaba):
    """Fanga talabalar qo'shish"""
    self.talabalar.append(talaba)
    self.talabalar_soni += 1

def get_name(self):
    """Fan nomi"""
    return self.nomi

def get_students(self):
    """Fanga yozilgan talabalar haqida ma'lumot"""
    return [x.get_info() for x in self.talabalar]

def get_students_num(self):
    """Fanga yozilgan talabalar soni"""
    return self.talabalar_soni

```

OBYEKTNING XUSUSIYATLARI VA METODLARINI KO'RISH

Obyektlar bilan ishlaganda, o'z-o'zidan, shu obyektga tegishli xususiyatlar va metodlarni bilish talab qilinadi. Agar obyekt tegishli bo'lgan klassni o'zimiz yaratgan bo'lsak, istalgan payt klass ichini ko'rib olishimiz mumkin. Lekin klass juda ham uzun bo'lsa yoki boshqalar yaratgan klass haqida ma'lumot olish talab qilinsa, buning bir nechta yo'li bor.

dir() FUNKSIYASI

dir() funksiyasi yordamida istalgan obyekt yoki klassning metodlarini ko'rib olishimiz mumkin:

```

>>> dir(Talaba)
['_class_',
 '_delattr_',
 '_dict_',
 '_dir_',

```

```

'_doc_',
'_eq_',
'_format_',
'_ge_',
'_getattribute_',
'_gt_',
'_hash_',
'_init_',
'_init_subclass_',
'_le_',
'_lt_',
'_module_',
'_ne_',
'_new_',
'_reduce_',
'_reduce_ex_',
'_repr_',
'_setattr_',
'_sizeof_',
'_str_',
'_subclasshook_',
'_weakref_',
'_get_age',
'_get_fullname',
'_get_info',
'_get_lastname',
'_get_name',
'_set_bosqich',
'_update_bosqich']

```

Lekin bunda har bir klass bilan keluvchi maxsus **dunder metodlar** ham chiqib keldi. Dunder metodlar ikki pastki chiziq (__) bilan boshlanadi va maxsus holatlar uchun saqlab qo'yilgan. Biz hozircha faqat `__init__` metodi bilan tanishdik, qolganlari bilan keyingi darslarimizda yana ko'rishamiz. Dunder metodlardan keyin esa biz murojaat etishimiz mumkin bo'lgan metodlar ro'yxati kelgan.

Keling, dunder metodlarni tashlab, bizga kerak metodlarni qaytaruvchi soddala funksiya yozamiz:

```
def see_methods(klass):
    return [method for method in dir(klass) if \
            method.startswith('__') is False]
```

Funksiya yordamida Talaba klassining metodlarini ko'ramiz:

```
>>> print(see_methods(Talaba))
['get_age', 'get_fullname', 'get_info', 'get_lastname', 'get_name', 'set_bosqich', 'update_bosqich']
```

Agar dir() funksiyasiga klass emas, obyekt uzatsak, metodlardan tashqari xususiyatlar ham chiqib keladi:

```
>>> print(see_methods(talabal))
['bosqich', 'familiya', 'get_age', 'get_fullname', 'get_info', 'get_lastname', 'get_name', 'ism', 'set_bosqich', 'tyil', 'update_bosqich']
```

__dict__ METODI

Yuqorida zikr qilingan dunder metodlardan biri __dict__ metodi bo'lib, bu metod obyektning xususiyatlarini lug'at ko'rinishida qaytaradi:

```
>>> print(talabal.__dict__)
{'ism': 'Alijon', 'familiya': 'Valiyev', 'tyil': 2000, 'bosqich': 1}
```

Natijadan faqatgina kalitlarni ajratib olsak, obyektning xususiyatlari chiqadi:

```
>>> print(talabal.__dict__.keys())
dict_keys(['ism', 'familiya', 'tyil', 'bosqich'])
```

AMALIYOT

- Avto degan yangi klass yarating. Unga avtomobillarga doir bo'lgan bir nechta xususiyatlar (model, rang, korobka, narx va hokazo) qo'shing. Ayrinim xususiyatlarga standart qiymat bering (masalan, kilometer=0).
- Avto ga oid obyektning xususiyatlarini qaytaradigan metodlar yozing.
- get_info() metodi avto haqida to'liq ma'lumotni matn ko'rinishida qaytarsin.
- Avto ga oid obyektning xususiyatlarini yangilaydigan metodlar yozing.
- update_km() metodi son qabul qilib olib, avtomobilning yurgan kilometrini yangilab borsin.
- Yangi Avtosalon degan klass yarating va kerakli xususiyatlar bilan to'ldiring (salon nomi, manzili, sotuvdagi avtomobillar va hokazo)
- Avtosalonga yangi avtomobillar qo'shish uchun metod yozing.
- Avtosalonidagi avtomobillar haqida ma'lumot qaytaruvchi metod yozing.
- Yuqoridagi obyektlar va ularga tegishli metodlarni tekshirib ko'ring.
- dir() funksiyasi va __dict__ metodi yordamida o'zingiz yozgan obyektlarning xususiyatlari va metodlarini toping.
- Yuqoridagi mashqni Pythonga tegishli funksiyalarga qo'llab ko'ring: dir(str), dir(int), dir(print) va hokazo.

9.5. VORISLIK VA POLIMORFIZM

Obyektga yo'natirilgan dasturlashning qulayliklaridan biri klasslardan boshqa klass yaratish imkoniyatidir. Bizga kerak bo'lgan yangi klassning avval yaratilgan boshqa klass bilan o'xshashlik joylari bo'lsa, bu klassdan voris klass yaratishimiz mumkin. Bunda asl klass ota yoki super-klass deb ataladi. Super-klassdan yaratilgan voris klass otaning barcha yoki tanlangan xususiyatlari va metodlarini meros olish bilan birga o'ziga xos xususiyat va metodlariga ega bo'ladi.

Keling, Shaxs klassini yaratamiz, bu klassimiz keyinchalik boshqa klasslar uchun super-klass vazifasini bajaradi:



<https://dli.co/Da-voris>

```
class Shaxs:
    """Shaxslar haqida ma'lumot"""
    def __init__(self, ism, familiya, passport, tyil):
        self.ism = ism
        self.familiya = familiya
        self.passport = passport
        self.tyil = tyil

    def get_info(self):
        """Shaxs haqida ma'lumot"""
        info = f"{self.ism} {self.familiya}. "
        info += f"Passport:{self.passport}, {self.tyil}-yilda tug'ilgan"
        return info

    def get_age(self, yil):
        """Shaxsning yoshini qaytaruvchi metod"""
        return yil - self.tyil
```

Klassimizni tekshirib ko'ramiz:

```
>>> inson = Shaxs("Hasan", "Alimov", "FB001122", 1995)
>>> print(f"{inson.get_info()}. {inson.get_age(2021)} yoshda.")
Hasan Alimov. Passport:FB001122, 1995-yilda tug'ilgan. 26 yoshda.
```

VORIS KLASSTI YARATISH

Endi avvalgi darsimizda yaratgan Talaba klassimizni qaytadan yaratamiz. Bu safar, avvalgisidan farqli ravishda, Talaba klassi yaratishda Shaxs super-klassidan foydalanamiz:

```
class Talaba(Shaxs):
    """Talaba klassi"""
    def __init__(self, ism, familiya, passport, tyil):
        """Talabaning xususiyatlari"""
        super().__init__(ism, familiya, passport, tyil)
```

Kodimizni tahlil qilaylik:

- 1-qatorda klass nomidan so'ng qavs ichida super-klass nomini berdik
- def __init__(self, ism, familiya, passport, tyil): xususiyatlarini meros olishini ko'rsatdik: super().__init__()

Yangi yaratgan Talaba klassimiz Shaxs klassining barcha xususiyatlari va metodlariga ega bo'ladi.

```
>>> talaba = Talaba("Valijon", "Aliyev", "FA112299", 2000)
>>> print(talaba.get_info())
Valijon Aliyev. Passport:FA112299, 2000-yilda tug'ilgan
```

Talaba klassi uchun alohida get_info() metodini yozmagan bo'lsak-da, bu metod Talabaga Shaxsdan meros o'tdi. Xuddi shu kabi, get_age() metodiga ham murojaat etishimiz mumkin:

```
>>> print(talaba.get_age(2021))
21
```


① Dastur davomida super-klass voris klasslardan avval yozilgan (chaqirilgan) bo'lishi kerak.

VORIS KLASSGA XOS XUSUSIYATLAR VA METODLAR

Hozirgi ko'rinishda Talaba va Shaxs klasslari o'rtasida hech qanday farq yo'q. Keling, Talaba klassimizga o'ziga xos xususiyatlar va metodlar qo'shaylik. Dastavval talabaning bosqichi va ID raqamini xususiyat sifatida qo'shamiz. Bunda ID raqami obyekt yaratilishida parametr sifatida uzatiladi, bosqich esa standart qiymatga ega.

```
class Talaba(Shaxs):
    """Talaba klassi"""
    def __init__(self, ism, familiya, passport, tyil, id):
        """Talabaning xususiyatlari"""
        super().__init__(ism, familiya, passport, tyil)
        self.idraqam = id
        self.bosqich = 1
```

Endi Talaba obyektini yaratishda qo'shimcha ID parametrini ham kiritish talab qilinadi:

```
talaba=Talaba("Valijon", "Aliyev", "FA112299", 2000, "0000012")
```

Navbat bu qiymatlarni qaytaruvchi metodlarga:

```
class Talaba(Shaxs):
    """Talaba klassi"""
    ***
    def get_id(self):
        """Talabaning ID raqami"""
        return self.idraqam
    def get_bosqich(self):
        """Talabaning o'qish bosqichi"""
        return self.bosqich
```

Metodlarni tekshirib ko'ramiz:

```
>>> print(f"{talaba.get_info()}. ID:{talaba.get_id()}")
Valijon Aliyev. Passport:FA112299, 2000-yilda tug'ilgan.
ID:0000012
>>> print(f"{talaba.get_bosqich()}-bosqich talabasi")
1-bosqich talabasi
```

Shu paytda yangi klassimizga istalgancha yangi xususiyatlar va metodlar qo'shishimiz mumkin. Agar yangi xususiyat yoki metod super-klassga ham aloqador bo'lsa, uni birdan super-klassga qo'shish tavsiya qilinadi.

① Voris klass boshqa klass uchun super-klass bo'lishi mumkin.

POLIMORFIZM – SUPER-KLASS METODLARINI QAYTA YOZISH

Voris klassga super-klassdan meros qolgan istalgan metodni qayta talqin etish mumkin. Avvalgi misolimizdagi get_info() super metodini ko'zdan kechiraylik, bu metod talabaning ismi, familiyasi, passport raqami va tug'ilgan yilini qaytaradi:

```
>>> print(talaba.get_info())
Valijon Aliyev. Passport:FA112299, 2000-yilda tug'ilgan
```

Keling, get_info() metodi talabaga mos ma'lumotlar qaytarishi uchun Talaba klassi ichida xuddi shu nomli metodni qayta yozamiz:

```
class Talaba(Shaxs):
    """Talaba klassi"""
    ***
    def get_info(self):
        """Talaba haqida ma'lumot"""
        info = f"{self.ism} {self.familiya}. "
        info += f"{self.get_bosqich()}-bosqich. ID: {self.idraqam}"
        return info
```

Metodni tekshirib ko'ramiz:

```
>>> print(talaba.get_info())
Valijon Aliyev. 1-bosqich. ID: 0000012
```

OBJEKT ICHIDA OBJEKT

Ba'zida klassimiz xususiyatlar va ular bilan ishlaydigan metodlarga to'lib ketishi, bu esa, o'z navbatida, obyektga muhojira etishni qiyinlashtirishi mumkin. Shunday holatlarda ba'zi xususiyatlarni alohida klass ko'rinishida yozish va keyinchalik bu klassdan yaratilgan obyektidan boshqa obyektning xususiyati sifatida foydalanish mumkin.

Misol uchun, yuqoridagi Shaxs klassimizga yana bir manzil degan xususiyat qo'shaylik. Odatda, manzil bir nechta qismlardan iborat bo'ladi (xonadon, ko'cha, mahalla, tuman/shahar, viloyat, indeks va hokazo) va ularning har biri uchun Shaxs ichida alohida xususiyat yaratmasdan, alohida Manzil degan klassga yuklash maqsadga muvofiq.

```
class Manzil:
    """Manzil saqlash uchun klass"""
    def __init__(self, uy, kocha, tuman, viloyat):
        """Manzil xususiyatlari"""
        self.uy = uy
        self.kocha = kocha
        self.tuman = tuman
        self.viloyat = viloyat

    def get_manzil(self):
        """Manzilni ko'rish"""
        manzil = f"{self.viloyat} viloyati, {self.tuman} tumani, "
        manzil += f"{self.kocha} ko'chasi, {self.uy}-uy"
        return manzil
```

Talaba klassimizga ham qo'shimcha manzil xususiyatini qo'shamiz:

```
class Talaba(Shaxs):
    """Talaba klassi"""
    def __init__(self, ism, familiya, passport, -
        tyil, id, manzil):
        """Talabaning xususiyatlari"""
        super().__init__(ism, familiya, passport, tyil)
        self.idraqam = id
        self.bosqich = 1
        self.manzil = manzil

    def get_id(self):
        """Talabaning ID raqami"""
        return self.idraqam

    def get_bosqich(self):
        """Talabaning o'qish bosqichi"""
        return self.bosqich

    def get_info(self):
        """Talaba haqida ma'lumot"""
        info = f"{self.ism} {self.familiya}. "
        info += f"{self.get_bosqich()}-bosqich. ID
        raqami: {self.idraqam}"
        return info
```

Keling, endi talaba obyektini qayta yaratamiz. Bu safar talabaning manzili ham alohida obyekt sifatida talaba ga uzatiladi:

```
manzil = Manzil(12, 'Olmazor', "Bog'bon", "Samarqand")
talaba = Talaba("Valijon", "Aliyev", "FA112299", 2000,
    "0000012", manzil)
```

Obyekt ichidagi obyektning xususiyatlari va metodlariga ham avvalgidek nuqta orqali murojrat etishimiz mumkin:

```
>>> print(talaba.manzil.get_manzil())
Samarqand viloyati, Bog'bon tumani, Olmazor ko'chasi, 12-uy
>>> print(talaba.manzil.tuman)
Bog'bon
```


AMALIYOT

- Talaba klassiga yana bir fanlar degan xususiyat qo'shing. Bu xususiyat parametr sifatida uzatilmasin va obyekt yaratilganida bo'sh ro'yxatdan iborat bo'lsin (`self.fanlar=[]`).
- Fan degan yangi klass yaratib va bu klassdan turli fanlar uchun alohida obyektlar yaratib.
- Talaba klassiga `fanga_yozil()` degan yangi metod yozing. Bu metod parametr sifatida Fan klassiga tegishli obyektlarni qabul qilib olsin va talabaning fanlar ro'yxatiga qo'shib qo'ysin.
- Talabaning fanlari ro'yxatidan biror fanni o'chirib tashlash uchun `remove_fan()` metodini yozing. Agar bu metodga ro'yxatda yo'q fan uzatilsa, "Siz bu fanga yozilmagansiz" xabarini qaytarsin.
- Yuqoridagi Shaxs klassidan boshqa turli voris klasslar yaratib ko'ring (masalan, Professor, Foydalanuvchi, So-tuvchi, Mijoz va hokazo).
- Har bir klassga o'ziga xos xususiyatlar va metodlar yuklang.
- Barcha yangi klasslar uchun `get_info()` metodini qayta yozib chiqing.
- Voris klasslardan yana boshqa voris klass yaratib ko'ring. Misol uchun, Foydalanuvchi klassidan Admin klassini yaratib.
- Admin klassiga foydalanuvchida yo'q yangi metodlar yozing, masalan, `ban_user()` metodi konsolga "Foydalanuvchi bloklandi" degan matn chiqarsin.

9.6. INKAPSULYATSIYA, KLASSGA XOS XUSUSIYAT VA METODLAR

INKAPSULYATSIYA

Obyektga yo'naltirilgan dasturlashning tamoyillaridan biri inkapsulyatsiya, ya'ni obyektning xususiyatlariga to'g'ridan to'g'ri (nuqta orqali) murojaat etishni va ularning qiymatini o'zgartirishni taqiqlab qo'yishdir. Pythonda bunday yopiq xususiyatlarning nomi ikki pastki chiziq (`__`) bilan boshlanadi:



<https://oloo/Da-inkaps>

```
from uuid import uuid4

class Avto:
    """Avtomobil klassi"""
    def __init__(self, make, model, rang, yil, narx, km=0):
        """Avtomobilning xususiyatlari"""
        self.make = make
        self.model = model
        self.rang = rang
        self.yil = yil
        self.narx = narx
        self.__km = km
        self.__id = uuid4()

    def get_km(self):
        return self.__km

    def get_id(self):
        return self.__id
```

Yuqoridagi klassni yaratishda `__init__` metodi ichidagi ikki xususiyatga e'tibor qaratib: `__km` va `__id`. Bu ikki xususiyat inkapsulyatsiya qilingan, ya'ni ularga tashqaridan to'g'ridan to'g'ri murojaat etib bo'lmaydi.

```
>>> avto1 = Avto("GM", "Malibu", "Qora", 2020, 40000, 100000)
>>> avto1.__km
AttributeError: 'Avto' object has no attribute '__km'
```

Ko'rib turganingizdek, `__km` xususiyatiga murojaat etishda kodimiz `AttributeError` xatosini qaytardi.

Inkapsulyatsiya qilingan xususiyatlarni ko'rish uchun alohida metodlar yozish maqsadga muvofiq bo'ladi (`get_km()` va `get_id()`):

```
print(f"ID: {avto1.get_id()}")
ID: 1d4f39a4-3222-4682-9231-6275ca5e1bff
```

① Har bir avto uchun noyob (takrorlanmas) ID yaratish uchun biz maxsus `uuid4()` funksiyasidan foydalandik.

Inkapsulyatsiyalangan xususiyatlarni o'zgartirish ham metodlar orqali amalga oshirilishi kerak. Masalan, mashinaning necha km yurganini o'zgartirish uchun klassimizga quyidagi metodni qo'shamiz:

```
class Avto:
    """Avtomobil klassi"""
    ***
    def add_km(self, km):
        """Mashinaning km ga yana km qo'shish"""
        if km >= 0:
            self.__km += km
        else:
            return "Mashina km kamaytirib bo'lmaydi"
```

Metodni tekshiramiz:

```
>>> avto1.add_km(1500)
>>> print(avto1.get_km())
>>> 101500
```

Inkapsulyatsiyaning maqsadi obyektning ma'lum xususiyatlarini tashqi ta'sirdan himoya qilishdir. Masalan, yuqorida-

gi misolimizda mashinaning qancha yurganini faqat musbat tarafga o'zgartirish mumkin, noyob ID raqamini esa umuman o'zgartirib bo'lmaydi.

KLASSNING XUSUSIYATLARI VA METODLARI

Klassdan yaratilgan har bir obyektning xususiyatlarini klass ichidagi `def __init__()` metodi yordamida beraymiz. Bu metod qanday ishlaydi? Biz har gal klassga murojaat etganimizda klass aynan shu `__init__` metodini ishga tushiradi va biz bergan xususiyatlar bilan yangi obyekt yaratadi.

Pythonda xususiyatlarni nafaqat obyektga, balki to'g'ridan to'g'ri klassga ham yuklash imkoniyati mavjud. Bunda klassning xususiyatiga berilgan qiymat barcha obyektlar uchun umumiy bo'ladi. Bu xususiyat bir obyekt orqali o'zgartirilganda shu klassga oid barcha obyektlarda ham uning qiymati o'zgaradi.

Klassga oid xususiyatlar `def __init__()` metodidan avval e'lon qilinadi.

Keeling, tushunarli bo'lishi uchun quyidagi misolni keltiramaniz:

```
class Avto:
    """Avtomobil klassi"""
    num_avto = 0
    def __init__(self, make, model, rang, yil, narx, km=0):
        """Avtomobilning xususiyatlari"""
        self.make = make
        self.model = model
        self.rang = rang
        self.yil = yil
        self.narx = narx
        self.__km = km
        self.__id = uuid4()
        Avto.num_avto += 1
```


Kodni tahlil etamiz:

- 1-qatorda Avto klassini e'lon qildik.
- 3-qatorda Avto **klassiga** oid bo'lgan xususiyat num_avto yaratdik va unga 0 qiymatini berdik.

- Keyingi qatorlarda __init__ metodi yordamida klassdan yaratiladigan obyektlarning xususiyatlarini e'lon qildik va har gal bu metodga murojaat etilganda num_avto ning qiymatini 1 ga oshiradigan qilib qo'ydik: Avto.num_avto += 1.

Yuqoridagi usul bilan endi dastur davomida Avto klassidan jami nechta obyekt yaratilganini kuzatib borishimiz mumkin bo'ladi. Bunda num_avto o'zgaruvchisiga istalgan obyekt orqali yoki klass nomi orqali murojaat etish mumkin:

```
>>> avto1 = Avto("GM", "Malibu", "Qora", 2020, 40000)
>>> avto2 = Avto("GM", "Lacetti", "Oq", 2020, 20000)
>>> print(avto1.num_avto)
2
>>> avto3 = Avto("Toyota", "Camry", "Silver", 2018, 45000)
>>> print(Avto.num_avto)
3
```

KLASSNING XUSUSIYATLARINI INKAPSULYATSIYA QILISH

Klassga oid xususiyatlar ham xuddi yuqoridagi kabi nomidan avval ikki pastki chiziq qo'shish bilan inkapsulyatsiya qilinadi:

```
class Avto:
    """Avtomobil klassi"""
    __num_avto = 0 # klassga oid xususiyat
    def __init__(self, make, model, rang, yil, narx):
        """Avtomobilning xususiyatlari"""
        self.make = make
        ***
        Avto.__num_avto += 1
```

KLASSGA OID METODLAR

Klasslarning o'ziga xos metodlari ham bo'lishi mumkin. Masalan, yuqoridagi num_avto xususiyatini ko'rish uchun alohida metod yozishimiz mumkin. Klassga oid metodlar @classmethod dekoratori bilan boshlanadi va, obyektga oid metodlardan farqli ravishda, self emas, cls (class) argumentini qabul qiladi.

```
class Avto:
    """Avtomobil klassi"""
    __num_avto = 0
    def __init__(self, make, model, rang, yil, narx, km=0):
        """Avtomobilning xususiyatlari"""
        self.make = make
        self.model = model
        self.rang = rang
        self.yil = yil
        self.narx = narx
        self.km = km
        self.__id = uuid4()
        Avto.__num_avto += 1

    @classmethod
    def get_num_avto(cls):
        return cls.__num_avto
```

Klass metodlarga klassning nomi orqali murojaat etiladi:

```
>>> avto1 = Avto("GM", "Malibu", "Qora", 2020, 40000)
>>> avto2 = Avto("GM", "Lacetti", "Oq", 2020, 20000)
>>> avto3 = Avto("Toyota", "Camry", "Silver", 2018, 45000)
>>> print(Avto.get_num_avto())
3
```

① @classmethod bu maxsus dekorator. Dekoratorlar – o'z ichiga funktsiya oluvchi funksiyalar.

KLASSLARNI MODULGA AJRATISH

Vaqt o'tishi bilan dasturimizda klasslar ko'payib borishi ta-biiy. Bizning asosiy dasturimiz uzun va tushunar-siz bo'lmashi-gi uchun klasslarni ham xuddi funksiyalar kabi alohida mo-dullarga ajratish maqsadga muvofiq bo'ladi. Dastur davomida kerak bo'ladigan klasslarga esa modulni chaqirish (import) orqali murojaat etamiz. Bunda bir-biriga bog'liq klasslarni bitta faylga joylashimiz mumkin.

Misol uchun, Talaba, Professor, Foydalanuvchi va Shaxs degan klasslarni bitta odamlar.py modulida, Avto, Bus, Train degan klasslarni esa boshqa transport.py modulida saqlab qo'yib, kelajakda ularga import orqali murojaat etsak bo'ladi.

Bitta klassni import qilish

Moduldan bitta klass import qilish uchun **from modul im-port** klass ifodasidan foydalanamiz:

```
from odamlar import Talaba
from transport import Avto

talaba=Talaba("Alijon", "Valiyev", "FA010101", "N00022")
avto = Avto("GM", "Malibu", "Qora", 2020, 40000)
```

Bir nechta klasslarni import qilish

Moduldan bir nechta klass chaqirish talab etilsa, import so'zidan so'ng klasslar ketma-ket vergul bilan ajratib yoziladi:

```
from odamlar import Talaba, Shaxs

talaba=Talaba("Alijon", "Valiyev", "FA010101", "N00022")
shaxs = Shaxs("Hasan", "Husanov", "FB0011223")
```

Modulning o'zini chaqirish

Modulni to'liq import qilish uchun import modul ifodasidan foydalanamiz. Bunda modul ichidagi klasslarga modul nomi va nuqta orqali murojaat etiladi:

```
import odamlar

talaba = odamlar.Talaba("Ali", "Valiyev", "FA010101", "N00022")
shaxs = odamlar.Shaxs("Hasan", "Husanov", "FB0011223")
```

Moduldagi barcha klasslarni import qilish

Moduldagi barcha klasslar quyidagicha import qilinadi: **from modul import ***. Bunda modul ichidagi istalgan klassga to'g'ridan to'g'ri uning nomi bilan murojaat etiladi.

```
from odamlar import *

talaba = Talaba("Ali", "Valiyev", "FA010101", "N00022")
shaxs = Shaxs("Hasan", "Husanov", "FB0011223")
```

Bu usul 2 sababga ko'ra tavsiya qilinmaydi:

1. Dasturni kelajakda qayta ochganimizda dastur davomida moduldagi aynan gaysi klasslardan foydalanganimizni bir qarashda bilib bo'lmaydi.

2. Agar modul juda katta bo'lsa, uning ichidagi ba'zi klasslarning biz o'zimiz yaratgan klasslar bilan nomi bir xil bo'lib qolish ehtimoli bor. Bu esa, o'z navbatida, dasturimiz xato ishlashiga olib keladi.

Bir modul ichiga boshqa modulni ham import qilish mumkin. Masalan, modul ichidagi ba'zi klasslar to'g'ri ishlashi uchun boshqa modul talab qilingan vaqtlarda.

AMALIYOT

- Avvalgi darslarimizda yaratgan Shaxs va Talaba klasslariga yopiq xususiyatlar qo'shing va ularning qiymatini ko'rsatuvchi va o'zgartiruvchi metodlar yozing.
- Yuqoridagi klasslarga talabalar_soni va odamlar_soni degan klassga oid xususiyatlar qo'shing.
- Klassga oid xususiyatlar bilan ishlash uchun maxsus @classmethod lar yozing.

9.7. DUNDER METODLAR

Pythonda obyektlar bilan ishlashni yanada qulay qilish uchun bir nechta maxsus metodlar bor. Bu metodlarning nomi ikki pastki chiziq bilan yozilgani uchun **double underscore** (ikki pastki chiziq) yoki qisqartirib **dunder** metodlar deb ataladi.

Dunder metodlar yordamida obyektlarga qo'shimcha qulayliklar va vazifalar qo'shishimiz mumkin.

Klass yoki obyektga oid dunder metodlar ro'yxatini ko'rish uchun `dir()` funksiyasidan foydalanamiz:

```
>>> dir(Avto)
['_Avto_num_avto',
 '_class_',
 '_delattr_',
 '_dict_',
 '_dir_',
 '_doc_',
 '_eq_',
 '_format_',
 '_ge_',
 '_getattribute_',
```



<https://olcoo/Da-dunder>

```
'_gt_',
'_hash_',
'_init_',
'_init_subclass_',
'_le_',
'_lt_',
'_module_',
'_ne_',
'_new_',
'_reduce_',
'_reduce_ex_',
'_repr_',
'_setattr_',
'_sizeof_',
'_str_',
'_subclasshook_',
'_weakref_',
'make',
'model',
'narx',
'rang',
'yil']
```

Dunder metodlardan biz `__init__` metodi bilan tanishdik. Bu metod klassdan obyekt yaratishda chaqiriladi va obyektning xususiyatlarini belgilaydi. Ushbu darsimizda esa maxsus metodlarning ba'zilarini bilan tanishamiz.

OBJEKT HAQIDA MA'LUMOT

Obyektga `print()` yoki `str()` orqali murojaat etilganda obyekt haqida tushunarli ma'lumot qaytarish uchun `__repr__` va `__str__` metodlaridan foydalanamiz. Tushunarli bo'lishi uchun avvalgi darsimizdagi Avto klassiga qaytamiz:

```

class Avto:
    num_avto = 0
    """Avtomobil klassi"""
    def __init__(self, make, model, rang, yil, narx):
        """Avtomobilning xususiyatlari"""
        self.make = make
        self.model = model
        self.rang = rang
        self.yil = yil
        self.narx = narx
        Avto.num_avto += 1

```

Yuqoridagi klassdan yangi obyekt yaratamiz va obyekt haqida ma'lumot olish uchun print() funksiyasini chaqiramiz:

```

>>> avto1 = Avto("GM", "Malibu", "Qora", 2020, 40000)
>>> print(avto1) # obyekt haqida ma'lumot
<__main__.Avto object at 0x0000238A6DAE0C8>

```

Qandaydir tushunarsiz ma'lumot. Ekrandagi natijadan biz faqat avto1 obyektimiz Avto klassiga tegishli ekanini ko'ramiz. Qanday qilib uning o'rniga obyekt haqida mazmunli ma'lumot olishimiz mumkin?

Gap shundaki, biz har gal obyektga print() (yoki str()) yoki repr() orqali murojaat etganimizda Python obyekt ichida __str__ yoki __repr__ metodlariga murojaat etadi. Agar biz bu metodlarni yozmagan bo'lsak, yuqoridagi kabi umumiy ma'lumot qaytaradi.

Ushbu metodlarni yangidan yozib, biz istagan ma'lumotni qaytaradigan qilishimiz mumkin. Odatda, yuqoridagi ikki metoddan birini yozish kifoya: __repr__ umumiyroq, __str__ esa batafsilroq ma'lumot olish uchun ishlatiladi.

Ikkalasidan birini tanlaganda __repr__ metodiga yon bosiladi, sababi, bu metod print(), str() va repr() funksiyalarining hammasi bilan ishlaydi. Keling, biz ham yuqoridagi klassimizga __repr__ metodini qo'shamiz:

```

class Avto:
    num_avto = 0
    """Avtomobil klassi"""
    ***
    def __repr__(self):
        """Obyekt haqida ma'lumot"""
        return f"Avto: {self.rang} {self.make} {self.model}"

```

Qaytadan print() funksiyasini chaqiramiz:

```

>>> avto1 = Avto("GM", "Malibu", "Qora", 2020, 40000)
>>> print(avto1)
Avto: Qora GM Malibu

```

Mana, endi natijamiz ancha tushunarli ko'rinishda chiqdi.

OBJEKT LARNI TAQQOSLASH

Taqqoslash operatorlari yordamida biz turli obyektlarni solishtirishimiz mumkin. Taqqoslash natijasi mantiqiy qiymat (True yoki False) ko'rinishida bo'ladi.

```

>>> x, y = 5, 10
>>> print(x > y)
False

```

Keling, endi Avto klassidan 2 ta obyekt yaratamiz va ularni taqqoslab ko'ramiz:

```

>>> avto1 = Avto("GM", "Malibu", "Qora", 2020, 40000)
>>> avto2 = Avto("GM", "Lacetti", "Oq", 2020, 20000)
>>> print(avto1 > avto2)
TypeError: '>' not supported between instances of 'Avto' and 'Avto'

```

Xatolik. Demak, bu ikki obyektни solishtirib bo'lmay ekan.

Biz taqqoslash operatorlariga murojaat etganimizda Python obyektlar ichida taqqoslash uchun maxsus metodlarni qidiradi, agar metodlar topilmasa, yuqoridagi kabi `TypeError` qaytaradi.

Taqqoslash metodlari quyidagilardan iborat:

Metod	Operator
<code>x.__lt__(self,y)</code>	<code>x<y</code>
<code>x.__le__(self,y)</code>	<code>x<=y</code>
<code>x.__gt__(self,y)</code>	<code>x>y</code>
<code>x.__ge__(self,y)</code>	<code>x>=y</code>
<code>x.__eq__(self,y)</code>	<code>x==y</code>
<code>x.__ne__(self,y)</code>	<code>x!=y</code>

Ikki obyektни taqqoslash uchun yuqoridagi obyektlardan yarmini uchun metodlar yozishimiz kifoya, masalan, `x<y` uchun metod yozsak, `x>y` metodini yozishimiz shart emas yoki `__le__` metodi `__ge__` metodini ham o'z ichiga oladi va hokazo.

Tushunarli bo'lishi uchun `Avto` klassiga obyektlarni solishtirish uchun metod yozamiz. Deylik, biz obyektlarni **narxi** bo'yicha solishtirmoqchimiz, unda klassimizga quyidagi metodlarni qo'shamiz (klassni to'liq yozmadik, faqat qo'shilgan metodlarni keltiramiz):

```
def __eq__(self,boshqa_avto):
    """Tenglik"""
    return self.narx == boshqa_avto.narx

def __lt__(self,boshqa_avto):
    """Kichik"""
    return self.narx<boshqa_avto.narx

def __le__(self,boshqa_avto):
    """Kichik yoki teng"""
    return self.narx<=boshqa_avto.narx
```

Kodimizga e'tibor qilsangiz, biz tenglik (`__eq__`) yoki kichiklikni (`__lt__`) tekshirmoqchi bo'lganimizda ikki avtoning aynan narxi bo'yicha solishtiramiz (`self.narx == boshqa_avto.narx`). Mana, endi avtolarni solishtirsak bo'ladi:

```
>>> avto1 = Avto("GM", "Malibu", "qora", 2020, 40000)
>>> avto2 = Avto("GM", "lacetti", "oq", 2020, 20000)
>>> print(avto1>avto2)
False
>>> avto3 = Avto("Honda", "Accord", "oq", 2017, 40000)
>>> print(avto1==avto3)
True
```

OBJEKT UZUNLIGI

Pythonida `len()` funksiyasi yordamida turli obyektlarning uzunligini bilishimiz mumkin, misol uchun, `matn`, `ro'yxat`, `lug'at`, `to'plan` va hokazo.

```
>>> matn = 'hello world'
>>> print(len(matn))
11
>>> sonlar = [12, 34, 56, 66]
>>> print(len(sonlar))
4
```

Pythonida `len()` funksiyasiga murojaat etganimizda Python funksiyaga uzatilgan obyektning ichidagi maxsus `__len__` metodiga murojaat etadi. Agar bu metod mavjud bo'lmasa, dasturimiz xato qaytaradi. Misol uchun, bizning `Avto` klassimizda bu metod yozilmagan, shu bois Python `TypeError` xatosini qaytardi.

```
>>> len(avto1)
Natija: TypeError: object of type 'Avto' has no len()
```

Keling, endi `__len__` metodini qanday ishlatishga ham misol keltiraylik. Boshlanishiga yangi, `AvtoSalon` degan klass yaratamiz. Bu klassimiz 2 ta xususiyatga ega: `salon nomi (name)` va `salondagi mashinalar (avtolar)`.

```
class AvtoSalon:
    """Avtosalon klassi"""
    def __init__(self, name):
        self.name = name
        self.avtolar = []

    def __repr__(self):
        return f"{self.name} avtosaloni"
```

Yuqoridagi klassdan yangi obyekt yaratamiz:

```
>>> salon1 = AvtoSalon("MaxAvto")
>>> print(salon1)
MaxAvto avtosaloni
```

`AvtoSalon` klassimizga `__repr__` metodini qo'shganimiz uchun natijamiz chiroyli ko'rinishda chiqdi.

Endi salonga avtomobil qo'shish uchun yangi `add_avto()` metodini yozamiz. Bu metodimiz `Avto` klassiga oid obyektlarni qabul qilishi kerak. `add_avto()` ga uzatilgan parametrlar `Avto` klassiga tegishli yoki tegishli emasligini tekshirish uchun `maxsus isinstance()` funksiyasidan foydalanamiz.

Bu funksiya biror obyekt ma'lum klassga tegishli ekanligini tekshirish uchun ishlatiladi:

```
>>> isinstance("salom", str)
True # "salom" bu str
>>> isinstance(9.5, int)
False # 9.5 int (butun son) emas
>>> isinstance(avto1, Avto)
True # avto1 Avto klassiga tegishli
```

`add_avto()` metodiga qaytamiz:

```
class AvtoSalon:
    """Avtosalon klassi"""
    def __init__(self, name):
        self.name = name
        self.avtolar = []

    def __repr__(self):
        return f"{self.name} avtosaloni"

    def add_avto(self, avto):
        if isinstance(avto, Avto):
            self.avtolar.append(avto)
        else:
            print("Avto obyekti kiriting")
```

Metodimizni tekshirib ko'ramiz:

```
# Avto obyektlarini yaratamiz
avto1 = Avto("GM", "Malibu", "Qora", 2020, 40000)
avto2 = Avto("GM", "Lacetti", "Oq", 2020, 20000)
avto3 = Avto("Toyota", "Carolla", "Silver", 2018, 45000)

# Yuqoridagi obyektlarni salon1 ga qo'shamiz
for avto in [avto1, avto2, avto3]:
    salon1.add_avto(avto)
```

Navbat `__len__` metodiga. Biz bu metod yordamida `len()` funksiyasi salonimizdagi avtomobillar sonini qaytaradigan qilamiz:

```
class AvtoSalon:
    """Avtosalon klassi"""
    ***
    def __len__(self):
        return len(self.avtolar)
```


Mana, endi bizning AvtoSalon klassimizga oid obyektlar uchun ham len() funksiyasini qo'llasak bo'ladi:

```
>>> print(len(salon1))
3 # Salonimizda 3 ta moshina bor
```

OBJEKT ELEMENTLARIGA MUHOJAT ETISH

Ba'zi obyektlarning (matn, ro'yxat, lug'at va hokazo) elementlariga alohida murojaat etish mumkin.

```
>>> mevalar = ['olma', 'anor', 'uzum']
>>> mevalar[0]
'olma'
```

Bizning salonimizda ham 3 ta avto bor, ularni ko'rish uchun yuqoridagi kabi element raqami orqali murojaat eta olamizmi?

```
>>> salon1[0]
TypeError: 'AvtoSalon' object is not subscriptable
```

Afsuski, yo'q. Ko'rib turganingizdek, bizning obyektimizga bunday murojaat etib bo'lmay ekan. Obyektimizga bu xususiyatni qo'shish uchun __getitem__ metodini yozishimiz kerak.

```
class AvtoSalon:
    """AvtoSalon klassi"""
    ***
    def __getitem__(self, index):
        return self.avtoalar[index]
```

Endi salon1 obyektimizning elementlariga murojaat etilganda __getitem__ metodi objekt ichidagi avtoalar ro'yxatidan ko'rsatilgan elementni (avtomobilni) qaytaradi.

```
>>> salon1[0]
Avto: qora GM Malibu
>>> salon1[1]
Avto: Oq GM Lacetti
>>> salon1[2]
Avto: Silver Toyota Carolla
>>> salon1[:3] # barcha elementlarni ko'rish
[Avto: qora GM Malibu, Avto: Oq GM Lacetti, Avto: Silver Toyota Carolla]
```

Keling, obyektlar elementlaridan birini o'zgartirib ko'ramiz:

```
>>> avto4 = Avto("Mazda", "6", 'qizil', 2015, 35000)
>>> salon1[0]=avto4
TypeError: 'AvtoSalon' object does not support item assignment
```

Yana xatolik. Gap shundaki, __getitem__ metodi o'z nomi bilan (get) element qaytaruvchi metoddir. Birov elementni o'zgartirish uchun esa __setitem__ metodini ham qo'shishimiz kerak. Bu metodimizga murojaat etilganda ham yangi qiymat Avto klassiga oid ekanligini tekshirib olish maqsadga muvofiq bo'ladi:

```
def __setitem__(self, index, value):
    if isinstance(value, Avto):
        self.avtoalar[index]=value
```

Qaytadan elementni o'zgartirishga harakat qilib ko'ramiz:

```
>>> avto4 = Avto("Mazda", "6", 'qizil', 2015, 35000)
>>> salon1[0]=avto4
>>> print(salon1[0])
Avto: qizil Mazda 6
```

Kutilgan natija chiqdi.

OPERATORLARNI QAYTA TALQIN QILISH (OVERLOADING)

Pythonda obyektlar o'rtasida turli arifmetik amallarni bajarish mumkin va bu amallar obyektning turiga qarab Python tomonidan turlicha talqin qilinadi. Masalan:

Sonlar:

```
>>> x,y=10,20
>>> x+y
30
>>> x*5
50
```

Matnlar:

```
>>> t1 = "hello"
>>> t2 = "world"
>>> t1+t2
'helloworld'
>>> t1*5
'hellohellohellohellohello'
```

Ro'yxatlar:

```
>>> l1 = [1, 2, 3]
>>> l2 = [4, 5, 6]
>>> l1+l2
[1, 2, 3, 4, 5, 6]
>>> l1*2
[1, 2, 3, 1, 2, 3]
```

va hokazo.

Keling, bu amallarni bizning obyektimizga ham qo'llab ko'ramiz. Boshlanishiga 2 ta avtosalon yarataylik va har biriga alohida avtomobillar qo'shaylik. Ishimizni osonlashtirish uchun `add_avto()` metodimizni ham istalgancha parametr qabul qilishga moslab o'zgartiramiz:

```
class AvtoSalon:
    """Avtosalon klassi"""
    def __init__(self, name):
        self.name = name
        self.avtolar = []
    def __repr__(self):
        return f"{self.name} avtosaloni"

    def __len__(self):
        return len(self.avtolar)

    def __getitem__(self, index):
        return self.avtolar[index]

    def __setitem__(self, index, value):
        if isinstance(value, Avto):
            self.avtolar[index]=value

    def add_avto(self, *qiyamat):
        for avto in qiyamat:
            if isinstance(avto, Avto):
                self.avtolar.append(avto)
            else:
                print("Avto obyketini kiriting")
```

Obyektlarni yaratamiz:

```
salon1 = AvtoSalon("MaxAvto")
salon2 = AvtoSalon("Avto Lider")
avto1 = Avto("GM", "Malibu", "Qora", 2020, 40000)
avto2 = Avto("GM", "Lacetti", "Oq", 2020, 20000)
avto3 = Avto("Toyota", "Carolla", "Silver", 2018, 45000)
avto4 = Avto("Mazda", "6", "Qizil", 2015, 35000)
avto5 = Avto("Volkswagen", "Polo", "Qora", 2015, 30000)
avto6 = Avto("Honda", "Accord", "Oq", 2017, 42000)
salon1.add_avto(avto1, avto2, avto3)
salon2.add_avto(avto4, avto5, avto6)
```


Mavzuga qaytamiz. Operatorlarni qayta talqin etish. Ke-
ling, boshlanishiga ikki obyektini qo'shib ko'ramiz:

```
>>> salon1+salon2
TypeError: unsupported operand type(s) for +: 'AvtoSa-
lon' and 'AvtoSalon'
```

Demak, bu ikki obyektini qo'shib bo'lmaz ekan. Biz obyekt
egasi sifatida qo'shish operatorini o'zimiz istagancha talqin
etishimiz mumkin. Misol uchun, AvtoSalon obyekti boshqa
AvtoSalon obyekti qo'shganda yangi AvtoSalon obyekti
qaytaraylik va bu yangi obyekt avvalgi ikki obyektning avto-
larga ega bo'lsin.

Qo'shish operatorini qayta talqin etish uchun AvtoSalon
klassemizga __add__ metodini qo'shamiz.

```
def __add__(self, qiymat):
    if isinstance(qiymat, AvtoSalon):
        yangi_salon=AvtoSalon(f'{self.name} {qiymat.name}')
        yangi_salon.avtolar=self.avtolar + qiymat.avtolar
        return yangi_salon
```

Qo'shish operatorini tekshirib ko'ramiz:

```
>>> salon3 = salon1+salon2
>>> print(salon3)
MaxAvto Avto lider avtosaloni
>>> salon3[:]
[Avto: Qora GM Malibu,
Avto: Oq GM Lacetti,
Avto: Silver Toyota Carolla,
Avto: Qizil Mazda 6,
Avto: Qora Volkswagen Polo,
Avto: Oq Honda Accord]
```

Istasak, qo'shish operatori yordamida salonga yangi Avto
qo'shish imkoniyatini ham yaratishimiz mumkin:

```
def __add__(self, qiymat):
    if isinstance(qiymat, AvtoSalon):
        yangi_salon = AvtoSalon(f'{self.name} {qiymat.name}')
        yangi_salon.avtolar = self.avtolar + qiymat.avtolar
        return yangi_salon
    elif isinstance(qiymat, Avto):
        self.add_avto(qiymat)
    else:
        print(f"AvtoSalon ga {type(qiymat)} qo'shib bo'lmaydi")
```

Tekshirib ko'ramiz:

```
>>> avto7 = Avto("BMW", 'X7', 'Qora', 2015, 75000)
>>> salon1 + avto7
>>> print(salon1[:])
[Avto: Qora GM Malibu, Avto: Oq GM Lacetti, Avto: Sil-
ver Toyota Carolla, Avto: Qora BMW X7]
```

Xuddi shu kabi, boshqa operatorlarni ham o'zingiz istal-
gancha talqin etishingiz mumkin:

Operator	Metod
Qo'shish	__add__
Ayirish	__sub__
Ko'paytirish	__mul__
Daraja	__pow__
Bo'lish	__div__

OBJEKTNI CHAQIRISH

Obyektlarni xuddi funksiyalarni chaqirgandek chaqirish
ham mumkin. Odatda, biror funksiyaga murojaat etganda
funksiya nomidan so'ng qavslar ochiladi va yopiladi. Agar
funksiya argument qabul qilsa, ular qavs ichida beriladi.

```
>>> print(10)
10
>>> len("salom")
5
```

Biz obyektimizga ham shunday imkoniyat qo'shishimiz mumkin. Buning uchun maxsus `__call__` metodidan foydalaniladi.

Parametrsiz chaqirish

`AvtoSalon` klassiga oid obyektlar chaqirilganda salondagi avtomobillar ro'yxatini ko'rsatadigan qilaylik. Buning uchun `AvtoSalon` klassiga quyidagi metodni qo'shamiz:

```
def __call__(self):  
    return [avto for avto in self.avtolar]
```

Mana, endi obyekt chaqirib ko'ramiz:

```
>> salon1()  
[Avto: GM Malibu. 40000$, Avto: GM Lacetti. 20000$,  
Avto: Toyota Carolla. 45000$]
```

Parametr bilan chaqirish

Yuqorida `salon1` ni parametrsiz chaqirishni ko'rib chiqdik. Endi parametr bilan chaqirishni ham yo'lga qo'yamiz. Bunda yuborilgan parametr yangi avto obyekti bo'lsin va bu parametr salondagi avtolar ro'yxatiga qo'shilsin. Metodimizni quyidagicha o'zgartiramiz:

```
def __call__(self,*param):  
    if param: # agar parametr bo'lsa,  
        for avto in param:  
            self.add_avto(avto)  
    else: # agar parametr bo'lmasa,  
        return [avto for avto in self.avtolar]
```

Endi bizning klassimizga ham parametr bilan, ham parametrsiz chaqirish imkoniyati qo'shildi.

```
>>> avto_new = Avto("Mercedes", 'E200', 'Silver', 2015, 80000)  
>>> salon1(avto_new) # Yangi avto qo'shamiz  
>>> salon1() # salondagi mashinalarni ko'ramiz  
[Avto: GM Malibu. 40000$, Avto: GM Lacetti. 20000$, Avto:  
Toyota Carolla. 45000$, Avto: Mercedes-Benz E200. 80000$]
```

`__call__` metodini qanday talqin etish ham sizning ixtiyoringizda.

SO'NGSO'Z

Ushbu bo'limda biz maxsus metodlarning ba'zilarini tanishtirdik. Bu metodlarning qulayligi shundaki, siz ularni istalgancha talqin etishingiz va har bir obyekt uchun mos keladigan qilib yaratishingiz mumkin. Ko'rib turganingizdek, dunder metodlar obyektingizning imkoniyatlarini kengaytiradi va ularga qo'shimcha vazifalar yuklaydi.

AMALIYOT

- Avvalgi darslarda yaratilgan obyektlarga (`Shaxs`, `Talaba`) turli dunder metodlarni qo'shishni mashq qiling.
 - Obyekt haqida ma'lumot (`__repr__`)
 - Talabalarni bosqichi bo'yicha solishtirish (`__lt__`, `__le__` va hokazo)
- Fan degan yangi klass yaratning. Fan obyektlari tarkibida shu fanga yozilgan talabalarning ro'yxati saqlansin. Buning uchun `Fanga.add_student()`, `__getitem__`, `__setitem__`, `__len__` kabi metodlarni qo'shing.
- `Fanga` qo'shish + operatori yordamida talaba qo'shish metodini yozing.
- Minus (-) operatori yordamida fandan talaba olib tashlash metodini yozing (bunda talabaning pasport raqami yoki ID raqami bo'yicha topib, olib tashlash mumkin).
- Fan obyektlarini chaqiriladigan qiling (masalan, `fizika()` yoki `fizika(talaba1)`). Bu metodlarni o'zingiz istagandek talqin eting.

10-BOB

FAYLLAR BILAN ISHLASH

Ushbu bo'limda katta hajmdagi ma'lumotlarni fayldan yuklab olish va dastur yakunida kerakli ma'lumotlarni va dastur natijasini faylga saqlashni o'rganamiz. Fayllar bilan ishlash dastur foydalanuvchilariga ham o'zlari istagan ma'lumotlarni dasturga yuklash imkonini beradi.



<https://elico/Da-fayl>

10.1. FAYLDAN MALUMOT O'QISH

Kompyuterimizda aksar ma'lumotlar fayl ko'rinishida saqlanadi. Xoh matn bo'lsin, xoh jadval, xoh rasm, xoh video. Fayllarda turli ma'lumotlar saqlanishi mumkin, ob-havo ma'lumotlari, yillik hisobotlar, mijozlarning telefon raqamlari, talabalarining baholari va hokazo.

Ko'pgina holatlarda dastur davomida katta ma'lumotlarni ayнан fayllardan o'qib olish talab qilinadi. Ayniqsa, tahliliy dasturlarda fayl ko'rinishida saqlangan, katta hajmdagi jadvallar bilan ishlash tabiiy. Lekin fayllar bilan ishlash boshqa holatlarda ham ko'p asqatadi, misol uchun, oddiy matni veb matn (html) ko'rinishiga o'tkazishni avtomatlashtiruvchi dastur yozishda.

Fayllar bilan ishlashning birinchi qadami fayldagi ma'lumotlarni kompyuter xotirasiga ko'chirishdir. Buning bir necha usuli bor, quyida ular bilan tanishamiz.

FAYLNI TO'LIQLIGACHA O'QISH

Boshlanishiga bizga fayl kerak. Keling, kompyuterda yangi pi.txt faylini yaratamiz va ichiga quyidagi matnни joylaymiz:

```
3.1415926535
8979323846
2643383279
```

Uch qatordan iborat faylimiz π sonining qiymatini 30 xona aniqlik bilan saqlaydi.

Faylni to'liq o'qish uchun quyidagi kodni yozamiz:

```
with open('pi.txt') as fayl:
    pi = fayl.read()
Kodni tahlil qilamiz:
```

- Birinchi qatorda `open()` funksiyasi yordamida faylni ochyamiz. Bunda funksiyaga argument sifatida fayl nomini berdik. Bu yerda biz ochayotgan fayl va dasturimiz bir papkada bo'lishi muhim.
- `open()` funksiyasi faylni obyekt sifatida qaytaradi, as operator yordamida esa biz obyektimizni fayl deb nomladik.
- Ikkinchi qatorda `.read()` metodi yordamida fayl obyektining tarkibidan bizga kerakli matnни olib, yangi, `PI` o'zgaruvchisiga yukladik.
- `with` operatorining vazifasi biz fayl bilan ishlab bo'lganimizdan so'ng faylni yopishdir. Yuqoridagi misolda 2-qatordan so'ng `Python` zudlik bilan faylni yopadi.

Yuqorida ko'rgan usulimiz fayl bilan ishlashning eng xavfsiz usulidir. Aslida, biz fayllarni to'g'ridan to'g'ri `fayl=open('pi.txt')` yordamida ochishimiz, fayl bilan ishlab bo'lgandan so'ng esa `fayl.close()` komandasi yordamida faylni yopishimiz ham mumkin edi:


```
fayl = open('pi.txt')
pi = fayl.read()
print(pi)
fayl.close()
```

Lekin bu usul xavfli hisoblanadi va tavsiya qilinmaydi. Gap shundaki, open() funksiyasi yordamida faylni ochganimizdan keyin toki close() metodini chaqirgunga qadar faylimiz ochiq holatda turadi. Agar faylni vaqtida yopmasak yoki fayl yopilmasidan avval dasturimiz to'xtab qolsa, fayl tarkibiga ziyon yetishi, ma'lumotlar yo'qotilishi mumkin. Boshqa dasturlarda ham (masalan, Microsoft Word) faylni yopmasdan oldin kompyuteringiz o'chib qolsa yoki dastur bexosdan yopilib ket-sa, faylingizga ziyon yetgani kabi.

Shuning uchun open() funksiyasiga with orqali murojaat etganimizda faylimiz with blokning oxirigacha ochiq turadi va with tugashi bilan fayl ham yopiladi. Demak, fayl ustidagi amallarni biz with bloki ichida bajarib olishimiz kerak.

Keling, endi pi ning qiymatini konsolga chiqaramiz:

```
>>> print(pi)
3.1415926535
8979323846
2643383279
```

Matn faylda qanday saqlangan bo'lsa, xuddi shu ko'rinishda konsolga chiqdi. Saqlangan ma'lumot son bo'lsa-da, fayldan o'qiganimizda qaytgan qiymat matn ko'rinishida bo'ladi. Matni songa o'tkazish uchun unga biroz ishlov beramiz:

```
#1 qator oxiridagi bo'shliqlarni olib tashlaymiz
>>> pi = pi.rstrip()
#2 qator tashlash belgisini almashtiramiz
>>> pi = pi.replace('\n', '')
#3 matnni float (o'nlik) songa o'tkazamiz
>>> pi = float(pi)
>>> print(pi)
3.141592653589793
```

① .replace() metodi matn tarkibidagi biror harf yoki belgini boshqa harf yoki belgi bilan almashtirish uchun ishlatiladi.

PAPKA ICHIDAGI FAYLLARNI OCHISH

Agar siz ochayotgan fayl dasturimiz bilan bir papkada emas, shu papka ichidagi papkada joylashgan bo'lsa, fayl nomidan avval papka nomi yoziladi:

```
with open('data/pi.txt') as fayl:
    pi = fayl.read()
```

Agar papkalar bir nechta qavat bo'lsa, fayl nomini va unga cha bo'lgan papkalarni alohida yozib olgan afzal:

```
faylnomi = 'data/math/numbers/pi.txt'
with open(faylnomi) as fayl:
    pi = fayl.read()
```

① Windowsda papkalar orasida \ belgisi ishlatilsa-da, Pythonda "/" belgisini ishlatamiz. Agar \ belgisini ishlatishni istasangiz, bu belgini 2 marta yozing: C:\\python\\darslar\\data

FAYLNI QATORMA-QATOR O'QISH

Ba'zida faylni to'liqligicha emas, qatorma-qator o'qish talab etilishi mumkin. Masalan, faylda talabalarning ismi yoki kundalik ob-havo ma'lumotlari saqlanganda va hokazo. Bunday hollarda for siklidan foydalanamiz:

```
filename = 'data/talabalar.txt'
with open(filename) as file:
    for line in file:
        print(line)
```


Natija:

```
alijon valiyev
hasan olimov
rahima mumino
```

Qatorlarni ro'yxat ko'rinishida saqlab olish uchun `.readlines()` metodidan foydalanamiz.

```
with open(filename) as file:
    talabalar = file.readlines()
    print(talabalar)
```

Natija:

```
['alijon valiyev\n', 'hasan olimov\n', 'rahima mumino-
va\n', 'hamida oqilova']
```

E'tibor bering, har bir talaba ismidan so'ng qator tashlash belgisi (`\n`) tushib qolgan. Biz bu belgilarni `.rstrip()` metodi yordamida olib tashlashimiz mumkin:

```
>>> talabalar = [talaba.rstrip() for talaba in talabalar]
>>> print(talabalar)
['alijon valiyev', 'hasan olimov', 'rahima mumino', 'ha-
mida oqilova']
```

10.2. FAYLGA MA'LUMOT YOZISH

Ma'lumotlarni saqlashning eng qulay usuli faylga yozishdir. Dasturimiz bajarilishdan to'xtaganidan so'ng xotiradagi ma'lumotlar o'chib ketishi mumkin, lekin faylga yozilgan ma'lumotlar saqlanib turaveradi. Fayllarni kelajakda qaytadan xotiraga yuklab, dasturimizni to'xtagan joyidan davom ettirishimiz mumkin.

Yuqorida biz faylni ochishda `open()` funksiyasidan foydalandik va yagona argument sifatida fayl nomini berdik. Bun-

da fayl faqatgina o'qish uchun ochiladi, unga yozib bo'lmaydi. Faylga ma'lumot yozish uchun `open()` funksiyasiga murojaat etishda fayl nomidan tashqari yana bir argument beramiz. Ikkinchi argument faylni aynan nima maqsadda ochishimizni ko'rsatadi.

Faylni ochish argumentlari va ularning vazifasi quyidagi jadvalda berilgan:

'r'	<code>open('file.txt', 'r')</code>	Faylni faqat o'qish uchun ochish (yozib bo'lmaydi)
'r+'	<code>open('file.txt', 'r+')</code>	Faylni o'qish va yozish uchun ochish.
'a+'	<code>open('file.txt', 'a+')</code>	Faylga ma'lumot qo'shish va o'qish uchun yozish. Fayl mavjud bo'lmasa, yangi fayl yaratiladi.

YANGI FAYLGA YOZISH

Yangi faylga ma'lumot yozish uchun `open()` funksiyasini chaqirishda 'w' (write) argumentidan foydalanamiz. Ochiq faylga ma'lumot qo'shish uchun esa `.write()` metodini chaqiramiz.

```
faylnomi = 'ustozlar.txt' # ochilayotgan fayl nomi
with open(faylnomi, 'w') as fayl:
    fayl.write('Hello World!') # faylga yozilayotgan matn
```

△ `open()` funksiyasini 'w' argumenti bilan chaqirganimizda ehtiyoq bo'lishimiz kerak, sababi, agar bunday fayl mavjud bo'lsa, uning ichidagi barcha ma'lumotlar o'chib ketadi.

Faylga yozayotgan ma'lumotlarimiz matn ko'rinishida bo'lishi kerak. Aks holda, dasturimiz xato beradi.

```
faylnomi = 'new_file.txt'
ism = 'Olimjon Hasanov'
tyil = 2004
with open(faylnomi, 'w') as fayl:
    fayl.write(ism)
    fayl.write(tyil)
```

Natija:

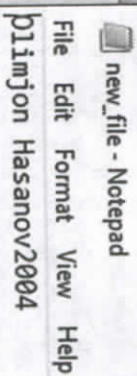
`TypeError: write() argument must be str, not int`

Xatoning oldini olish uchun sonlarni avval `str()` funksiyasi yordamida matnga keltirib olamiz.

```
faylnomi = 'new_file.txt'
ism = 'Olimjon Hasanov'
tyil = 2004
with open(faylnomi, 'w') as fayl:
    fayl.write(ism)
    fayl.write(str(tyil))
```

Fayllar matn formatida yoziladi, biz ularni istalgan matn muharriri yordamida ochib ko'rishimiz mumkin.

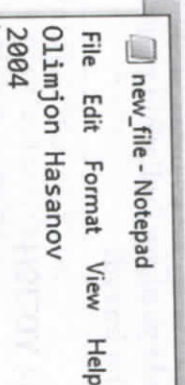
Afsuski, faylga bir nechta ma'lumot yozganimizda ma'lumotlar alohida qatordan emas, bir qatorda bir-biriga qo'shib yoziladi.



30-RASM. Faylda saqlangan ma'lumotlar

Buning oldini olishimiz uchun har bir qator so'ngida `\n` belgisini qo'shib ketishimiz kerak bo'ladi:

```
faylnomi = 'new_file.txt'
ism = 'Olimjon Hasanov'
tyil = 2004
with open(faylnomi, 'w') as fayl:
    fayl.write(ism+'\n')
    fayl.write(str(tyil)+'\n')
```



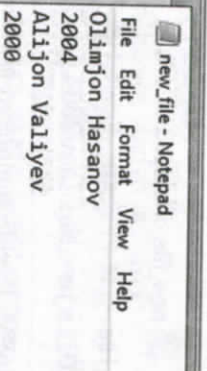
31-RASM. Fayldagi ma'lumotlar yangi qatordan saqlandi

FAYLGA MALUMOT QO'SHISH

Agar mavjud faylga ma'lumot qo'shish talab qilinsa, `open()` funksiyasiga murojaat etishda 'a' (append) argumentidan foydalanamiz. Bunda yangi qo'shilgan ma'lumotlar faylning oxiriga qo'shiladi.

```
with open(faylnomi, 'a') as fayl:
    fayl.write('Alijon Valiyev\n')
    fayl.write('2000')
```

Agar biz ochayotgan fayl mavjud bo'lmasa, Python yangi fayl yaratadi.



32-RASM. Faylga yangi ma'lumotlar qo'shildi

10.3. O'ZGARUVCHILARNI FAYLDA SAQLASH

Yuqorida biz ma'lumotlarni matn ko'rinishida saqlashni ko'rib o'tdik. Bu usul doim ham qulay bo'lmagligi mumkin. Agar dastur davomida turli o'zgaruvchilarni sof holatda faylda saqlash talab qilinsa, pickle modulidan foydalanamiz. Pickle ma'lumotlarni biz qanday ko'rinishda bersak, shunday ko'rinishda faylga yozadi. Yuqoridagi usuldan farqli ravishda, pickle yordamida yozilgan fayllarning tarkibini Pythondan tashqarida ko'rib bo'lmaydi.

PICKLE FAYLGA YOZISH

Pickledan foydalanish uchun biz avval bu modulni import qilamiz. Faylni ochishda esa open() funksiyasiga ikkinchi argument sifatida 'wb' (write binary) beramiz, ya'ni ikkilik sanoq tizimida yozishni ko'rsatamiz. Faylga yozish uchun pickle.dump() metodidan foydalanamiz:

```
import pickle

talabal = {'ism':'hasan', 'familiya':'husanov',
            'tyil':2003, 'kurs':2}
talabaz = {'ism':'alijon', 'familiya':'valiyev',
            'tyil':2004, 'kurs':1}

with open('info', 'wb') as file:
    pickle.dump(talabal, file)
    pickle.dump(talabaz, file)
```

E'tibor berib, yuqorida fayl nomini yozishda uning turini ko'rsatmadik, sababi, avval aytganimizdek, bu fayllar Pythondan tashqarida ochilmaydi va biz buning oldini olishimiz kerak. Aslida, fayl nomiga .txt qo'shimchasini ham qo'shishimiz mumkin, bu bilan dastur xato ishlamaydi, lekin bu bizni kelajakda chalg'itishi mumkin.

① Iltisangiz, faylga .dat (data so'zidan olingan) qo'shimchasini qo'shib qo'yishingiz mumkin (info.dat).

PICKLE FAYLDAN O'QISH

Pickle fayldan o'qish uchun open() funksiyasini 'rb' (read binary) argumenti bilan chaqiramiz. O'zgaruvchilarni bitta faylga yozganimizda har bir o'zgaruvchi alohida qatordan yoziladi. Fayldan o'qishda ham har bir qatorni alohida o'qishimiz kerak bo'ladi:

```
with open('info', 'rb') as file:
    talabal = pickle.load(file)
    talabaz = pickle.load(file)
```

O'zgaruvchilar tarkibini ko'ramiz:

```
>>> print(talabal)
{'ism': 'hasan', 'familiya': 'husanov', 'tyil': 2003,
 'kurs': 2}
>>> print(talabaz)
{'ism': 'alijon', 'familiya': 'valiyev', 'tyil': 2004,
 'kurs': 1}
```

Adashib ketmaslik uchun alohida o'zgaruvchilarni alohida fayllarga saqlash tavsiya qilinadi.

AMALIYOT

- Bugun o'rganangan narsalaringizni matnga yozing va matnini Python yordamida oching.
- Ushbu bog'lamadan https://cli.co/DA-pi_pi_million_digits.txt faylini yuklab oling (faylda π soni nuqtadan so'ng million xona aniqlik bilan yozilgan).
- Sizing tug'ilgan kuningiz π soni tarkibida uchrash yoki uchramasligini aniqlovchi funksiya yozing. Misol uchun, tug'ilgan sanangiz 25-fevral, 2000-yil bo'lsa, 25022000 ketma-ketligi yuqoridagi matnda uchrash-uchramasligini toping.
- Fayl ichidagi matni float ma'lumot turiga o'tkazing va pickle yordamida yangi faylga saqlang.
- Foydalanuvchidan turli xil ma'lumotlarni so'rab, har bir kiritilgan ma'lumotni yangi qatordan faylga yozib boruvchi dastur tuzing. Dastur qayta chaqirilganida yangi ma'lumotlar fayl oxiridan qo'shib borsin (yangi faylga emas).

10.4. JSON

JSON (**J**ava**S**cript **O**bject **N**otation) bugungi kunda ma'lumotlarni saqlash va internet orqali uzatish uchun qo'llaniladigan eng mashhur format hisoblanadi. Dastavval JavaScript tili uchun yaratilgan mazkur format bugungi kunda deyarli barcha dasturlash tillari tomonidan ishlatiladi. Qolaversa, JSON formatidagi fayllarning tarkibini oddiy matn muharriri yordamida ko'rish va tahrirlash mumkin.



<https://doi.org/10.1002/DA.1390>

Aksar holatlarda dastur va server orasidagi ma'lumotlar aynan JSON ko'rinishida uzatiladi. Quyidagi rasmda Wikipedia sahifasidan olingan ma'lumot ham JSON formatida berilgan:

```

1 // https://az.wikipedia.org/w/api.php?
2 format=json&action=query&prop=extract&showintro=&explain=text&redirects=1&title=AlisherNavoiy
3 {
4     "batchcomplete": "...",
5     "query": {
6         "pages": {
7             "1929": {
8                 "pageid": 1929,
9                 "ns": 0,
10                "title": "Alisher Navoiy",
11                "extract": "Bu muqaddas oʻzbek va boshqa turkiy xalqlarning shoiri Alisher Navoiy haqidir. Navoiy
12                soʻzining boshiga maʼnolari uchun mana bu satrlarga oʻting.Alisher Navoiy (1443-yil, 9-fevral-1501-yil, 3-yanvar) -
13                ulugʻ oʻzbek va boshqa turkiy xalqlarining shoiri, mufakkirlari va davlat arbobi boʻlgan. Gʻarbda chiqargʻay
14                adabiyotining buyuk vakili deb qaratiladi, shunday „mufton aliflari va dir“ (din va millatning nizomi) unvoni bilan
15                atalgʻanadi. Hali Vashiy nazariyasi tushgan, Navoiyon turkiy shoira isteʼdodiga yopiq baho bergan, kamo
16                Turbekiy etirofida oʻzomagan, asosiyda Hasan Ardashari, Pahlavon Muhammad kabi ustozlaridan taʼlim olgan, shuhratnomaga
17                Jamoai bilani ijodiy hamkorlikda boʻlgan.Navoiy 1469-yilgacha temuriylar orasidagi iktiki nizolar sababli Hirodon
18                yitiroqida yashagan,Amirshayroxo Navoiyoxuroson xarkatiga oʻtiribket (1469). Navoiy navoi va ijodida yangi boschi
19                boylandi, mahmudogʻli (1469) mansabatga, vazirlilik (1472) va Astrorod hozirlatigʻi (1477)ga tayinlandi. 1480-1500-
20                yillar mobaynida oʻz mahlagʻlari hisobidan bir necha madrasa, 40 rabot (safringargʻi yoʻlovchilar toʻrab oʻtish
21                joyi), 17 masjid, 10 xonagoh, 9 hammam, 9 koʻprikl, 20 ta hovuz qurdiradi. Husayn Boyqaro Alisher Navoiygʻa
22                „maqsadli harakati suloyisi“ („kutubon harakatlarining eng yagʻin kitablari“) degan unvoni beradi. Unga koʻra Navoiy
23                davlatning barcar ishlariga aralasht olardi. Volialsher Navoiy tarjimai holi oʻr davidda kondamiʼ, Vosifiy, Husayn
24                Boyqaro, Bobur kabi tarixtori va davlat arboblarning asarlarda aks etgan."
25            }
26        }
27    }

```

33-RASM. Wikipedia sahifasidan olingan
JSON formatdagi ma'lumot

Rasmga e'tibor bersangiz, JSON formatidagi ma'lumotlar Pythondagi lug'atlar kabi kalit so'z va qiymat ko'rinishida saqlangan. Lekin JSON yordamida biz natıqat lug'at, balki boshqa turdagi ma'lumotlarni ham saqlashimiz mumkin. Bunda Pythondagi ma'lumot turlari quyidagi jadval asosida JavaScript ma'lumot turlariga konvertatsiya qilinadi:

list	Array
str	String
float	Number
false	false

Demak, dasturimiz davomida ma'lumotlarni JSON ko'rinishida saqlashimiz, internet orqali boshqa foydalanuvchilarga, dasturlarga yoki serverga yuborishimiz, JSON fayllarni Pythonda ochib, unga ishlov berishimiz va turli amallar bajarishimiz mumkin.

① JSON ko'rinishidagi o'zgaruvchilar tarkibidan qat'i nazar, matn ko'rinishida saqlanadi.

PYTHONDAN JSONGA

JSON ma'lumotlar turi va fayllar bilan ishlash uchun Pythonda maxsus json moduli bor. Dasturimiz boshida biz bu modolni yuklab olishimiz kerak: `import json`.

Ma'lumotlarni JSON ko'rinishidagi matnga o'tkazish uchun ikki funksiyadan foydalanamiz:

- `json.dumps(x)` – berilgan `x` o'zgaruvchini JSON matniga o'zgartiradi.
- `json.dump(x, fayl)` – berilgan `x` o'zgaruvchini JSON ko'rinishiga o'zgartirib, ko'rsatilgan faylga yozadi.

json.dumps()

Ma'lumotlarni JSON formatiga o'tkazish uchun `json.dumps()` funksiyasidan foydalanamiz:

```
import json
x = 10
x_json = json.dumps(x)
ism = "anvar"
ism_json = json.dumps(ism)
sonlar = [12, 45, 23, 67]
sonlar_json = json.dumps(sonlar)
```

Yuqoridagi aytganimizdek, ko'p holatlarda JSON ma'lumotlar lug'at ko'rinishida uzatiladi.

```
bemor = {
    "ism": "Alijon Valiyev",
    "yosh": 30,
    "ola": True,
    "farzandlar": ("Ahmad", "Bonu"),
    "allergiya": None,
    "dorilar": [
        {"nomi": "Analgin", "miqdori": 0.5},
        {"nomi": "Panadol", "miqdori": 1.2}
    ]
}
bemor_json = json.dumps(bemor)
```

`print()` funksiyasi yordamida JSON tarkibini ko'rishimiz mumkin:

```
>>> print(bemor_json)
{"ism": "Alijon Valiyev", "yosh": 30, "ola": true,
"farzandlar": ["Ahmad", "Bonu"], "allergiya": null, "dorilar": [{"nomi": "Analgin", "miqdori": 0.5}, {"nomi": "Panadol", "miqdori": 1.2}]}
```

Yuqoridagi natija o'qish uchun noqulay ko'rinishda chiqdi. Buni to'g'rilash uchun `dumps()` funksiyasiga qo'shimcha `indent=4` parametrini beramiz. Bu parametr ma'lumotlarni saqlashda chapdan qancha joy tashlashni ko'rsatadi:

```
bemor_json = json.dumps(bemor, indent=4)
print(bemor_json)
```

Natija ham o'qishga qulay ko'rinishda chiqadi:

```
{
    "ism": "Alijon Valiyev",
    "yosh": 30,
    "ola": true,
    "farzandlar": [
        "Ahmad",
        "Bonu"
    ]
}
```

```

    ],
    "allergiya": null,
    "dorilar": [
        {
            "nomi": "Analgin",
            "miqdori": 0.5
        },
        {
            "nomi": "Panadol",
            "miqdori": 1.2
        }
    ]
}

```

Mavzu boshida JSON ichidagi ma'lumotlar JavaScript ma'lumot turlariga konvertatsiya qilinadi dedik. Buni yuqoridagi misolda ham ko'rishimiz mumkin (farzandlar, oila, allergiya kalitlari qiymatini asl lug'at bilan solishtirng).

json.dump()

Ma'lumotlarni JSON formatiga o'tkazish va faylga yozish uchun json.dump() funksiyasini chaqiramiz. Funksiya parametri sifatida o'zgaruvchi va fayl nomlarini ko'rsatamiz. Albatta, buning uchun avval faylni yozish uchun ochgan bo'lishimiz kerak:

```

bemor = {
    "ism": "Alijon Valiyev",
    "yosh": 30,
    "oilal": True,
    "farzandlar": ("Ahmad", "Bonu"),
    "allergiya": None,
    "dorilar": [
        {"nomi": "Analgin", "miqdori": 0.5},
        {"nomi": "Panadol", "miqdori": 1.2}
    ]
}

with open('bemor.json', 'w') as f:
    json.dump(bemor, f)

```

JSON fayl tarkibini istalgan matn muharririda ochib ko'rishimiz mumkin:

❏ **Listing 34-RASM. JSON fayl tarkibi**

```

File Edit Options Encoding Help
[...]\python-darslar\34-json\bemor.json
{"ism": "Alijon Valiyev", "yosh": 30, "oilal": true, "farzandlar": ["Ahmad", "Bonu"], "allergiya": null, "dorilar": [{"nomi": "Analgin", "miqdori": 0.5}, {"nomi": "Panadol", "miqdori": 1.2}]}

```

JSONDAN PYTHONGA

JSON formatidagi ma'lumotlarni Python ma'lumot turlariga keltirish uchun json.loads() yoki json.load() funksiyalaridan foydalanamiz. Yuqoridagi kabi json.loads() funksiyasi to'g'ridan to'g'ri JSON matn bilan ishlasa, json.load() funksiyasi JSON fayllarni o'qish uchun ishlatiladi.

json.loads()

Bu funksiya parameter sifatida JSON matn qabul qiladi va matnmi Python ma'lumot turlariga o'tkazadi.

```

sonlar = json.loads(sonlar_json)
bemor = json.loads(bemor_json)
print(bemor)

```

Natija:

```

{'ism': 'Alijon Valiyev', 'yosh': 30, 'oilal': True,
'farzandlar': ['Ahmad', 'Bonu'], 'allergiya': None, 'dorilar':
[{'nomi': 'Analgin', 'miqdori': 0.5}, {'nomi': 'Panadol',
'miqdori': 1.2}]}

```

E'tibor qiling, oila va allergiya kalitlarining qiymati qaytadan Python ma'lumot turlariga qaytdi.

json.load()

Bu funksiya JSON fayllarning tarkibini Pythonga yuklab olish uchun ishlatiladi.

```
filename = 'bemor.json'
with open(filename) as f:
    bemor = json.load(f)

print(type(bemor))
```

Natija:

```
<class 'dict'>
```

JSON BILAN ISHLASH

Ko'pincha internet orqali JSON fayllarni qabul qilganimizda ma'lumotlar bir necha qavatli lug'at ko'rinishida bo'ladi. JSON matnidan aynan o'zimizga kerakli ma'lumotni ajratib olish uchun lug'atni biroz tahlil qilish, uning kalitlari va qiymatlarini topish talab etilishi mumkin. Bu, ayniqsa, juda uzun JSON fayllarga tegishli. Shu bois JSON bilan samarali ishlash uchun lug'atlar bilan ishlashni yana bir bor takrorlab oling.

Quyidagi misolda Google Maps xizmati qaytargan JSON matni lug'at ko'rinishida berilgan. Bu Toshkent shahridagi Olmazor tumaniining geografik manzildir:

```
{
  "address_components": [
    {
      "long_name": "Almazar District",
      "short_name": "Almazar District",
      "types": [
        "political",
        "sublocality",
        "sublocality_level_1"
      ]
    }
  ]
}
```

```
]
{
  "long_name": "Tashkent",
  "short_name": "Tashkent",
  "types": [
    "locality",
    "political"
  ]
},
{
  "long_name": "Tashkent Region",
  "short_name": "Tashkent Region",
  "types": [
    "administrative_area_level_1",
    "political"
  ]
},
{
  "long_name": "Uzbekistan",
  "short_name": "UZ",
  "types": [
    "country",
    "political"
  ]
},
],
"formatted_address": "Almazar District, Tashkent, Uzbekistan",
"geometry": {
  "bounds": {
    "northeast": {
      "lat": 41.3954567,
      "lng": 69.269883
    },
    "southwest": {
      "lat": 41.3249733,
      "lng": 69.16497629999999
    }
  }
}
```

```

    }
    },
    "location": {
      "lat": 41.3645355,
      "lng": 69.2281531
    },
    "location_type": "APPROXIMATE",
    "viewport": {
      "northeast": {
        "lat": 41.3954567,
        "lng": 69.269883
      },
      "southwest": {
        "lat": 41.3249733,
        "lng": 69.16497629999999
      }
    }
  },
  "types": [
    "political",
    "sublocality",
    "sublocality_level_1"
  ]
}

```

Keling, shu ma'lumotlar orasidan tumanning geografik koordinatalarini topamiz. Avvalo, lug'atga ko'z yugurtirib chiqib, bizga kerak ma'lumotlar quyidagi ko'rinishda berilganini ko'rishimiz mumkin:

```

"location": {
  "lat": 41.3645355,
  "lng": 69.2281531
}

```

Bizga aynan latitude (kenglik) va longitude (uzunlik) qiymatlari kerak. Bu qiymatlar "location" kaliti ichidagi lug'at-

da lat va lng kalitlariga tegishli qiymatlarda joylashgan. location kalitining o'zi ham geometry kaliti ichida joylashganini ko'rishimiz mumkin.

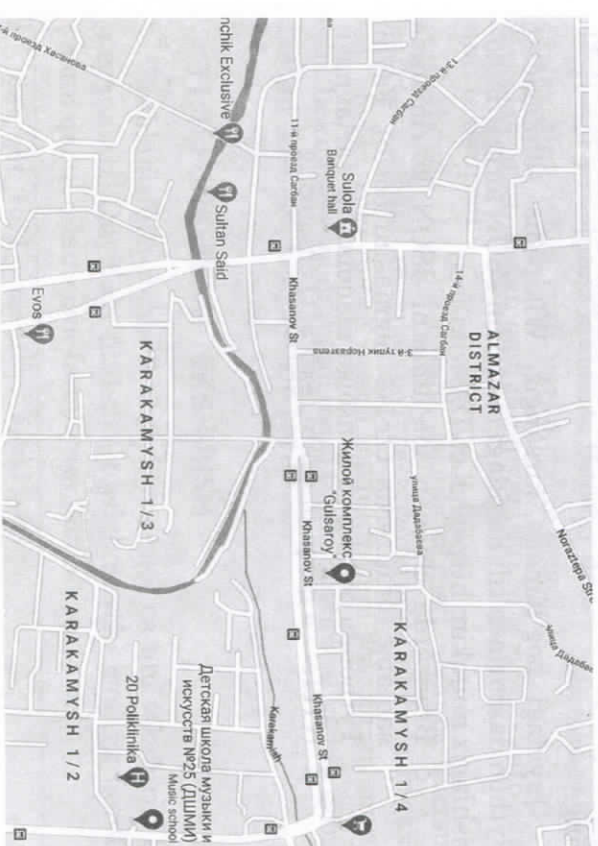
Demak, lug'at ichidan lat va lng qiymatlarini olish uchun quyidagi kodni yozamiz:

```

>>> kenglik = data['geometry']['location']['lat']
>>> uzunlik = data['geometry']['location']['lng']
>>> print(f"{kenglik},{uzunlik}")
41.3645355,69.2281531

```

Olingan koordinatalarni Google Maps xizmatiga kiritib ko'ramiz va quyidagi natijani olamiz.



35-RASM. Olmazor tumani

AMALIYOT

- Ushbu o'zgaruvchini JSON ko'rinishida saqlang va JSON matnini konsolga chiqaring: `data = {"Model": "Malibu", "Rang": "qora", "Yil": 2020, "Narx": 40000}`
- Ushbu JSON matni ko'chirib oling, talabani ismi va familiyasini konsolga chiqaring: `talaba_json = {"ism": "Hasan", "familya": "Husanov", "yil": 2000}""`
- Yuqoridagi ikki o'zgaruvchini alohida JSON fayllarga saqlang.

Quyidagi bog'lama orqali <https://cli.co/DA-students> JSON faylni yuklab oling. Faylda 3 ta talabani ismi va familiyasi saqlangan. Ularning har birini alohida qatordan "Ism Familiya, n-kurs, Fakultet talabasi" ko'rinishida konsolga chiqaring.

Quyidagi bog'lama kirsangiz, Wikipedia sahifasidagi Python dasturlash tili haqidagi maqolani JSON formatda ko'rishingiz mumkin: <https://cli.co/DA-wiki-json>

- Brauzerda chiqqan ma'lumotni JSON ko'rinishida saqlang (brauzerda Ctrl+S tugmasini bosib). Faylni Pythonda oching va konsolga maqolaning sarlavhasi (title) va qisqa matnini (extract) chiqaring

11-BOB

DASTURNI TEKSHIRISH

Dastur davomida yangi funksiya yoki klass yozar ekanmiz, ularning to'g'ri ishlashini ham tekshirishimiz tabiiy. Kodni tekshirish kelajakda dasturimiz xato ishlashining oldini oladi. Odatda, funksiya va klasslarni tekshirish uchun alohida test dasturlar yozishimiz mumkin. Bunday dasturlar funksiyaga turli parametrlar orqali murojaat etib, undan qaytgan qiymatlar to'g'ri yoki xato ekanini tekshiradi.

Pythonda bu jarayonni osonlashtirish uchun maxsus unittest moduli mavjud. unittest yordamida alohida funksiya, obyekt yoki butun boshli modulni ham tekshirish mumkin. Lekin tavsiya qilingan usuli testni dastavval kichik qismlardan boshlab, kengaytirib borishdir.



<https://cli.co/DA-test1>

11.1. FUNKSIYANI TEKSHIRISH

Boshlanishiga soddala funksiya yozamiz va uni tekshirishni o'rganamiz. Quyidagi funksiya foydalanuvchi ismi va familiyasini qabul qiladi, ism-familiyani jamlab qaytaradi:

```
def get_full_name(ism, familya):  
    return f"{ism} {familya}".title()
```

Funksiyani tekshiramiz:

```
>>> print(get_full_name('alijon', 'valiyev'))  
'Alijon Valiyev'
```

Keling, endi shu funksiyamizni tekshirish uchun dastur yozamiz. Bu yerda ikki narsaga e'tibor bering:

1. Avvalo, funksiyamizni alohida modulda saqlaymiz (name.py).

2. Ikkinchidan, test dasturimizni ham alohida modulda yozamiz va unga ham tushunarli nom beramiz (masalan, name_test.py).

Test faylimizning (name_test.py) tarkibi quyidagicha bo'ladi:

```
import unittest
from name import get_full_name

class NameTest(unittest.TestCase):
    def test_toliq_ism(self):
        formatted_name = get_full_name('alijon', 'valiyev')
        self.assertEqual(formatted_name, 'Alijon Valiyev')

unittest.main()
```

Dasturni tahlil qilamiz:

- Dastavval unittest modulini chaqiramiz: import unittest.
- Keyingi qatorda name.py modulimizdan tekshirmoqchi bo'lgan funksiyamizni ham yuklab olamiz: get_full_name.
- 4-qatorda test klassini yaratamiz, bu klass unittest.TestCase klassidan meros oladi. Bu klass berilgan parametrlar uchun funksiyadan qaytgan qiymatlarni tekshirishga mo'ljallangan. Klassimizga o'zimiz istagan, tushunarli nom beramiz: NameTest.
- Klassimiz ichida test_toliq_ism metodini yaratdik. Bu metod get_full_name funksiyasidan qaytgan qiymatning biz avvaldan bergan qiymatga teng yoki yo'q ekanini tekshiradi. Buning uchun esa maxsus .assertEqual() metodidan foydalandik.

△ Test metodlarning nomi har doim test so'zi bilan boshlanishi kerak.

- assertEquals() metodi ikki qiymat qabul qiladi va ularning teng ekanligini tekshiradi (assert ingliz tilidan tasdiqlash deb tarjima qilinadi). Agar get_full_name('alijon', 'valiyev') funksiyamiz to'g'ri ishlasa, funksiyadan 'Alijon Valiyev' qiymati qaytishi kerak. assertEquals() metodi aynan shuni tekshirishga mo'ljallangan.

- So'nggi qatorda unittest klassini chaqiramiz, bu esa, o'z navbatida, biz yuqorida yozgan testni chaqiradi. name_test.py dasturimizni bajaramiz va quyidagi natijani olamiz:

```
Ran 1 test in 0.001s
OK
```

Natijadan bitta test bajarilganini va bu test muvaffaqiyatli o'tganini (OK) ko'rishimiz mumkin.

Keling, endi dars boshida yaratilgan get_full_name funksiyamizga o'zgartirish kiritamiz:

```
def get_full_name(ism, familiya, otasi=''):
    if otasi:
        return f'{ism} {otasi} {familiya}'
    else:
        return f'{ism} {familiya}'.title()
```

Bu safar funksiyamiz otasining ismini ham qabul qiladi, lekin bu argument ixtiyoriy. Testimizga ham o'zgartirish kiritamiz. Bu safar ikki xil ism uchun ikkita alohida test yozamiz:


```
import unittest
from name import get_full_name

class NameTest(unittest.TestCase):
    def test_toliq_ism(self):
        formatted_name = get_full_name('Alijon', 'Valiyev')
        self.assertEqual(formatted_name, 'Alijon Valiyev')
    def test_toliq_ism_otasi(self):
        name = get_full_name('hasan', 'husanov', 'olimovich')
        self.assertEqual(name, 'Hasan Olimovich Husanov')

unittest.main()
```

Testni bajaramiz va quyidagi natijani olamiz:

```
AssertionError: 'hasan olimovich husanov' != 'Hasan
Olimovich Husanov'
- hasan olimovich husanov
? ^      ^      ^
+ Hasan Olimovich Husanov
? ^      ^      ^
-----
Ran 2 tests in 0.001s
FAILED (failures=1)
```

Bu safar funksiyamiz 1 ta testdan yiqildi (FAILED failures=1). Biz kutgan natija (Hasan Olimovich Husanov) va funksiya qaytargan natija (hasan olimovich husanov) bir xil emas. Ya'ni ism-familiya va otasining ismi katta harflar bilan qaytishi kerak edi, lekin natija unday emas. Demak, funksiyamizda xato bor. Xatoni to'g'rilaymiz:

```
def get_full_name(ism, familiya, otasi=''):
    if otasi:
        return f"{ism} {otasi} {familiya}".title()
    else:
        return f"{ism} {familiya}".title()
```

Testni qaytadan chaqiramiz:

```
Ran 2 tests in 0.001s
OK
```

Bu safar funksiyamiz ikki testdan ham muvaffaqiyatli o'tdi.

SONLARNI TEKSHIRISH

Yuqorida matn qaytaruvchi funksiyani tekshirishni ko'rdik. Keling, endi sonlar bilan ishlashni ko'zdan kechiramiz. Misol tariqasida yangi circle.py modulini yaratamiz, uning ichida doiraning yuzini va perimetrini hisoblaydigan funksiyalar yozamiz:

```
def getArea(r, pi=3.14159):
    """Doiraning yuzini qaytaruvchi funksiya"""
    return pi*(r**2)

def getPerimeter(r, pi=3.14159):
    """Doiraning perimetrini qaytaruvchi funksiya"""
    return 2*pi*r
```

E'tibor bering, ikki funksiya ham, agar foydalanuvchi aniq qiymat bermasa, π ning qiymatini standart argument sifatida 3.14159 ga teng deb qabul qilyapti. Ushbu funksiyalarni tekshirish uchun alohida circle_test.py test dasturini yozamiz. Matnlardan farqli ravishda, sonlarni taqqoslash uchun assertAlmostEqual() metodidan foydalanamiz. Bu metod ikki sonni nuqtadan keyin 7 xonagacha aniqlikda tekshiradi:

```
import unittest
from circle import getArea, getPerimeter

class CircleTest(unittest.TestCase):
    def test_area(self):
        self.assertAlmostEqual(getArea(10), 314.159)
        self.assertAlmostEqual(getArea(3), 28.27431)
    def test_perimeter(self):
        self.assertAlmostEqual(getPerimeter(10), 62.8318)
        self.assertAlmostEqual(getPerimeter(4), 25.13272)

unittest.main()
```

Bu safar har bir funksiya uchun ikkitadan test yozdik. Testni bajaramiz va quyidagi natijani olamiz:

```
Ran 2 tests in 0.001s
OK
```

MANTIQIY QIYMATLARNI TEKSHIRISH

Agar funksiya mantiqiy qiymat qaytarsa, bunday funksiyalarni `assertTrue()` va `assertFalse()` metodlari yordamida tekshirishimiz mumkin.

Quyidagi funksiyamiz kiritilgan son tub yoki tub emasligini tekshiradi:

```
def tubSonmi(n):
    if n==2 or n==3: return True
    if n%2==0 or n<2: return False
    for i in range(3, int(n**0.5)+1, 2):
        if n%i==0:
            return False
    return True
```

Funksiyani alohida `tubSonmi.py` faylga saqlaymiz. Funksiyani tekshirish uchun `tubSon_test.py` dasturini yozamiz:

```
import unittest
from tubSonmi import tubSonmi

class tubSonTest(unittest.TestCase):
    def test_true(self):
        self.assertTrue(tubSonmi(7))
        self.assertTrue(tubSonmi(193))
        self.assertTrue(tubSonmi(547))
    def test_false(self):
        self.assertFalse(tubSonmi(6))
        self.assertFalse(tubSonmi(265))
        self.assertFalse(tubSonmi(489))

unittest.main()
```

Test davomida `tubSonmi()` funksiyasini bir nechta tub (7, 193, 547) va tub bo'lmagan (6, 265, 489) sonlar bilan chagirdik. Bunda `assertTrue()` metodi funksiyamiz haqiqatan ham True qiymatini qaytarishini, `assertFalse()` metodi esa funksiyamiz False qiymat qaytarishini tekshiradi.

TAQQOSLASH METODLARI

TestCase klassi tarkibidagi barcha taqqoslash metodlari:

Metod	Nimani taqqoslaydi?
<code>assertEqual(a, b)</code>	<code>a == b</code>
<code>assertNotEqual(a, b)</code>	<code>a != b</code>
<code>assertTrue(x)</code>	x ning qiymati True
<code>assertFalse(x)</code>	x ning qiymati False
<code>assertIs(a, b)</code>	a bu b
<code>assertIsNot(a, b)</code>	a bu b emas
<code>assertIsNone(x)</code>	x ning qiymati None
<code>assertIsNotNone(x)</code>	x ning qiymati None emas
<code>assertIn(a, b)</code>	a b ning ichida
<code>assertNotIn(a, b)</code>	a b ning ichida emas
<code>assertIsInstance(a, b)</code>	a b ning vorisi
<code>assertNotIsInstance(a, b)</code>	a b ning vorisi emas

AMALIYOT

Quyidagi funksiyalarni yarating, **ularning har biri uchun test dasturlar yozing:**

- Uchta son qabul qilib, ulardan eng kattasini qaytaruvchi funksiya
- Matnlardan iborat ro'yxat qabul qilib, ro'yxatdagi har bir matnning birinchi harfini katta harfga o'zgartiruvchi funksiya
- Berilgan sonlar ro'yxatidan juft sonlarni ajratib oluvchi funksiya
- Berilgan son Fibonachchi ketma-ketligida uchraydimi (True) yoki yo'q (False) qaytaruvchi funksiya

11.2. KLASSLARNI TEKSHIRISH

Navbat klasslarni tekshiruvchi test dasturlarga. Klass to'g'ri bo'lsa, undan yaratilgan obyektlar ham to'g'ri ishlaydi.

Keling, boshlanishiga biror klass yaratamiz:



<https://olico/DA-test2>

```
class Car:
    """(self, make, model, year, km=0, price=None)"""
    def __init__(self, make, model, year, km=0, price=None):
        self.make = make
        self.model = model
        self.year = year
        self.price = price
        self.__km = km

    def set_price(self, price):
        self.price = price
```

```
def add_km(self, km):
    """Mashinaning km ga yana km qo'shish"""
    if km > 0:
        self.__km += km
    else:
        raise ValueError("Musbat qiymat kiriting!")

    def get_info(self):
        info = f"{self.make.upper()} {self.model.title()}, "
        info += f"{self.year}-yil, {self.__km}km yurgan."
        if self.price:
            info += f" Narxi: {self.price}"
        return info

    def get_km(self):
        return self.__km
```

Yuqoridagi klassimiz avtomobil haqida ma'lumotlarni saqlaydi. Klassimizning e'tibor qaratishimiz kerak bo'lgan jihatlari:

- km va price (narx) argumentlariga standart qiymat berilgan;
- __km parametri inkapsulyatsiyalangan (self.__km);
- avtomobil narxini set_price() metodi yordamida yangilash mumkin;
- add_km() metodi faqat musbat qiymat qabul qiladi. Agar manfiy qiymat uzatilsa, raise operatori yordamida ValueError xatosini qaytaradi;
- get_info() metodidan qaytadigan qiymat avtomobil narxi bor yoki yo'qligiga qarab turli ko'rinishda bo'lishi mumkin;
- avtomobil kilometrajini ko'rish uchun get_km() metodiga murojaat etamiz.

XUSUSIYATLARNI TEKSHIRISH

Klassdan obyekt yaratish jarayonida biz obyektning xususiyatlarini parametr ko'rinishida beramiz. Quyidagi testda aynan shu jarayon to'g'ri kechganini tekshiramiz:

```
import unittest
from cars import Car

class CarTest(unittest.TestCase):
    """Car klassini tekshirish uchun test"""
    def test_create(self):
        # avto1 obyektini km va narxini bermasdan yaratamiz
        avto1 = Car("toyota", "camry", 2020)
        # assertIsNone() qiymat mavjudligini tekshiradi
        self.assertIsNone(avto1.make)
        self.assertIsNone(avto1.model)
        self.assertIsNone(avto1.year)
        # assertIsNone qiymat mavjud emasligini tekshiradi
        self.assertIsNone(avto1.price)
        # assertEquals qiymatlar tengligini tekshiradi
        self.assertEqual(0, avto1.get_km())
        # Yangi obyekt yaratamiz va narxini ham ko'rsatamiz
        avto2 = Car("toyota", "camry", 2020, price=75000)
        self.assertEqual(75000, avto2.price)
    unittest.main()
```

Testimizni tahlil qilamiz:

- Dastavval obyektimiz to'g'ri yaratilayotganini tekshirish uchun avto1 obyektini 3 ta parametr bilan yaratib oldik (make, model, year) va bu xususiyatlar obyekt xususiyatlariga yozilganini assertIsNone() metodi bilan tekshirdik.
- avto1 obyektini yaratishda uning narxini ko'rsatmadik, demak, bu xususiyat standart qiymatga (None) teng bo'lishi

kerak. Buni tekshirish uchun esa assertIsNone() metodiga murojaat etdik.

- Nihoyat, avtomobil kilometraji 0 ga teng ekanligini assertEquals() metodi yordamida test qildik.
- Test so'ngida biz yana bir obyekt yaratdik (avto2) va bu safar avtomobil narxini ko'rsatganimiz uchun assertEquals() metodi yordamida bu qiymat to'g'ri saqlanganini tekshirib oldik.

Testni bajaramiz va quyidagi natijani olamiz:

```
Ran 1 test in 0.001s
OK
```

Testlarni yozishni davom etamiz. Navbat obyektga tegishli turli metodlarga.

△ Test dasturlarni alohida faylga yozishni unutmang.

setUp() METODI

Yuqoridagi misolda bitta test davomida 2 ta obyekt yaratdik, obyektning parametrlarini qo'lda yangidan kiritdik. Agar shu yo'sinda davom etadigan bo'lsak, turli testlar uchun har gal yangi obyekt yaratishimiz va ularning har biriga xususiyatlarni qayta-qayta kiritishimiz talab qilinadi. Buning oldini olish uchun test klassimizning avvalida setUp() metodini yaratib, bu metod ichida barcha kerakli qiymatlarni va obyektlarni saqlab qo'yishimiz, turli testlarda shu qiymatlarga murojaat etishimiz mumkin:


```

import unittest
from cars import Car

class CarTest(unittest.TestCase):
    """Car klassini tekshirish uchun test"""
    def setUp(self):
        make = "GM"
        model = "Malibu"
        year = 2020
        self.price = 40000
        self.km = 10000
        self.avto1 = Car(make, model, year)
        self.avto2 = Car(make, model, year, price=self.price)

    def test_create(self):
        self.assertIsNotNone(self.avto1.make)
        self.assertIsNotNone(self.avto1.model)
        self.assertIsNotNone(self.avto1.year)
        self.assertIsNone(self.avto1.price)
        self.assertEqual(0, self.avto1.get_km())
        self.assertEqual(self.price, self.avto2.price)

unittest.main()

```

E'tibor berib, `setUp()` metodi ichida ba'zi o'zgaruvchilar `self` yordamida berilgan (`self.price`, `self.km`, `self.avto1`, `self.avto2`). Bu o'zgaruvchilarga biz `CarTest()` klassining ichida istalgan joydan murojaat etishimiz mumkin. Shuning uchun ham `test_create()` funksiyasi ichida yangi obyekt yaratmasdan, `setUp()` ichidagi `avto1` va `avto2` obyektlariga murojaat etdik.

METODLARNI TEKSHIRISH

Obyektimiz bir nechta metodlardan iborat. Ularning har biri uchun alohida test yozamiz. Bu metodlarni `CarTest` ichida yozishni unutmang.

Osonidan boshlaymiz:

```

def test_set_price(self):
    self.avto2.set_price(self.price)
    self.assertEqual(self.price, self.avto2.price)

```

Endi `add_km()` metodini tekshiraylik. Bu metodimiz faqatgina musbat qiymat qabul qilishi, manfiy qiymat uzatilgan da `ValueError` xatosini qaytarishi kerak edi. Shuning uchun metodni test qilishda avval musbat, keyin esa manfiy qiymat berib ko'ramiz.

```

def test_add_km(self):
    #1 Musbat qiymat berib ko'ramiz
    self.avto1.add_km(self.km)
    self.assertEqual(self.km, self.avto1.get_km())
    #2 Manfiy qiymat berib ko'ramiz
    new_km = -5000
    try:
        self.avto1.add_km(new_km)
    except ValueError as error:
        self.assertEqual(type(error), ValueError)

```

Navbat `get_info()` metodiga. Bu metod ham obyektning xususiyatlaridan kelib chiqqan holda 2 xil qiymat qaytarishi mumkin, demak, testimiz bu ikki holatni hisobga olishi kerak.

```

def test_get_info(self):
    avto1_info = 'GM Malibu, 2020-yil, 0km yurgan.'
    self.assertEqual(avto1_info, self.avto1.get_info())
    # avto1 narxi va km o'zgartiramiz
    self.avto1.set_price(50000)
    self.avto1.add_km(20000)
    avto1_info = 'GM Malibu, 2020-yil, 20000km yurgan.'
    self.assertEqual(avto1_info, self.avto1.get_info())

```

Narxi: 50000'

Yakuniy test dasturimiz quyidagi ko'rinishga ega bo'ldi:

```
import unittest
from cars import Car

class CarTest(unittest.TestCase):
    """Car klassini tekshirish uchun test"""
    def setUp(self):
        make = "GM"
        model = "Malibu"
        year = 2020
        self.price = 40000
        self.km = 10000
        self.avto1 = Car(make, model, year)
        self.avto2 = Car(make, model, year, price=self.price)

    def test_create(self):
        # Qiymatlar mavjudligini tekshiramiz
        self.assertIsNotNone(self.avto1.make)
        self.assertIsNotNone(self.avto1.model)
        self.assertIsNotNone(self.avto1.year)
        # Qiymat mavjud emasligini tekshiramiz
        self.assertIsNone(self.avto1.price)
        # Qiymat tengligini tekshiramiz
        self.assertEqual(0, self.avto1.get_km())
        # avto2 narxini tekshiramiz
        self.assertEqual(self.price, self.avto2.price)

    def test_set_price(self):
        self.avto2.set_price(self.price)
        self.assertEqual(self.price, self.avto2.price)

    def test_add_km(self):
        # Musbat qiymat berib ko'ramiz
        self.avto1.add_km(self.km)
        self.assertEqual(self.km, self.avto1.get_km())
        # Manfiy qiymat berib ko'ramiz
        new_km = -5000
        try:
```

```
        self.avto1.add_km(new_km)
    except ValueError as error:
        self.assertEqual(type(error), ValueError)

    def test_get_info(self):
        avto1_info = 'GM Malibu, 2020-yil, 0km yurgan.'
        self.assertEqual(avto1_info, self.avto1.get_info())
        # avto1 narxi va km o'zgartiramiz
        self.avto1.set_price(50000)
        self.avto1.add_km(20000)
        avto1_info = 'GM Malibu, 2020-yil, 20000km yurgan.
        Narxi: 50000'
        self.assertEqual(avto1_info, self.avto1.get_info())

unittest.main()
```

Testni bajaramiz va quyidagi natijani olamiz:

```
....
-----
Ran 4 tests in 0.002s
OK
```

AMALIYOT

- 9.6-bo'limda Shaxs va Talaba klasslarini yaratgan edik. Shu ikki klassning xususiyatlari va metodlarini tekshiruvchi test dasturlar yozing.
- Ikki klass uchun yagona test yoza olasizmi? (isInstance() funksiyasini eslang)

12-BOB

KUTUBXONALAR

Python dasturlash tili yildan yilga ommalashib borishiga sodd va tushunarli sintaksis sabab bo'lsa, ikkinchi va, ehtimol, eng muhim sabab Pythonning keng qamrovli kutubxonalar to'plamidir. Ushbu bobda Python dasturlash tilining ichki va tashqi kutubxonasidagi ba'zi muhim modullar bilan tanishamiz.



<https://docs.python.org/3/library/datetime.html>

① Kutubxona – boshqalar tarafidan yozilgan tayyor funksiyalar, o'zgaruvchilar va obyektlar to'plami.

12.1. PYTHON STANDART KUTUBXONASI

Python dasturlash tili yuzlab tayyor modullar bilan o'rnatiladi. Bu modullar jamlanmasi standart kutubxona deb ataladi. Ushbu bo'limda standart kutubxonaga kiruvchi ba'zi modullar bilan tanishamiz. Kutubxonaning to'liq tarkibini esa quyidagi bog'lama orqali ko'rishingiz mumkin: <https://docs.python.org/3/library/>

datetime – SANA VA VAQT

Ushbu modul yordamida Pythonda sanalar bilan ishlaymiz. Har qanday moduldan foydalanishdan avval uni `import` yordamida chaqirib olishimiz kerak:

```
import datetime as dt
```

Hozirgi vaqt va sanani ko'rish uchun `datetime.now()` metodiga murojaat etamiz:

```
hozir = dt.datetime.now()
print(hozir)
```

Natija: 2021-03-09 12:12:19.894899

Kurib turganingizdek, natija yil, oy, kun, soat, minut, sekund va mikrosekund ko'rinishida chiqdi. Biz bu qiymatlarning har birini maxsus metodlar yordamida ajratib olishimiz mumkin:

```
# sanani ajratib olish
>>> print(hozir.date())
2021-03-09
# vaqtni ajratib olish
>>> print(hozir.time())
12:15:35.367013
# soatni ajratib olish
>>> print(hozir.hour)
12
# minutni ajratib olish
>>> print(hozir.minute)
15
# sekundni ajratib olish
>>> print(hozir.second)
35
```

Agar bugungi kunning sanasi talab qilinsa, `datetime` moduli ichidagi `date.today()` metodiga murojaat etamiz.

```
>>> bugun = dt.date.today()
>>> print(f"Bugungi sana: {bugun}")
Bugungi sana: 2021-03-09
```

Agar biror sanani qo'lda kiritish talab qilinsa, `date()` metodiga kerakli sanani (yil, oy, kun ko'rinishida) kiritamiz.

```
>>> ertaga = dt.date(2021, 3, 10)
>>> print(f"Ertangi sana: {ertaga}")
Ertangi sana: 2021-03-10
```

Faqatgina vaqt bilan ishlash uchun `.datetime.now().time()` metodiga murojaat etamiz:

```
>>> hozir = dt.datetime.now()
>>> vaqthozir = hozir.time()
>>> print(f"Hozir soat: {vaqthozir}")
Hozir soat: 12:21:54.529788
```

Istalagan vaqtni qo'lda kiritish uchun esa `.time()` metodiga kerakli vaqtni (soat, minut, sekund ko'rinishida) beramiz:

```
>>> vaqtkeyin = dt.time(23,45,00)
```

Ayirish operatori yordamida sanalar va vaqtlar orasidagi fargni hisoblashimiz mumkin:

```
bugun = dt.date.today()
ramazon = dt.date(2021, 4, 13)
farg = ramazon-bugun
print(f"Ramazonga {farg.days} kun qoldi")
```

Natija:

Ramazonga 35 kun qoldi

Xuddi shu kabi ikki vaqt oraliq'ini sekundlarda yoki soatlarda ham ko'rishimiz mumkin:

```
hozir = dt.datetime.now()
futbol = dt.datetime(2021, 3, 10, 23, 45, 00)
farg= futbol-hozir
sekundlar = farg.seconds
minutlar = int(sekundlar/60)
soatlar = int(minutlar/60)
print(f"Futbol boshlanishiga {sekundlar} sekund qoldi")
print(f"Futbol boshlanishiga {minutlar} minut qoldi")
print(f"Futbol boshlanishiga {soatlar} soat qoldi")
```

Natija:

```
Futbol boshlanishiga 40797 sekund qoldi
Futbol boshlanishiga 679 minut qoldi
Futbol boshlanishiga 11 soat qoldi
```

Yuqorida sanalar AQSh standartiga ko'ra yil-oy-kun ko'rinishida chiqyapti. Sanani o'zimizga moslab chiqarish uchun `.strftime()` metodini chaqiramiz va sanani o'zimizga qulay formatda chiqaramiz.

```
# vaqtni millisekundsiz chiqaramiz
>>> vaqt = hozir.strftime("%H:%M:%S")
>>> print(f"Hozir soat: {vaqt}")
Hozir soat: 12:28:21
# sanani kun-oy-yil ko'rinishida chiqaramiz
>>> sana = hozir.strftime("%d-%m-%Y")
>>> print(f"Bugun sana: {sana}")
Bugun sana: 09-03-2021
# sanani kun/oy/yil ko'rinishida chiqaramiz
>>> sana_vaq = hozir.strftime("%d/%m/%Y, %H:%M")
>>> print(sana_vaq)
09/03/2021, 12:28
```

math – MATEMATIK FUNKSIYALAR

Bu modul bilan 8.6-bo'limda qisqacha tanishgan edik. `math` o'z ichiga matematikaga oid turli funksiyalar va o'zgaruvchilarni oladi. Keling, ularning ba'zilari bilan tanishamiz.

π ning qiymati

```
>>> import math
>>> PI = math.pi
>>> print(f"PI ning qiymati: {PI}")
PI ning qiymati: 3.141592653589793
```


e – natural logarifm asosi

```
>>> E = math.e
>>> print(f"e ning qiymati: {E}")
e ning qiymati: 2.718281828459045
```

Trigonometriya

math moduli tarkibida deyarli barcha trigonometrik funksiyalar mavjud (cos, sin, tangens, arccos va hokazo)

```
math.sin(math.pi/2)
math.cos(0)
math.tan(PI)
```

Shuningdek, degrees va radians metodlari yordamida burchakdan radianga va, aksincha, konvertatsiya qilishimiz ham mumkin:

```
math.degrees(math.pi/2)
math.radians(90)
```

Logarifmlar

log() va log10() funksiyalari yordamida natural va o'n asosli logarifmlarni hisoblash mumkin:

```
# natural logarifm
math.log(5)
# 10 asosli logarifm
math.log10(100)
```

Sonlarni yaxitlash

Sonlarni yaxitlash uchun Pythonda maxsus round() funksiyasi mavjud. Bunga qo'shimcha ravishda math moduli ichidagi ceil() funksiyasi yordamida berilgan o'nlik sonni keyingi butun songa, floor() yordamida esa quyi butun songa yaqinlashtirish mumkin:

```
>>> x = 4.6
>>> print(math.ceil(x))
5
>>> print(math.floor(x))
4
```

Ildiz va daraja

Berilgan sonning kvadrat ildizini hisoblash uchun sqrt(), sonni darajaga oshirish uchun esa pow() funksiyalariga murojaat etamiz:

```
>>> x = 81
# Kvadrat ildiz
>>> math.sqrt(x)
9.0
# Darajaga oshirish
>>> math.pow(x,3) # x ning kubi
531441
>>> math.pow(x,5) # x ning 5-darajasi
3486784401
>>> math.pow(125,1/3) # 125 dan kub ildiz
5
```

math moduli tarkibida boshqa funksiyalar ham mavjud. Yuqorida biz ularning ba'zilari bilan tanishdik. Bu modul asosan butun va o'nlik sonlar bilan ishlashga mo'ljallangan. Kompleks sonlar bilan ishlash uchun esa alohida cmath moduli mavjud.

pprint – CHIROYLI PRINT

pprint moduli yordamida turli o'zgaruvchilarni chiroyli ko'rinishda konsolga chiqarishimiz mumkin. Bu bizga uzun lug'atlar, JSON fayllar yoki matnlar bilan ishlashda juda asqatadi.

Misol uchun, avvalgi darslarimizning birida yaratgan `bemor.json` faylini ochamiz va avval `print()`, keyin `pprint()` yordamida konsolga chiqaramiz.

```
from pprint import pprint
import json

filename = 'bemor.json'
with open(filename) as f:
    bemor = json.load(f)

print(bemor)
```

Natija:

```
{'ism': 'Alijon Valiyev', 'yosh': 30, 'ola': True,
'farzandlar': ['Ahmad', 'Bonu'], 'allergiya': None, 'do-
rilar': [{'nomi': 'Analgin', 'miqdori': 0.5}, {'nomi':
'Panadol', 'miqdori': 1.2}]}
```

Navbat `pprint()` funksiyasiga:

```
pprint(bemor)
```

Natija:

```
{'allergiya': None,
'dorilar': [{'miqdori': 0.5, 'nomi': 'Analgin'},
{'miqdori': 1.2, 'nomi': 'Panadol'}],
'farzandlar': ['Ahmad', 'Bonu'],
'ism': 'Alijon Valiyev',
'ola': True,
'yosh': 30}
```

Regex – ANDOZA YORDAMIDA MATN IZLASH

Pythondagi juda foydali modullardan biri `re` (regular expressions) modulidir. Bu modul yordamida biror matn berilgan andozaga tushish-tushmasligini tekshiramiz. Yoki berilgan andoza asosida matnlar orasidan kerakli matnlarni ajratib olish mumkin.

Keling, boshlanishiga soddami solishtiramiz. Quyida biz 3 ta soʻz va soʻzlarni tekshirish uchun andoza yaratdik. Quyidagi andozamiz `r` harfdan boshlanuvchi (`^r`), `p` harfi bilan tugovchi (`p$`), 5 harfdan iborat soʻzlarni qidiradi (`^r...p$`).

Avvaliga andozalarni tushunish biroz qiyin boʻlishi mumkin, lekin vaqt oʻtishi bilan andoza qanday ishlashini tushunib olasiz deb umid qilamiz.

Soʻzlarni andozaga solishtirish uchun `re.match()` funksiyasidan foydalanamiz. Agar tekshirgan soʻzimiz andozaga mos tushsa, `re.match()` metodi soʻzning oʻzini qaytaradi, aks holda, `None` qiymatini qaytaradi.

```
import re

word1 = "темр"
word2 = "томр"
word3 = "тылор"

andoza = "^r...p"
```

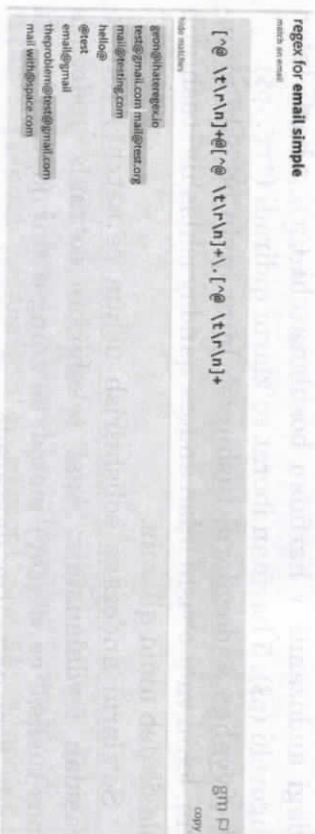
```
print(re.match(andoza, word1))
print(re.match(andoza, word2))
print(re.match(andoza, word3))
```

Natija:

```
<re.Match object; span=(0, 5), match='темр'>
<re.Match object; span=(0, 5), match='томр'>
None
```


Natijadan ko'rishimiz mumkin, word1 va word2 o'zga-ruvchilari andozaga tushdi, word3 esa tushmadi.

Andozalar biror matnda biz uchun kerakli ma'lumotlarni ajratib olish uchun juda qulay. Masalan, Telegram orqali yuborilgan xabardan e-mail manzilini yoki telefon raqamini ajratib olish uchun maxsus andoza yozishimiz mumkin. ihaterexex.io sahifasidan esa loyihangiz uchun tayyor andozalarni topishin-giz mumkin.



36-RASM. ihaterexex.io sahifasida e-mail uchun andoza

Keling, yuqoridagi andoza asosida biror matndan e-mail manzilini ajratib olamiz. Buning uchun re.findall() funksiyasidan foydalanamiz.

```
matn = """Maqolalar 2020-yilning 20-martiga qadar  
rtmkonferensiya2021@mail.ru elektron pochtaida qabul  
qilinadi.  
Quyidagi yo'nalishdagi maqolalar qabul qilinadi:  
✪ Aniq va tabiiy fanlarni zamonaviy pedagogik texno-  
logiyalar asosida o'qitish metodikasi.  
✪ Umumta'lim fanlarini o'qitishda STEAM yondashuv-  
ning o'rni va ahamiyati. """  
andoza = '[^@ \t\r\n]+\.[^\t\r\n]+\.[^\t\r\n]+'  
email = re.findall(andoza, matn)  
print(email)
```

Natija:

```
['rtmkonferensiya2021@mail.ru']
```

Andoza yordamida foydalanuvchi kiritgan qiymatlarning ham ma'lum shartlarga javob berishini tekshirib olishimiz mumkin:

```
# Kuchli parolni tekshirish  
# Quyidagi andoza ham ihaterexex.io sahifasidan olindi  
andoza = '^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])  
(?=.*?[#!@$%^&*~]).{8,}$'  
msg = "Yangi parol kiriting"  
msg += '(kamida: 8 xona, 1 ta lotin bosh va 1 ta kichik, '  
msg += '1 ta son va 1 ta maxsus belgi bo'lishi kerak): '  
  
while True:  
    password = input(msg)  
    if re.match(andoza, password):  
        print("Maxfiy so'z qabul qilindi")  
        break  
    else:  
        print("Maxfiy so'z talabga javob bermadi")
```

Python kutubxonasi yuzlab foydali modullardan iborat. Biz ulardan ba'zilari bilan tanishdik, xolos.

AMALIYOT

- Bugungi sanadan boshlab 2 hafta farq bilan 10 ta sanani konsolga chiqaring.
- Ramazon va qurbon hayitigacha qolgan kunlarni konsolga chiqaring.
- Tug'ilgan kuningizdan bugungi sanagacha qancha yil, oy, kun o'tganini qaytaruvchi funksiya yozing.
- Foydalanuvchidan telefon raqamini kiritishni so'rang. Kiritilgan qiymatni andoza yordamida tekshiring.

```
matn=""Assalomu alaykum, hurmatli do'stlar. Navbat-  
dagi darsimiz YouTube'ga yuklandi: https://youtu.be/vsx-  
JpRLXpGI  
Bugungi dars manzili: https://python.sariq.dev/tes-  
ting/37-klass-test""
```

12.2. TASHQI KUTUBXONALAR

Avvalgi bo'limda Python bilan birga o'rnatiluvchi standart kutubxona va undagi ba'zi foydali modullar bilan tanishdik. Ushbu darsimizda esa tashqi kutubxona bilan tanishamiz. Bu kutubxonalar yillar davomida turli foydalannuvchilar tarafidan yaratilib, yangilanib keladi. Bunday kutubxonalarda boshqa dasturchilar o'zlar yaratgan turli paketlarni boshqalar bilan ulashishadi.



<https://pypi.org/project/Da-pip/>

① Paket (package) – modullar yig'indisi.

Tashqi kutubxonalar va ular ichidagi paketlar shunchalik ko'pki, deyarli istalgan vazifa yoki xizmat uchun katta ehtimol bilan kerakli funksiyalar allaqachon bir nechtadan yaratilgan. Bugungi kunda Python uchun eng katta tashqi kutubxonalardan biri PyPi.org sahifasidir.

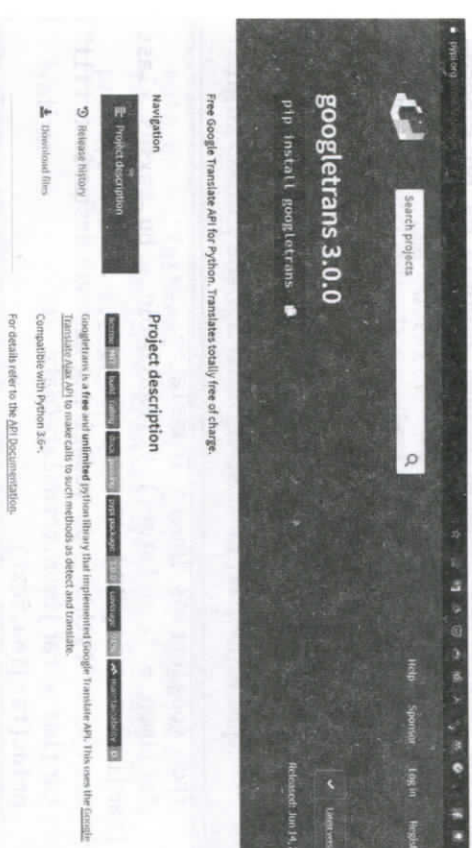
PIP

Tashqi paketlarni o'rnatish uchun Python'da maxsus pip paket menejeri mavjud. pip, odatda, Python bilan birga o'rnatiladi, lekin turli sabablarga ko'ra kompyuteringizda pip o'rnatilmagan bo'lsa, uni quyidagi sahifadan yuklab olishingiz mumkin: <https://pypi.org/project/pip/>

Paket menejer yordamida tashqi paketlarni o'rnatish juda oson. Buning uchun Windowsda Windows terminal dasturini

ochamiz va qora oynada pip install paket_nomi komandasini yozamiz.

Paket nomi qanday yozilishini paketning rasmiy sahifasidan ko'rib olishingiz mumkin. Misol uchun, quyidagi rasmda googletrans paketining sahifasi va pip komandasi qanday yozilishi ko'rsatilgan.



37-RASM. googletrans paketi sahifasi

Demak, ushbu paketni o'rnatish uchun Command Prompt oynasida pip install googletrans deb yozamiz.

Biror paketni o'chirib tashlash uchun esa pip uninstall paket_nomi deb yozamiz.

Ushbu darsimizning maqsadi tashqi paketlar va modullar bilan tanishish orqali Pythonning naqadar keng gamrovli til ekanini ko'rsatishdir. Aslida, har bir modul va uning imkoniyatlari haqida soatlab gapirish mumkin, lekin biz vaqtini tejash maqsadida turli yo'nalishlardagi ba'zi modullar bilan tanishamiz va ularning ishlashiga soddagina misollar ko'rish bilan chegaralanamiz.

googletrans

O'rnatilishi: `pip install googletrans`

Ushbu modul yordamida Googlening tarjimonlik xizmatiga murojaat etib, istalgan matni turli tillarga tarjima qilishimiz mumkin. Modulan foydalanish uchun avval googletrans modulidan Translator klassini import qilamiz va bu klassdan yangi obyekt yaratamiz (tarjimon). Bevosita tarjimonlik xizmatiga murojaat etish uchun tarjimon obyekt ichidagi .translate() metodiga murojaat etamiz va parametr sifatida tarjima qilish kerak bo'lgan matni uzatamiz. .translate() metodi tarjimoni obyekt ko'rinishida qaytaradi, obyekt ichidan tarjima matnini olish .text xususiyatiga murojaat etamiz.

```
from googletrans import Translator
tarjimon = Translator() # Translator - bu maxsus klass
(tarjimon esa obyekt)
matn_uz = "Python - dunyodagi eng mashhur dasturlash tili"
tarjima = tarjimon.translate(matn_uz)
print(tarjima.text)
```

① googletrans modulini ishlatish uchun kompyuterimiz internet tarmog'iga ulangan bo'lishi kerak.

Natija:

Python is the most popular programming language in the world

△ Agar yuqoridagi kod xato bersa (TKK token error), googletrans modulini o'chirib tashlab, modulning alfa versiyasini o'rnatish:

```
pip uninstall googletrans # modulni o'chirish
pip install googletrans==3.1.0a0 # yangi versiyani o'rnatish
```

Agar matni boshqa tillarga tarjima qilish kerak bo'lsa, .translate() metodiga matnga qo'shimcha ravishda dest parametrini ham uzatamiz va bu parametrga tarjima qilinishi

kerak bo'lgan tilning qisqartmasini beramiz. Masalan, rus tiliga tarjima qilish uchun dest='ru' deb yozamiz.

① Tarjima uchun mavjud tillarni quyidagi manzilda ko'rishingiz mumkin: https://sites.google.com/site/opti365/translate_codes

```
tarjima_ru = tarjimon.translate(matn_uz, dest='ru')
print(tarjima_ru.text)
```

Natija:

Python - самый популярный язык программирования в мире

Ingliz tilidan boshqa tillarga tarjima ham shunday:

```
matn_en = "Tashkent is the capital of Uzbekistan"
tarjima_uz = tarjimon.translate(matn_en, dest='uz')
print(tarjima_uz.text)
```

Natija:

Toshkent - O'zbekistonning poytaxti

Odatda, Google asl matnning tilini o'zi aniqlaydi. Lekin matn tilini ham alohida ko'rsatmoqchi bo'lsangiz, src parametridan foydalaning:

```
tarjima_uz = tarjimon.translate(matn_en, src='uz', dest='en')
```

requests

O'rnatilishi: `pip install requests`

Bu paket yordamida Pythonda veb sahifalarga murojaat etishimiz (so'rov yuborishimiz) va ulardan qaytgan ma'lumotlar ustida turli amallar bajarishimiz mumkin. Misol uchun, quyida requests yordamida kun.uz sahifasini to'liqligicha tortib olamiz. Natija juda ham uzun bo'lgani uchun bu yerda keltirmadik. Kodni o'zingiz bajarib ko'rishingiz mumkin.


```
import requests
from pprint import pprint

manzil = "https://kun.uz/news/main"
r = requests.get(manzil)
pprint(r.text)
```

Ko'pincha requests paketidan API-lar bilan ishlashda foydalaniladi. API ma'lum bir veb xizmatga so'rov yuborish orqali undan foydalanish. Misol uchun, Yandex tarjimonga yoki Google xaritalari xizmatiga requests paketi yordamida API so'rov yuborish va o'zimizga kerakli ma'lumotlarni olishimiz mumkin. API haqida internetda ko'proq ma'lumot olishingiz mumkin. Biz esa soddagina misol bilan cheklanamiz.

Internetda `restcountries.eu` sahifasi mavjud. Bu sahifa orqali dunyodagi davlatlar haqida turli ma'lumotlarni olishingiz mumkin. Sahifadan foydalanish qulay bo'lishi uchun esa sahifa yaratuvchilari bir nechta tayyor API-lar e'lon qilishgan. Misol uchun, O'zbekiston haqida ma'lumot olish uchun quyidagi manzilga so'rov yuborasiz: `https://restcountries.eu/rest/v2/name/Uzbekistan`

API dan qaytgan natija JSON (lug'at) ko'rinishda bo'ladi va biz bu lug'atdan o'zimizga kerakli ma'lumotni sug'urib olishimiz mumkin. Misol uchun, quyidagi kodimiz APIga yuborilgan davlatning poytaxtini ko'rsatadi:

```
country = "Uzbekistan"
url = f"https://restcountries.eu/rest/v2/name/{country}"
r = requests.get(url)
r_json = r.json()[0]
print(r_json['capital'])
```

Natija:

Tashkent

BeautifulSoup4

O'rnatilishi: `pip install beautifulsoup4`

BeautifulSoup juda kuchli modullardan biri bo'lib, bu modul yordamida turli veb-sahifalardan istalgan ma'lumotlarni yig'ishtirib (scrapping) olish mumkin. Biror kishining Instagram sahifasidagi barcha rasmlar deysizmi, Facebook guruhidagi barcha postlar va izohlar deysizmi, oldi-sotdi bozoridagi e'lonlar deysizmi, marhamat, bs4 moduli yordamida buni bimalol avtomatlashtrish mumkin.

Odatda, bs4 moduli requests moduli bilan hamkorlikda ishlaydi. Keling, soddagina misol keltiramiz. Avvalgi bo'linda requests yordamida `kun.uz` sahifasini tortib olgan edik. Endi esa bs4 yordamida sahifadan oxirgi yangiliklarning mavzusini ajratib olamiz.

```
import requests
from bs4 import BeautifulSoup

sahifa = "https://kun.uz/news/main"
r = requests.get(sahifa)

soup = BeautifulSoup(r.text, 'html.parser')
# yangiliklarning mavzusini ajratib olamiz
news = soup.find_all(class_="news-title")
# eng birinchi yangilikni konsolga chiqaramiz
print(news[0].text)
```

Natija:

Коронавирус: АстраЗенеца гумон остида, Европада вазионт
о'йриналишмоқда

wordcloud va matplotlib

O'rnatilishi:

```
pip install wordcloud
pip install matplotlib
```



```
import requests
from pprint import pprint

manzil = "https://kun.uz/news/main"
r = requests.get(manzil)
pprint(r.text)
```

Ko'pincha requests paketidan API-lar bilan ishlashda foydalaniladi. API ma'lum bir veb xizmatga so'rov yuborish orqali undan foydalanish. Misol uchun, Yandex tarjimonga yoki Google xaritalari xizmatiga requests paketi yordamida API so'rov yuborish va o'zimizga kerakli ma'lumotlarni olishimiz mumkin. API haqida internetda ko'proq ma'lumot olishingiz mumkin. Biz esa soddagina misol bilan cheklanamiz.

Internetda `restcountries.eu` sahifasi mavjud. Bu sahifa orqali dunyodagi davlatlar haqida turli ma'lumotlarni olishingiz mumkin. Sahifadan foydalanish qulay bo'lishi uchun esa sahifa yaratuvchilari bir nechta tayyor API-lar e'lon qilishgan. Misol uchun, O'zbekiston haqida ma'lumot olish uchun quyidagi manzilga so'rov yuborasiz: `https://restcountries.eu/rest/v2/name/Uzbekistan`

API dan qaytgan natija JSON (lug'at) ko'rinishda bo'ladi va biz bu lug'atdan o'zimizga kerakli ma'lumotni sug'urib olishimiz mumkin. Misol uchun, quyidagi kodimiz APIga yuborilgan davlatning poytaxtini ko'rsatadi:

```
country = "Uzbekistan"
url = f"https://restcountries.eu/rest/v2/name/{country}"
r = requests.get(url)
r_json = r.json()[0]
print(r_json['capital'])
```

Natija:

Tashkent

BeautifulSoup4

O'rnatilishi: `pip install beautifulsoup4`

BeautifulSoup juda kuchli modullardan biri bo'lib, bu modul yordamida turli veb-sahifalardan istalgan ma'lumotlarni yig'ishtirib (scrapping) olish mumkin. Biron kishining Instagram sahifasidagi barcha rasmlar deysizmi, Facebook guruhidagi barcha postlar va izohlar deysizmi, oldi-sotdi bozordagi e'lonlar deysizmi, marhamat, bs4 moduli yordamida buni bimalol avtomatlashtirish mumkin.

Odatda, bs4 moduli requests moduli bilan hamkorlikda ishlaydi. Keling, soddagina misol keltiramiz. Avvalgi bo'limda requests yordamida `kun.uz` sahifasini tortib olgan edik. Endi esa bs4 yordamida sahifadan oxirgi yangiliklarning mavzusini ajratib olamiz.

```
import requests
from bs4 import BeautifulSoup

sahifa = "https://kun.uz/news/main"
r = requests.get(sahifa)

soup = BeautifulSoup(r.text, 'html.parser')
# yangiliklarning mavzusini ajratib olamiz
news = soup.find_all(class_="news-title")
# eng birinchi yangilikni konsolga chiqaramiz
print(news[0].text)
```

Natija:

Коронавирус: Астразенеца гумон остида, Европада вазиат оғирлашмоқда

wordcloud va matplotlib

O'rnatilishi:

```
pip install wordcloud
pip install matplotlib
```



```

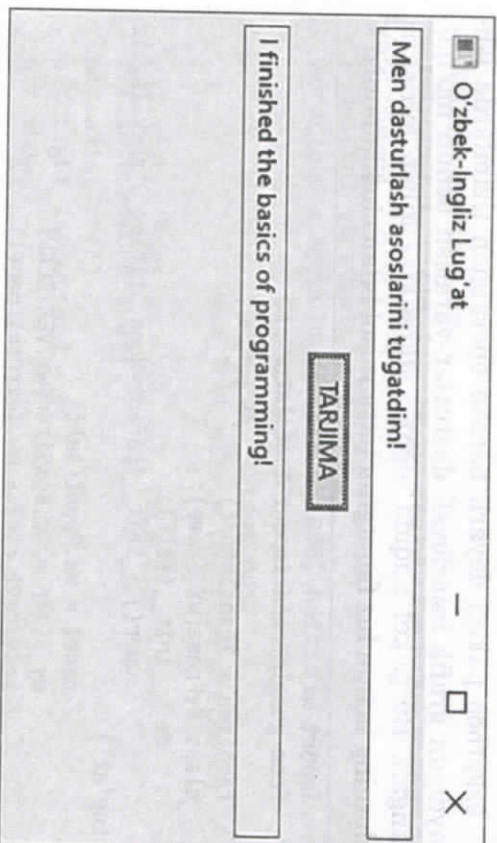
my_btn = wx.Button(panel, label='TARJIMA')
my_btn.Bind(wx.EVT_BUTTON, self.on_press)
my_sizer.Add(my_btn, 0, wx.ALL | wx.CENTER, 5)

self.text_out=wx.TextCtrl(panel,style=wx.TE_READONLY)
my_sizer.Add(self.text_out, 0, wx.ALL | wx.EXPAND, 5)
panel.SetSizer(my_sizer)
self.Show()

def on_press(self, event):
    value = self.text_ctrl.GetValue()
    if not value:
        self.text_out.SetValue("So'z kiritmadingiz")
    else:
        tarjima = tarjimon.translate(value, src='uz')
        self.text_out.SetValue(tarjima.text.capitalize())

if __name__ == '__main__':
    app = wx.App()
    frame = MyFrame()
    app.MainLoop()

```



39-RASM. wxPython asosida yaratilgan sodda dastur

opencv

O'rnatish: pip install opencv-python

opencv – kompyuter yordamida rasm va video tasvirlar bilan ishlash uchun maxsus kutubxona. Bugungi kunda sun'iy intellekt yordamida tasvirlar bilan ishlaydigan dasturlarning deyarli barchasi opencv yordamida yaratiladi.

Bu dastur yordamida rasm va videolardagi turli obyektlarni "ko'rish", ajratib olish mumkin. Avtomobillar raqamini aniqlash, odamlarni yuzidan tanish, obyektlarni klassifikatsiya qilish kabi dasturlarning aksari aynan opencv kutubxonasi yordamida ishlaydi.

Bu kutubxona haqida @sariqdev kanalimizda Sun'iy Intel-лектga bag'ishlangan darslarda batafsil to'xtalamiz. Hozircha esa quyidagi dastur yordamida videodagi odamlarning yuzi va ko'zini ajratishga misolni ko'zdan kechiramiz.

```

import cv2

cap = cv2.VideoCapture(0)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades
+ 'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades
+ 'haarcascade_eye.xml')

while True:
    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 5)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray, 1.3, 5)
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey
+ eh), (0, 255, 0), 5)

```

```

cv2.imshow('frame', frame)

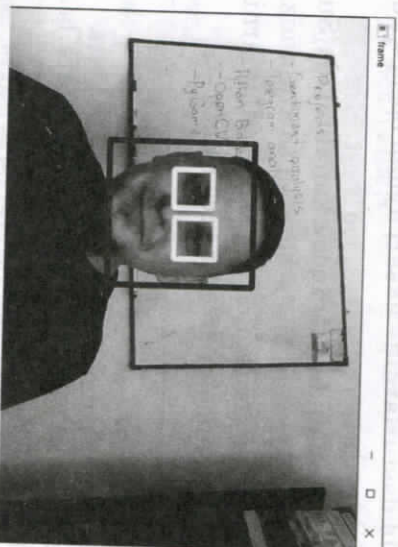
if cv2.waitKey(1) == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

# copyright Tim Ruscia aka techwithtim
# code from https://github.com/techwithtim/OpenCV-Tutorials

```

Natija:



40-RASM. openCV yordamida videoda yuz va ko'zni aniqlash

PyWebIO

O'rnatilishi: pip install pywebio

Ushbu modul veb-dasturlashdan mutlaqo xabari yo'q odam-larga ham soddaveb-dasturlar yaratish imkoniyatini beradi. pywebio yordamida dasturlashga yangi kirganlar ham konsol dasturlarini brauzerda chiroyli ko'rinishda taqdim etishlari mumkin.

Keling, quyidagi soddamisolni ko'ramiz. Bu dastur doira-niqradiusidan uning yuzi va perimetrini hisoblab qayta-radi.

```

from pywebio.input import input, FLOAT
from pywebio.output import put_text
from math import pi

def doira():
    r = input("Doira radiusini kiriting:", type=FLOAT)
    yuza = pi*(r**2)
    per = 2*pi*r
    put_text(f"Doira yuzi {yuza}ga, \nperimetri {per}ga teng")
doira()

```

Dasturni bajariganda pywebio tarkibidagi input() funksiya-si kompyuteringizda veb-brauzer oynasini ochadi va qiymat kiritishni so'raydi:

Doira radiusini kiriting:

Submit Reset

41-RASM. pyWebIO input() funksiyasi

Kerakli qiymatni kiritganimizdan so'ng put_text() funksiyasi natijani brauzer oynasida ko'rsatadi:

Doira yuzi 50.26548245743669ga
parametri 25.132741228718345 ga teng

42-RASM. pyWebIO put_text() funksiyasi

Ko'rib turganingizdek, pywebio moduli yordamida soddadasturlarimizni ham chiroyli taqdim etishimiz mumkin ekan. Keyingi qismda yuqoridagi modullardan foydalangan holda bir nechta loyihalar ustida ishlaymiz.

△ Vaqt o'tishi bilan kutubxona ichidagi modullar yangilanib turishi, turli funksiya va obyektlarning ishlashida o'zgarishlar yuz berishi tabiiy. Agar kitobimizda keltirilgan modullar siz kutingandek ishlamasa, har bir modulning sahifasidagi qo'llanmaga murojaat eting.

2-QISM

LOYIHALAR

Tabriklaymiz! Siz endi qisqa va mazmunli loyihalar ustida ishlash uchun yetarli bilimga egasiz. O'z loyihalaringiz ustida ishlash sizda yangi ko'nikmalar hosil qiladi va 1-qismda o'rgangan bilimlaringizni yanada mustahkamlaydi.

Ikkinchi qismda biz bir nechta amaliy loyihalar ustida ishlaymiz, yangi dasturlar yaratamiz. 13-bobda "Son topish" va "So'z topish" o'yinlarini yaratamiz. O'yinlar Python konsoli uchun mo'ljallangan bo'lsa-da, pywebio moduli yordamida konsol o'yinlarini veb-brazuerda chiqarishni o'rganamiz.

14-bobda onlayn anketani Word faylga o'tkazib beruvchi veb-dastur yaratamiz. Frontend va backend tushunchalari bilan tanishamiz. Fayllar bo'yicha bilimlaringizni mustahkamlaymiz.

15-bobda esa Telegram messengeri uchun O'zbek-Ingliz-O'zbek tarjimon botini yaratamiz.

2-qismini yakunlagandan so'ng sizda loyiha ustida ishlash va o'z dasturlaringizni yaratish haqida tushuncha hosil bo'ladi. Bu esa, o'z navbatida, kelajakdagi katta rejalaringiz uchun muhim poydevor bo'ladi.

13-BOB

"SON TOPISH" VA "SO'Z TOPISH" O'YINLARI

Ushbu bobda biz ikkita yangi o'yin ustida ishlaymiz. Dasturni yaratish jara-yonida avval o'rgangan bilimlarimizni amalda qo'llaymiz. Shuningdek, yangi lo-yiha boshlashda muhim bo'lgan algoritm tushunchasi bilan tanishamiz.



<https://olico/Da-sontopi>

13.1. "SON TOPISH" O'YINI

Birinchi dasturimiz sodd "Son topish" o'yini bo'ladi. Bu o'yin quyidagicha o'ynaladi:

1. Kompyuter biror berilgan oraligda (masalan, 1 dan 10 gacha) son "o'ylaydi".
2. Foydalanuvchi kompyuter o'ylagan sonni topishga urinadi va biror qiymat taxmin qilib kiritadi.
3. Agar taxmin to'g'ri bo'lsa, o'yin tugaydi, agar xato bo'lsa, kompyuter foydalanuvchi taxmin qilgan son o'ylangan sonidan katta yoki kichikligini aytadi va yangi son taxmin qilishni so'raydi.
4. Dastur shu zaylda foydalanuvchi to'g'ri sonni topguniga qadar davom etadi.

ALGORITM

Dasturlashning muhim qismi to'g'ri reja qilish va dasturni mayda qadamlarga bo'lib olishdir. Bu Algoritm deyiladi. Xud-

di ovqat pishirishda retseptdan foydalangandek, dastur yozishda ham Algoritm dan foydalanamiz.

Yuqoridagi o'yin qoidasini ham bir nechta sodd qadamlarga bo'lib olamiz:

1. Kompyuter ma'lum oraligda son "o'ylaydi". Albatta, kompyuter o'ylashga qodir emas, shuning uchun biz berilgan oraligda biror tasodifiy sonni qaytarishimiz kifoya (random).
2. Foydalanuvchi taxmin qilgan sonni qabul qilib olish (input()).
3. Taxmin va tasodifiy sonlarni taqqoslash (==).
4. Taqqoslash natijasiga ko'ra (if-else):
 - a. yo o'yinni to'xtatish;
 - b. yo foydalanuvchiga ishora berish va qayta taxmin qilishni so'rash va 3 - 4-qadamlarni takrorlash (while).

Sodd algoritm tayyor, endi bevosita dastur yozishga kirish-sak ham bo'ladi.

DASTUR

Boshlanishiga dastur uchun kompyuterimizda alohida papka yaratamiz va dasturga oid fayllarni (modullarni) shu papkada saqlab boramiz. Dastur yozishda asosiy dasturimizni main.py, qo'shimcha modullarni esa shu modul ichidagi funksiyaga oid so'zlar bilan nomlash to'g'ri bo'ladi.

Biz ham boshlanishiga sonTop.py faylini yaratamiz va fayl ichiga yuqoridagi algoritm asosida son-

top(x) funksiyasini yozamiz. Bu funksiyamiz argument sifatida kompyuter o'ylashi kerak bo'lgan sonlar oralig'ining yuqori chegarasini qabul qiladi (0 dan x gacha) va biz bu parametrga standart 10 qiymatini berib qo'yamiz (ya'ni foydalanuvchi aniq oraligni ko'rsatmasa, kompyuter 0 dan 10 gacha son "o'ylaydi").



<https://olico/Da-sontop2>

Funksiya ichida birinchi qiladigan ishimiz – tasodifiy son "o'ylash". Buning uchun random modulidagi randint funksiyasidan foydalanamiz. Demak, dastur boshida random modulini import qilishni ham esdan chiqarmaslik kerak:

```
import random
def sontop(x=10):
    tasodifiy_son = random.randint(1,x)
```

Navbat foydalanuvchidan qiymatlar qabul qilishga. Keling, bundan avval kompyuter aynan qaysi oraliqda son o'ylaganini ham aytaylik.

```
def sontop(x=10):
    tasodifiy_son = random.randint(1,x)
    print(f"Men 1 dan {x} gacha son o'yladim. Topa olasizmi?", end="")
```

Ana endi bevosita input() funksiyasi yordamida foydalanuvchi kiritgan sonni qabul qilamiz va bu sonni tasodifiy son bilan solishtiramiz.

Algoritimga ko'ra, biz foydalanuvchining taxminiga qarab turli amallarni bajarishimiz kerak:

- Agar taxmin tasodifiy son dan kichik bo'lsa, "Kattaroq son ayting";
- Agar taxmin tasodifiy son dan katta bo'lsa, "Kichikroq son ayting";
- Agar taxmin to'g'ri bo'lsa, "Yutdingiz!" deb aytishimiz kerak.

O'yin foydalanuvchi yutguniga qadar davom etishi uchun kodimizni while sikli ichiga olamiz.

Marhamat, funksiya tayyor.

```
import random

def sontop(x=10):
    tasodifiy_son = random.randint(1,x)
    print(f"Men 1 dan {x} gacha son o'yladim. Topa olasizmi?", end="")
    while True:
        taxmin = int(input(">>>"))
        if taxmin < tasodifiy_son:
            print("Kattaroq son ayting:", end="")
        elif taxmin > tasodifiy_son:
            print("Kichikroq son ayting:", end="")
        else:
            print("Yutdingiz!")
            break
```

Biz funksiyani alohida modulda saqladik, keling main.py faylini yaratamiz-da, yuqoridagi modul va funksiyani import qilamiz:

```
from sontop import sontop
sontop()
```

Faylimiz tarkibi qisqagina bo'ldi. main.py faylini bajaramiz

```
Men 1 dan 10 gacha son o'yladim. Topa olasizmi?
>>>8
Kichikroq son ayting:
>>>4
Kattaroq son ayting:
>>>6
Kichikroq son ayting:
>>>5
Yutdingiz!
```

pyWebIO

Avvalgi bobda pywebio moduli bilan tanishgan edik, keling, ushbu modul yordamida dasturimizni brauzerda ishlaydigan qilamiz.

Boshlanishiga modul ichidan input va put_text funksiyalarini chaqirib olamiz. input() funksiyasi bevosita qiymat qabul qilish uchun, put_text() esa foydalanuvchiga matn ko'rsatish uchun ishlatiladi.

Avvalgi dasturda "Men 1 dan 10 gacha son o'yladim. Topa olasizmi?>>>" matnini ikkiga bo'lgan edik, bu safar ikkalasini ham input() funksiyasiga joylaymiz. Konsolga chiqarilayotgan matnlarni esa brauzerga chiqarish uchun kodimizdagi print funksiyasini put_text() bilan almashiramiz:

```
from pywebio.input import input
from pywebio.output import put_text
import random

def sontop(x=10):
    tasodifiy_son = random.randint(1,x)
    while True:
        taxmin = int(input(f"Men 1 dan {x} gacha son
o'yladim. Topa olasizmi?"))
        if taxmin<tasodifiy_son:
            put_text("Kattaroq son ayting!")
        elif taxmin>tasodifiy_son:
            put_text("Kichikroq son ayting!")
        else:
            put_text("Yutdingiz!")
            break
```

Asosiy dasturimizga (main.py) qaytamiz va dasturni bajarimiz:

Men 1 dan 10 gacha son o'yladim. Topa olasizmi?

Submit Reset

43-RASM. "Son top" o'yini

pywebio tarkibida foydalanuvchiga matn ko'rsatishning turli yo'llari bor, masalan, popup() funksiyasi yordamida "Yutdingiz!" matnini alohida oynada ochib ko'rsatishimiz ham

mumkin. Buning uchun kodimizning boshida popup funksiyasini import qilamiz va dastur so'ngidagi put_text("Yutdingiz!") gatorini popup("Yutdingiz!") bilan almashiramiz:

```
from pywebio.input import input
from pywebio.output import put_text, popup
import random

def sontop(x=10):
    ***
    if taxmin<tasodifiy_son:
        ***
    else:
        popup("Yutdingiz!")
        break
sontop()
```

Dasturni bajarib ko'ramiz:

Yutdingiz!

x

44-RASM. pywebio popup funksiyasi

DASTURNI TERMINALDAN BAJARISH

Biz shu vaqtgacha yaratgan dasturlarimizni Spyder muhiti orqali bajarayotgan edik. Dastur yaratish jarayonida bu tez va qulay. Lekin dastur bitganidan so'ng uni Python orqali bajarish to'g'ri bo'ladi.

Buning uchun Windows Command Prompt dasturini ochamiz (Linux va MacOS da Terminal). Yangi ochilgan oynada cd komandasini va ortidan main.py faylimiz joylashgan papkaga yo'lni kiritamiz. Misol uchun, mening dasturim C:\python-darslar\son-top-game papkasida joylashgan. Demak, men Command Prompt oynasida quyidagi buyruqni yozaman:

cd C:\python-darslar\son-top-game

Agar hammasi to'g'ri bo'lsa, dasturimiz joylashgan papkaga kiramiz. Agar The system cannot find the path specified degan xato chiqsa, papka nomlari to'g'ri yozilganini tekshirib, yuqoridagi buyruqni qayta yozing.

Papkaga kirgandan so'ng dasturni ishga tushirish uchun quyidagi buyruqni yozing (menda fayl nomi main.py):

python main.py

2-bobdagi qo'llanmaga ko'ra Python tilini to'g'ri o'rnatgan bo'lsangiz, dastur bajarila boshlaydi.

KOMPYUTER BIZGA QARSHI

Avvalgi bo'limda biz kompyuterga qarshi o'ynadik. Keling, endi buning aksini, biz son o'ylaymiz, kompyuter topsin. Bu safar dasturimizni to'liqligicha pywebio yordamida yozamiz. Dastavval o'yin qoidasini kelishib olaylik:

1. Kompyuter bizga son o'ylash uchun 3 soniya vaqt beradi.
 2. Biz berilgan oralikda son o'ylaymiz.
 3. Kompyuter "taxmin" qiladi va bizdan taxmin to'g'riligini so'raydi.
 4. Biz "Kattaroq", "Kichikroq" yoki "To'g'ri" deb javob qaytaramiz.
 5. Agar javob to'g'ri bo'lsa, o'yin tugaydi.
 6. Agar javob noto'g'ri bo'lsa, kompyuter taxmin qilishda davom etadi.
 7. O'yin to'g'ri kompyuter yutgunga qadar davom etadi.
- O'yin qoidasi tushunarli bo'ldi deb o'ylayman. Navbat al-goritimga.
1. O'yin boshlanishida foydalanuvchiga 3 soniya (time) o'ylash uchun vaqt beramiz (put_text).
 2. Foydalanuvchi biror oralikda son o'ylaydi (oraligini ham avvaldan kelishib olamiz).

3. Kompyuter "taxmin" qiladi. Buning uchun yana o'sha randint funksiyasiga murojaat etamiz.

4. Kompyuter taxmin to'g'ri yoki noto'g'ri ekanini so'raydi. Foydalanuvchi "To'g'ri", "Kattaroq" yoki "Kichikroq" deb javob qaytaradi. Buning uchun pywebio modulidagi put_buttons() funksiyasi yordamida ekranga tugmalar chiqaramiz.

a. Agar foydalanuvchi "Kattaroq" tugmasini bosgan bo'lsa, kompyuter taxmin qilishi kerak bo'lgan sonlar oralig'ining pastki chegarasini foydalanuvchi kiritgan songa o'zgartiramiz (chunki endi kompyuter kattaroq son o'ylashi kerak) va 3-qadamga qaytamiz.

b. Agar foydalanuvchi "Kichikroq" tugmasini bossa, aksincha, yuqori chegarani foydalanuvchi kiritgan songa tenglaymiz va 3-qadamga qaytamiz.

c. Agar taxmin to'g'ri bo'lsa, o'yinni tugatamiz.

Navbat dasturga.

sontop.py fayliga qaytamiz va avvalgi modullarga qo'shimcha sifatida foydalanuvchiga 3 soniya vaqt berish uchun time modulidagi sleep() va brauzerga tugmalar qo'shish uchun put_buttons funksiyalarini chaqiramiz.

```
from pywebio.output import put_text, put_buttons
import pywebio
from time import sleep
import random
```

Bu safar yangi funksiyamizga sontop_pc() deb nom beramiz. Dastur boshida brauzer ekranini tozalash uchun pywebio ichidagi clear funksiyasiga murojaat etamiz.

Men dasturni bir nechta maydaroq funksiyalarga ajratishni afzal topdim. Jumladan, guess() funksiyasi faqatgina son taxmin qiladi va foydalanuvchidan taxmin to'g'ri yoki noto'g'riligini so'raydi. Bu yerda pywebio modulidagi put_buttons modulidan foydalanamiz va brauzerda 3 ta tugmani ko'rsatamiz.

Foydalanuvchi qaysi tugmani bosganiga qarab katta(), kichik() va bingo() funksiyalaridan biri chaqiriladi. katta() va kichik() funksiyalari yuqori yoki quyi chegaralarni o'zgartiradi, bingo() funksiyasi esa kompyuter to'g'ri taxmin qilgani haqida xabarni chiqaradi.

Dasturimizdagi global kalit so'ziga e'tibor bering. Bu operator o'zidan keyin kelgan o'zgaruvchilarning global o'zgaruvchi ekanini anglatadi. Esingizda bo'lsa, funksiyalar to'g'ri-da gaplashganimizda funksiya ichidagi o'zgaruvchilar local, mahalliy, ya'ni funksiya tashqarisidagi o'zgaruvchilardan alohida bo'ladi degan edik. global operatori yordamida esa o'zgaruvchining mahalliy emasligiga ishora qilyapmiz. Bunday o'zgaruvchilarning qiymatini istalgan joydan o'zgartirishimiz mumkin bo'ladi.

Mana, endi kodimizni to'liqligicha keltiramiz:

```
from pywebio.output import put_text, put_buttons
import pywebio
from time import sleep
import random

def sontop_pc(x=10):
    pywebio.output.clear(scope=-1)
    global quyi, yuqori, taxmin
    quyi=1
    yuqori=x
    put_text(f"1 dan {x} gacha son o'ylang.\nSizga 3 soniya vaqt!")
    for i in [3,2,1,'Boshladik!']:
        put_text(str(i))
        sleep(1)
    def guess():
        pywebio.output.clear(scope=-1)
        global taxmin
        taxmin = random.randint(quyi,yuqori)
```

```
put_text(f"Siz {taxmin} sonini o'yladingiz!")
put_buttons(['Kattaroq', 'Tog'ri', 'Kichikroq'],
onclick=[katta,bingo,kichik])
pywebio.session.hold()
```

```
def kichik():
    global yuqori
    yuqori = taxmin-1
    guess()

def katta():
    global quyi
    quyi = taxmin+1
    guess()

def bingo():
    put_text("Men yutdim!")
    guess()
```

main.py faylga qaytamiz, yangi funksiyamizni chaqiramiz va dasturni bajarib ko'ramiz.

```
from sontop import sontop, sontop_pc
sontop()
sontop_pc()
```

AMALIYOT

- Yuqorida biz ikki o'yin uchun alohida funksiyalar yozdik: sontop() va sontop_pc().
- Sizning vazifangiz mana shu ikki funksiyani obyektga yo'naltirilgan dasturlash tamoyillari asosida qayta yozish-dir (funksiya klassga o'zgartirasiz).
 - O'zgaruvchilar xususiyatlarga, ichki funksiyalar esa metodga o'zgaradi.

13.2. "SO'Z TOPISH" O'YINI

Navbatdagi dasturimiz bilan "So'z topish" o'yini yaratamiz. Bu dasturda foydalanuvchi kompyuterga qarshi o'ynaydi.

O'yin qoidasi juda oddiy:

- Kompyuter biror so'z "o'ylaydi".
- Foydalanuvchi kompyuter o'ylagan so'zning tarkibidagi biror harfni taxmin qiladi.
- O'yin to hamma harflar topilgunga qadar davom etadi.



<https://olico/Da-word1>

ALGORITM

Kompyuter so'z "o'ylashi" uchun bizga o'zbek tilidagi so'zlar ro'yxati zarur. Kompyuter esa ro'yxatdan tasodifiy so'zni tanlaydi. O'zbek tilidagi so'zlar ro'yxatini quyidagi bog'lama orqali yuklab olishingiz mumkin: <https://github.com/anvarnarz/uzwords>

Bevosita algoritnga o'tamiz.

1. So'zlar ro'yxatidan tasodifiy so'z tanlash. Bu yerda biz random modulidagi `choice()` funksiyasidan foydalanamiz. Ro'yxatimizda tireli (-) yoki bo'sh joyli so'zlar ham mavjud, demak, so'z tanlashda shunday so'zlar tushib qolmasligiga ahamiyat beramiz.

2. So'z tarkibidagi harflar to'plamini tuzamiz (`set()`).

3. Foydalanuvchiga tanlangan so'z nechta harfdan iborat ekanini bildiramiz. Harflar o'rnini pastki chiziqcha bilan ko'rsatamiz.

4. Bevosita o'yinni boshlaymiz. Foydalanuvchidan harf kiritishni so'raymiz va kiritgan harf so'zda bor-yo'qligiga qarab turli yo'l tutamiz:

a. Agar harf so'zda bo'lsa, shu harfni ochamiz va so'z tarkibidagi harflar ro'yxatidan topilgan harfni o'chiramiz.

b. Agar foydalanuvchi hamma harflarni topgan bo'lsa, o'yinni tugatamiz.

c. Agar harf so'zda bo'lmasa, foydalanuvchidan yangi harf kiritishni so'raymiz.

DASTUR

Avvalgidek yangi dastur uchun yangi papka yaratamiz. Men dasturni uch qismga bo'ldim:

1. `get_word()` funksiyasi `uzwords` ro'yxatidan tasodifiy so'z tanlaydi.
2. `display()` funksiyasi foydalanuvchi topgan so'zlar va tasodifiy so'zni qabul qiladi va topilgan harflar ro'yxatini qaytaradi.
3. `play()` bevosita o'yin jarayoni.

Dasturimizni `uzwords` modulidagi `words` ro'yxatini import qilishdan boshlaymiz. Shuningdek, ro'yxatdan tasodifiy so'z tanlash uchun `random` moduli ham kerak bo'ladi.



<https://olico/Da-word2>

```
import random
from uzwords import words
```

Navbat `get_word()` funksiyasiga. Yuqorida aytganimizdek, agar tanlagan so'zimiz tarkibida tire (-) yoki bo'sh joy bo'lsa, ro'yxatdan boshqa tasodifiy so'zni olamiz. Buni `while` sikli yordamida amalga oshiramiz. Tanlangan so'zni bosh harflar bilan qaytaramiz:

```
def get_word():
    word = random.choice(words)
    while "-" in word or ' ' in word:
        word = random.choice(words)
    return word.upper()
```

Funksiyani tekshirib ko'ramiz:

```
>>> get_word()
'ЮЗЛИК'
>>> get_word()
'Кўча'
>>> get_word()
'Йўлланма'
```

Endi display() funksiyasiga o'tamiz. Bu funksiya foydalanuvchi to'rtta harflar va tasodifiy so'zni qabul qiladi, so'z tarqibidagi topilgan harflarni o'z o'rniga qo'yib, topilmagan harflar o'rnini esa tire (-) bilan almashtirib yangi matn qaytaradi:

```
def display(user_letters, word):
    display_letter=""
    for letter in word:
        if letter in user_letters:
            display_letter += letter
        else:
            display_letter += "-"
    return display_letter
```

Bu funksiyani ham tekshirib ko'ramiz:

```
>>> harflar = 'АКЙВМЭ'
>>> word = 'Йўлланма'
>>> display(harflar, word)
'Й--А-М' #<- 'Йўлланма' so'zida topilgan harflar
```

Navbat play() funksiyasiga. Bu funksiyamiz asosiy o'yin jarayonini boshqaruvchi funksiya bo'ladi. Funksiyani get_word() yordamida tasodifiy so'z tanlashdan boshlaymiz.

```
def play():
    word = get_word()
```

Keyingi qadam so'zdaagi harflar to'rtlamini shakllantirish:

```
word_letters = set(word)
```

Yodingizda bo'lsa, set() funksiyasi berilgan to'xtatdan (matndan) takrorlanuvchi elementlarni tashlab yuborib, yangi to'rtlam shakllantiradi.

O'yin davomida foydalanuvchi kiritgan harflarni saqlab borish uchun ham bitta bo'sh matn tuzib olamiz.

```
user_letters = ''
```

Foydalanuvchiga o'yin boshlanganini va tanlangan so'z nechta harfdan iborat ekanini ko'rsatamiz:

```
print(f"Men {len(word)} xonali suz uyladim. Topa olasizmi?")
```

O'yinimiz toki (while) foydalanuvchi barcha harflarni topguncha davom etishi kerak, buning uchun yuqoridagi word_letters() to'rtlamidan foydalanamiz. O'yin boshida to'rtlam harflarga to'la bo'ladi va siklimiz ishga tushadi, vaqt o'tishi bilan foydalanuvchi to'rtta harflar to'rtlamdan o'chiriladi. Ma'lum vaqtdan keyin foydalanuvchi barcha harflarni topadi va to'rtlam bo'shaydi. Bu esa, o'z navbatida, siklimizni ham to'xtatadi, ya'ni o'yin tugaydi.

To'liq funksiyamiz quyidagi ko'rinishga ega bo'ladi:

```
def play():
    word = get_word()
    word_letters = set(word)
    user_letters = ''
    print(f"Men {len(word)} xonali suz uyladim. Topa olasizmi?")
    while word_letters:
        print(display(user_letters, word))
        if user_letters:
            print(f"Shu vaqtgacha kiritgan xarflaringiz: {user_letters}")
        letter = input("Xarf kiriting: ").upper()
        if letter in user_letters:
            print("Bu xarfi avval kiritgansiz.")
            continue
        elif letter in word:
            word_letters.remove(letter)
            print(f"{letter} xarfi tug'di.")
```



```

else:
    print("Бундай харф йўқ.")
    user_letters += letter
    print(f"Табриклайман! {word} сўзини {len(user_letters)} та уринишда топдингиз.")

```

Ушбу коддаги uchta funksiyani alohida funksiyalar.py moduliga saqlaymiz (siz istalgan nom berishingiz mumkin).

Yangi main.py faylini yaratamiz va quyidagi kodni yozamiz:

```

from funksiyalar import play
play()

```

Dasturni bajaramiz:

```

Мен 13 хонали сўз ўйладим. Топа оласми?
-----
Харф киритинг: а
А харфи тўғри.
-А---А-----
Шу вақтгача киритган ҳарфларингиз: А
***
Харф киритинг: о
О харфи тўғри.
-А-ДОЛА-----Ш
Шу вақтгача киритган ҳарфларингиз: АШДЛО
***
Харф киритинг: р
Р харфи тўғри.
Табриклайман! КАНДОЛАТФУРУШ сўзини 11 та уринишда
топдингиз.

```

АМАЛИЙОТ

- Ушбу коддаги dasturda xatolarning oldini olish uchun foydalanuvchidan faqat bitta harf qabul qiling.
- get_word() va display() funksiyalarini tekshiruvchi test dasturlar yozing (unittest).
- Dasturni ruwebio moduli yordamida brauzerda ishlaydigan qilib o'zgartiring.

14-BOB

WEB-DASTUR YARATAMIZ

Navbatdagi dasturimiz foydalanuvchiga turli anketalarni to'ldirishga yordam beradi. Biz misol tariqasida O'zbekiston Respublikasi fuqarosining xorijsa chiqish biometrik pasportini rasmiylashtirish uchun topshiriladigan anketani to'ldiruvchi veb-dastur yozamiz.

Foydalanuvchi brauzer oynasida o'zining shaxsiy ma'lumotlarini kiritadi va dastur yakunida to'ldirilgan anketaning Word faylini yuklab oladi. Juda qulay.

14.1. BACKEND-FRONTEND

Veb-dasturlar ikki qismdan iborat bo'ladi:

1. Frontend – foydalanuvchiga ko'rinadigan va bevosita u bilan muloqot qiladigan qismi.
2. Backend – dasturning orqasida bo'ladigan jaraonlar.

Bu jaraonlar foydalanuvchiga ko'rinmaydi.

① "Front" va "back" so'zlari ingliz tilidan "old" va "orqa" deb tarjima qilinadi.

Biz ham dasturimizni shu qismlarga bo'lamiz. Dasturning frontend qismiga avvalgi bo'limda o'rgangan ruwebio moduli javob beradi. ruwebio yordamida biz foydalanuvchining ma'lumotlarini qabul qilish uchun maxsus sahifa yaratamiz. Foydalanuvchi kerakli ma'lumotlarni kiritganidan so'ng esa ma'lumotlar o'zgaruvchiga saqlanadi va backendga uzatiladi.

Dasturning backend qismida qabul qilingan ma'lumotlar Word faylga yoziladi va foydalanuvchi faylni yuklab olishi uchun qaytib frontendga uzatiladi.

Odarda, katta loyihalarda backend va frontend ustida alohida dasturchilar parallel ishlaydilar. Loyihamiz kichik bo'lgani uchun biz avval dasturimizning orqa qismini tayyorlab olamiz, keyin esa old qismiga o'tamiz.

BACKEND

Dasturimizning ortki qismida bizga dastavval kerak bo'ladigan narsa to'ldirilgan anкета blankasidir. Biz anketani Ichki ishlar vazirligi sahfasidan yuklab oldik, siz esa bizning GitHub sahifamizga murojaat etishingiz mumkin.

Navbat Word fayllar bilan ishlaydigan maxsus modulga. Bu modul python-docx-template deb ataladi. Modulni o'rnatish uchun pip install docxtempl komandasini bajaramiz.

Modul ishlashi juda ham oddiy. Avvalo, yuqorida aytganimizdek, bizga anкета blankasi kerak. Blanka ichida to'ldirishi kerak bo'lgan o'rinlarni kalit so'zlar bilan to'ldirib chiqamiz. Kalit so'zlar takrorlanmasligi va ikki jingalak qavs ichida, qavslardan oldin ikki bo'sh joy tashlab yozilishi lozim. Quyidagi rasmda {{ familiya }}, {{ ism }} va {{ otasi }} kalit so'zlarini ko'rishingiz mumkin. Qolgan o'rinlarni ham shu zaylda kalitlar bilan to'ldiramiz.

Ўзбекистон Республикаси фуқаросининг
Хорিজга чиқиб буюмларни расмийлаштириш учун
АНКЕТА

1. Фамилияси	{{ familiya }}	1
Иsmи	{{ ism }}	2
Отасининг исми	{{ otasi }}	3

45-RASM. Kalit so'zlar bilan to'ldirilgan anкета

Kalit so'zlar bilan to'ldirilgan anketani anкета.docx deb nomlaymiz.

Anketamiz tayyor, navbat bevosita dasturga. Backend uchun alohida backend.py faylini yaratamiz. Fayl ichida, birinchi navbatda, docxtempl moduli ichidan DocxTemplate klassini chaqirib olamiz:

```
from docxtempl import DocxTemplate
```

DocxTemplate klassidan yangi anкета obyektini yaratamiz.

Obyekt yaratishda "anketa.docx" faylimizni argument sifatida uzatamiz.

```
anketa = DocxTemplate("anketa.docx")
```

anketa obyektini ichida maxsus render metodi bor. Bu metod parametr sifatida anкета.docx fayli ichiga yozilishi kerak bo'lgan qiymatlarni lug'at ko'rinishida qabul qiladi va faylmi to'ldiradi. Biz yuqorida kalit so'zlarni yaratib bo'ldik (ism, familiya, otasi), endi shu kalitlarga mos qiymatlarni jamlab lug'at tuzamiz:

```
info = {'ism': 'Alijon',
        'familiya': 'Valiyev',
        'otasi': 'Bogdiyevich'}
```

Lug'atni render metodiga uzatamiz:

```
anketa.render(info)
```

Anкета to'ldirildi. anкета.save() metodi yordamida to'ldirilgan anketani yangi faylga saqlaymiz:

```
anketa.save("yangi_anketa.docx")
```

1. Фамилияси	Valiyev
Иsmи	Alijon
Отасининг исми	Bogdiyevich

46-RASM. To'ldirilgan anкета

Yuqoridagi kodimiz bir martalik dastur bo'ldi, xolos (faqat Alijon Bogdiyevich uchun). Dasturdan unumli foydalanish maq-

sadida uni funksiya holiga keltiramiz. Funksiyamiz parametrlar sifatida frontenddan foydalanuvchi haqidagi ma'lumotlarni lug'at ko'rinishida qabul qiladi. Ikkinchi parametr sifatida anketa faylining nomini beramiz, lekin hozircha biz yagona anketa bilan ishlaganimiz uchun bu parametrga standart qiymat berib qo'yamiz.

Funksiyaga murojaat etganda yaratiladigan yangi fayl nomiga foydalanuvchi familiyasini ham qo'shib ketamiz (anketalar aralashib ketmasligi uchun). Aynan shu faylning nomi funksiyadan qaytib frontendga uzatiladi.

backend.py moduli va ichidagi yakuniy funksiyamiz quyidagicha ko'rinishga ega bo'ladi:

```
from docxtpl import DocxTemplate
def fillDoc(info, fayl="anketa.docx"):
    anketa = DocxTemplate("anketa.docx")
    anketa.render(info)
    filename = f"{info['familiya']}_anketa.docx"
    anketa.save(filename)
    return filename
```

Dasturimizning backend qismi tayyor. Navbat frontendga.

① O'quvchida "Turli modullar qanday ishlashi haqida ma'lumotlarni qayerdan olamiz?" degan o'rinni savol tug'ilishi tabiiy. Odatda, har bir modulning veb-sahifasida mukammal qo'llanma berilgan bo'ladi. Masalan, docxtpl moduli haqida batafsil ma'lumot olish uchun <https://docxtpl.readthedocs.io/> sahifasiga murojaat etishingiz mumkin.

FRONTEND

Dasturimizning old qismi foydalanuvchidan anketada ko'rsatilgan ma'lumotlarni qabul qilishi va backendga uzatishi kerak. Foydalanuvchidan ma'lumot qabul qilish uchun pywebio moduli ichidagi input() funksiyasiga murojaat etamiz. Lekin bu funksiya faqatgina yagona qiymat qabul qiladi, bizga esa bir nechta qiymatlarni qabul qilish zarur. Buning uchun pywebio ichidagi input_group() funksiyasini ham import qila-

miz. Bu funksiya argument sifatida bir nechta input() funksiyalarini ro'yxat ko'rinishida qabul qiladi va barcha kiritilgan ma'lumotlarni lug'at ko'rinishida qaytaradi.

Frontend uchun yangi frontend.py faylini yaratamiz va ichiga quyidagi qatorlarni yozamiz:

```
from pywebio.input import input, input_group

def getInfo():
    info = input_group("Fuqaro haqida ma'lumot", [
        input('Ism', name='ism'),
        input('Familiya', name='familiya'),
        input('Otasining ismi', name='otasi'),
        input("Tug'ilgan yil", name='tyil'),
        input("Tug'ilgan kun", name='tkun'),
        input("Tug'ilgan oy", name='toy'),
        input('Telefon raqami', name='telefon'),
        input('Pasport seriyasi', name='seriya'),
        input('Pasport raqami', name='pass_raqam'),
        input('Pasport berilgan sana', name='pass_sana'),
    ])
    return info
```

getInfo() funksiyamiz to'g'ri ishlashini tekshirib ko'rish uchun konsolda funksiyaga murojaat etamiz:

```
>>> getInfo()
```

Fuqaro haqida ma'lumot

Ism

Familiya

Otasining ismi

Tug'ilgan yil

Tug'ilgan kun

47-RASM. Ma'lumotlarni qabul qilish oynasi tayyor

14.2. ASOSIY DASTUR

Dasturimizning yuzi va orqa qismlari tayyor. Keling, endi bu ikkisini birlashtiramiz. Yangi main.py fayliga yuqoridagi funksiyalarni chaqirib olamiz:

```
from frontend import getInfo
from backend import fillDoc
```

getInfo() funksiyamiz foydalanuvchidan ma'lumotlarni qabul qilib oladi va ma'lumotlar lug'atini qaytaradi. Biz bu lug'atni fillDoc() funksiyasiga uzatamiz va to'ldirilgan anketaning nomini qaytarib olamiz.

```
userInfo = getInfo()
filename = fillDoc(userInfo)
```

Anketa tayyor bo'lganidan so'ng uni foydalanuvchiga yuklab olish imkoniyatini ham berishimiz kerak. Buning uchun pywebio modulidagi maxsus put_file() funksiyasidan foydalanamiz. Ushbu funksiya parameter sifatida fayl nomini va tarkibini (ikkilik sanoq sistemasi ko'rinishida) qabul qiladi va foydalanuvchiga yuklab olish uchun bog'lamani ko'rsatadi. put_file() funksiyasidan foydalanish uchun pywebio.output modulidan funksiyani chaqirib olamiz.

Dasturimiz foydalanuvchi anketani yuklab olishidan avval to'xtab qolmasligi uchun pywebio.session modulidan hold funksiyasini ham chaqiramiz. hold() funksiyasi to'xtatilgan nuqta o'zi brauzer oynasini yopgunga qadar dastur bajarilishini to'xtatmaydi.

main.py modulimiz quyidagi ko'rinishga keladi:

```
from pywebio.output import put_file
from pywebio.session import hold
from frontend import getInfo
from backend import fillDoc
```

```
userInfo = getInfo()
filename = fillDoc(userInfo)
with open(filename, 'rb') as fayl:
    anketa = fayl.read()
    put_file(filename, content=anketa)
    hold()
```

Dasturni ishga tushiramiz:
python c:\python-darslari\41-anketa\main.py

Fuqaro haqida ma'lumot

Ism
Ailjon
Familya
Valiyev
Otasining ismi
Boqiyevich
Tug'ilgan yil
2000

48-RASM. Ma'lumotlarni kiritish oynasi



Valiyev_anketa.docx
49-RASM. Fayl yuklab olish uchun bog'lama

Ўзбекистон Республикаси фузариосининг
хоризга чикни биометрик паспортни расмийлаштириш учун
АНКЕТА

1. Фамилияси	Valiyev	2. Исми	Ailjon	3. Otasining ismi	Boqiyevich	4. Tug'ilgan yili	2000
5. Oila nomi	Valiyev	6. Oila nomi	Valiyev	7. Oila nomi	Valiyev	8. Oila nomi	Valiyev
9. Oila nomi	Valiyev	10. Oila nomi	Valiyev	11. Oila nomi	Valiyev	12. Oila nomi	Valiyev
13. Oila nomi	Valiyev	14. Oila nomi	Valiyev	15. Oila nomi	Valiyev	16. Oila nomi	Valiyev
17. Oila nomi	Valiyev	18. Oila nomi	Valiyev	19. Oila nomi	Valiyev	20. Oila nomi	Valiyev
21. Oila nomi	Valiyev	22. Oila nomi	Valiyev	23. Oila nomi	Valiyev	24. Oila nomi	Valiyev
25. Oila nomi	Valiyev	26. Oila nomi	Valiyev	27. Oila nomi	Valiyev	28. Oila nomi	Valiyev
29. Oila nomi	Valiyev	30. Oila nomi	Valiyev	31. Oila nomi	Valiyev	32. Oila nomi	Valiyev
33. Oila nomi	Valiyev	34. Oila nomi	Valiyev	35. Oila nomi	Valiyev	36. Oila nomi	Valiyev
37. Oila nomi	Valiyev	38. Oila nomi	Valiyev	39. Oila nomi	Valiyev	40. Oila nomi	Valiyev
41. Oila nomi	Valiyev	42. Oila nomi	Valiyev	43. Oila nomi	Valiyev	44. Oila nomi	Valiyev
45. Oila nomi	Valiyev	46. Oila nomi	Valiyev	47. Oila nomi	Valiyev	48. Oila nomi	Valiyev
49. Oila nomi	Valiyev	50. Oila nomi	Valiyev	51. Oila nomi	Valiyev	52. Oila nomi	Valiyev
53. Oila nomi	Valiyev	54. Oila nomi	Valiyev	55. Oila nomi	Valiyev	56. Oila nomi	Valiyev
57. Oila nomi	Valiyev	58. Oila nomi	Valiyev	59. Oila nomi	Valiyev	60. Oila nomi	Valiyev
61. Oila nomi	Valiyev	62. Oila nomi	Valiyev	63. Oila nomi	Valiyev	64. Oila nomi	Valiyev
65. Oila nomi	Valiyev	66. Oila nomi	Valiyev	67. Oila nomi	Valiyev	68. Oila nomi	Valiyev
69. Oila nomi	Valiyev	70. Oila nomi	Valiyev	71. Oila nomi	Valiyev	72. Oila nomi	Valiyev
73. Oila nomi	Valiyev	74. Oila nomi	Valiyev	75. Oila nomi	Valiyev	76. Oila nomi	Valiyev
77. Oila nomi	Valiyev	78. Oila nomi	Valiyev	79. Oila nomi	Valiyev	80. Oila nomi	Valiyev
81. Oila nomi	Valiyev	82. Oila nomi	Valiyev	83. Oila nomi	Valiyev	84. Oila nomi	Valiyev
85. Oila nomi	Valiyev	86. Oila nomi	Valiyev	87. Oila nomi	Valiyev	88. Oila nomi	Valiyev
89. Oila nomi	Valiyev	90. Oila nomi	Valiyev	91. Oila nomi	Valiyev	92. Oila nomi	Valiyev
93. Oila nomi	Valiyev	94. Oila nomi	Valiyev	95. Oila nomi	Valiyev	96. Oila nomi	Valiyev
97. Oila nomi	Valiyev	98. Oila nomi	Valiyev	99. Oila nomi	Valiyev	100. Oila nomi	Valiyev

50-RASM. To'ldirilgan anketa

Agar html va css texnologiyalaridan xabardor bo'lsangiz, dasturingizga biron chiroq qo'shishingiz mumkin.

```
from ruwebio.output import put_html, put_file
from ruwebio.session import hold
from frontend import getInfo
from backend import fillDoc

title = "<h1 style='text-align:center'>Ўзбекистон \
Республикаси Фўқаросининг хорижа чиқиш биометрик \
паспортини расмийлаштириш учун АНКЕТА</h1>"
put_html(title)

userInfo = getInfo()
filename = fillDoc(userInfo)

text = "<h3>Анкета тайёр. \
Юқлаб олиш учун қўйидаги боғламани босинг.</h3>"
put_html(text)

with open(filename, 'rb') as f:
    anketa = f.read()
    put_file(filename, content=anketa)
    hold()
```

Dasturimiz tayyor. Albatta, dasturdan to'laqonli foydalanish uchun uni onlayn serverga yuklash zarur. Bu esa alohida katta mavzu. Bizning maqsadimiz Python dasturlash tilining imkoniyatlarini ko'rsatish bo'lgani bois loyihamizga shu yerda yakun yasaymiz.

АМАЛИЙОТ

- Анкета dasturini kengaytirish va boshqa blankalar ham qo'shish.
- Foydalanuvchiga blanka tanlash imkoniyatini berish uchun asosiy sahifaga tugmalar qo'yishingiz mumkin (put_buttons).

15-BOB

ТАРJIМОН TELEGRAM BOT

Ushbu bobda biz Telegram messengeri uchun O'zbek-Ing-liz tarjimon botini yaratishni o'rganamiz. Aslida, botlar ham veb-dasturlarning o'ziga xos ko'rinishidir. Avvalgi bobda yaratgan dasturimizdan farqli ravishda, bu loyihamizda frontend va-zifasini Telegram messengeri bajaradi. Ya'ni foydalanuvchilar backend bilan Telegram orqali muloqot qiladilar. Demak, frontend uchun qilishimiz kerak bo'lgan yagona ish Telegram botini ro'yxatdan o'tkazish va foydalanuvchi kiritgan matnlarni backendga uzatib turishni yo'lga qo'yish ekan. Loyihani boshlaymiz.

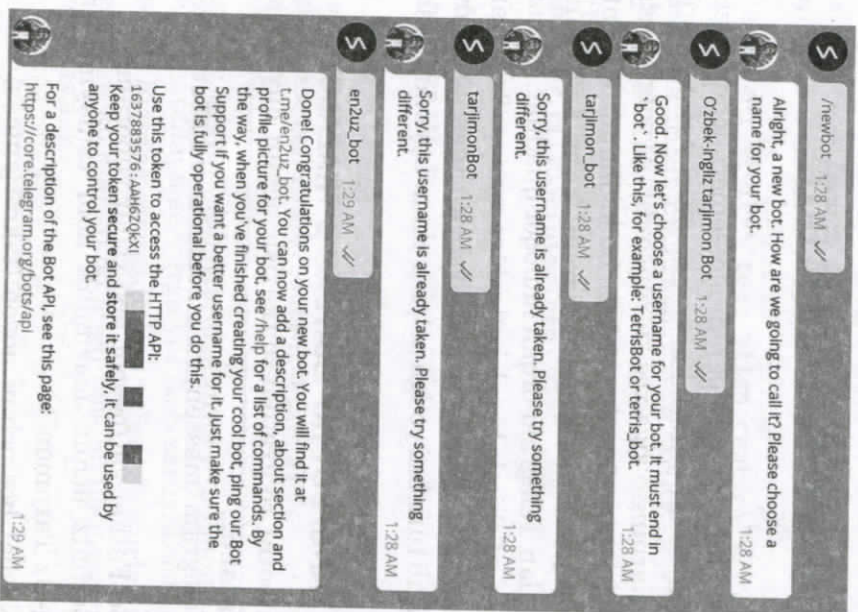
15.1. YANGI BOT RO'YXATDAN O'TKAZAMIZ

Yangi botni ro'yxatdan o'tkazish uchun quyidagi qadamlarni bajaramiz:

1. Telegram messengerini ochamiz va @BotFather botini topamiz.
2. BotFather ga /newbot deb xabar yuboramiz.
3. Botimiz uchun mazmunli nom tanlaymiz (masalan, "Uz-Eng-Uz Tarjimon").
4. Navbat bot uchun poyob (takrorlanmas) nom tanlash. Bot nomi lotin harflarida, joy tashlamasdan yozilishi va, albatta, bot so'zi bilan tugashi kerak. Masalan, eng2uz_bot. Agar ushbu nom band bo'lsa, BotFather boshqa nom tanlashni so'raydi.
5. BotFather siz tanlagan nomni qabul qilsa, sizga botning manzili va botga Admin sifatida ulanish uchun maxfiy kalit so'z (TOKEN) yuboradi. TOKENni saqlab qo'ying va boshqalar bilan ulashmang.

Marhamat, botimiz ro'yxatdan o'tdi. Keyingi qadam botimizni backend bilan bog'lash.

Bot yaratish jarayonini quyidagi rasmda ko'rishingiz mumkin.



51-RASM. Botni ro'yxatdan o'tkazish jarayoni

15.2. BOT UCHUN BACKEND

Backend ikki qismdan iborat bo'ladi:

1. Foydalanuvchi bilan mulqot (xabar olish-yuborish).
2. Matnlarni tarjima qilish.

Birinch qism uchun pyTelegramBotAPI moduli javob beradi. Modulni o'rnatish uchun quyidagi buyruqdan foydalanamiz:

```
pip install pyTelegramBotAPI
```

Matnlarni tarjima qilish uchun esa googletrans modulidan foydalanamiz. Biz bu modul bilan 12-bobda tanishgan edik. Agar o'rnatmagan bo'lsangiz, modulni quyidagicha o'rnatishingiz mumkin:

```
pip install googletrans
```

TARJIMON

Dasturimizni matn tarjimonidan boshlaymiz. Bu funktsiyamizni translate deb ataymiz. Funktsiya quyidagicha ishlaydi:

- Foydalanuvchidan matn qabul qiladi.
- Matn tilini aniqlaydi.
- Agar matn O'zbek tilida bo'lsa, Ingliz tiliga, agar Ingliz tilida bo'lsa, O'zbek tiliga tarjima qiladi.
- Foydalanuvchiga tarjima qilingan matnни qaytaradi.

Yangi translate.py faylini yaratamiz va quyidagilarni yozamiz:

```
from googletrans import Translator

def translate(matn):
    translator = Translator()
    # matn tilini aniqlaymiz:
    til = translator.detect(matn).lang
    if til=='en': # agar matn ingliz tilida bo'lsa...
        return translator.translate(matn,dest='uz').text
    else:
        return translator.translate(matn,dest='en').text
```

Tarjimon tayyor. Navbat foydalanuvchi bilan mulqotga.

BOT BILAN ALOQA

Dasturimiz foydalanuvchi bilan bot orqali muloqot qiladi. Demak, biz dasturimizni botga ulashimiz va foydalanuvchidan kelgan xabarlarini dasturimizga yo'naltirishimiz va, aksincha, tarjima qilingan matnlarni bot orqali qaytarib foydalanuvchiga yuborishimiz kerak.

Mana shu bizning asosiy dasturimiz bo'lgani uchun unga `main.py` deb nom beramiz va avvalroq yaratgan `translate.py` faylidan `translate` funksiyasini hamda `pyTelegramBotAPI` modulini import qilamiz.

```
① pyTelegramBotAPI dan to'g'ri foydalanish uchun modulning rasmiy sahifasiga murojaat etamiz:  
https://github.com/eternnoir/pyTelegramBotAPI
```

```
from translate import translate  
import telebot # pyTelegramBotAPI ning nomi
```

`telebot` – bu `pyTelegramBotAPI` modulining rasmiy nomi. Botimiz bilan bog'lanish uchun `telebot` moduli ichidagi `TeleBot` klassidan yangi obyekt yaratamiz (`tarjimobot`). Obyektning xususiyati sifatida `BotFather`dan kelgan `TOKEN`ni ko'rsatamiz.

```
TOKEN = "1555261510:AH74HavhV0A1WVaXUZxY5s8Ev83hFr"  
tarjimobot = telebot.TeleBot(token=TOKEN)
```

Bot bilan ishlash juda oson. `tarjimobot` obyekti ichida foydalanuvchidan keluvchi xabarlarini ushlaydigan maxsus `message_handler()` metodi bor. Bu metod parametr sifatida foydalanuvchidan kelgan xabarning turini qabul qiladi va shu xabarga mos keluvchi funksiyani ishga tushiradi.

Biz soddagina Telegram bot qilayotganimiz uchun ikki turli xabarlariga javob qaytarsak kifoya.

1. `/start` – foydalanuvchi botni ishga tushirganda keluvchi xabar.
2. Har qanday matnlarga ega xabarlar.

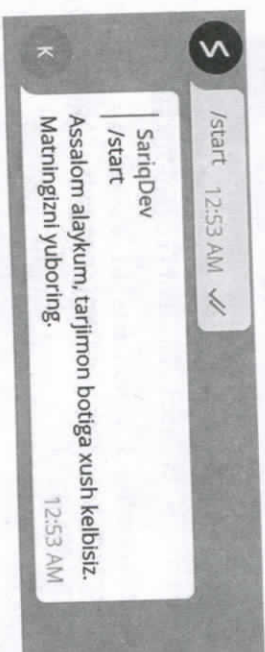
`/start` komandasi uchun funksiya yozamiz va bu funksiyani `message_handler()` metodiga uzatamiz. `message_handler()` bu funksiya qabul qiluvchi funksiya yoki dekorator deyiladi va unga funksiya uzatish uchun kodimizni quyidagicha yozamiz:

```
@tarjimobot.message_handler(commands=['start'])  
def salom(message):  
    xabar = "Assalomu alaykum, tarjimon botiga xush kelibsiz."  
    xabar += '\nMatningizni yuboring.'  
    tarjimobot.reply_to(message, xabar)  
    # botni ishga tushiramiz  
    tarjimobot.polling()
```

Kodni tahlil qilamiz:

- Dastavval biz `message_handler()` metodini chaqirdik va metod aynan qaysi turdagi xabarlariga javob berishini ko'rsatdik.
 - Agar foydalanuvchidan `start` komandasi kelsa, `message_handler()` o'zidan keyin turgan `salom()` funksiyasiga foydalanuvchidan kelgan xabarni (`message`) uzatadi.
 - `salom()` funksiyasi `.reply_to()` metodi yordamida foydalanuvchiga "Xush kelibsiz" xabarini qaytaradi.
 - Botni ishga tushirish uchun dasturimiz so'ngida `tarjimobot.polling()` metodini chaqiramiz.
- Dasturni ishga tushiramiz va Telegram orqali `en2uz_bot` ga murojaat etamiz:

```
python c:\python-darslari\42-tarjimon\main.py
```



Write a message...

52-RASM. en2uz_bot ishga tushdi

Botimiz ishga tushdi va biz kutgan xabarni qaytardi. Botni to'xtatish uchun Command Prompt oynasida Ctrl+C tugmasini bosamiz.

Navbat foydalanuvchidan kelgan xabarlarni tarjima qilishga. Foydalanuvchidan xabarlar matn, audio, video, emoji ko'rinishida kelishi mumkin. Xabarlar orasidan aynan matnlarni tanlab olish uchun yangi message_handler() yaratamiz:

```
@tarjimobot.message_handler(func=lambda msg: msg.
text is not None)
```

Qavs ichidagi lambda funksiyaning vazifasi xabarlar orasidan matni borlarini ajratib olish. Navbat matnlarni tarjima qiluvchi funksiyaga:

```
@tarjimobot.message_handler(func=lambda msg: msg.
text is not None)
def tarjima(message):
    msg = message.text
    tarjimobot.reply_to(message, translate(msg))
```

tarjima() funksiyasi xabar matnini ajratib oladi va translate() funksiyasiga yuboradi. reply_to() metodi esa translate() funksiyasidan qaytgan tarjimani foydalanuvchiga yuboradi.

main.py faylimizning to'liq ko'rinishi quyidagicha bo'ladi:

```
from translate import translate
import telebot

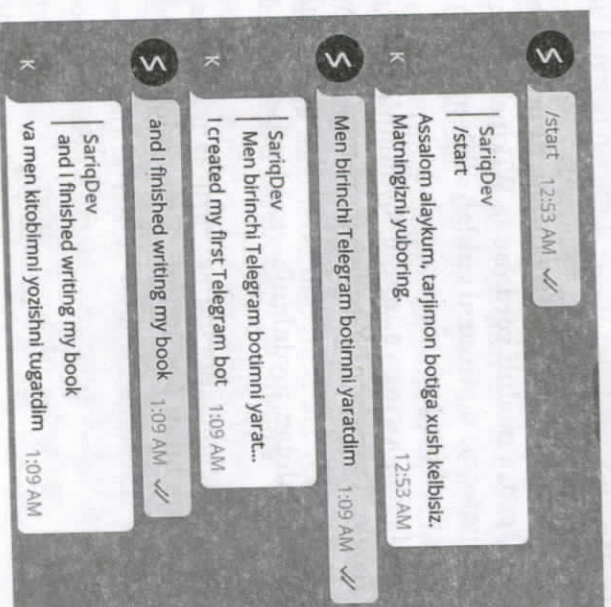
TOKEN = "1555261510:AAH784HAvh0A1BVaxUzXV5s8E83hFrN"
tarjimobot = telebot.TeleBot(token=TOKEN)

# \start komandasi uchun mas'ul funksiya
@tarjimobot.message_handler(commands=['start'])
def salom(message):
    xabar = "Assalomu alaykum, tarjimon botiga xush kelbisiz."
    xabar += '\Matningizni yuboring.'
    tarjimobot.reply_to(message, xabar)
```

```
# matnlar uchun mas'ul funksiya
@tarjimobot.message_handler(func=lambda msg: msg.
text is not None)
def tarjima(message):
    msg = message.text
    tarjimobot.reply_to(message, translate(msg))

# botni ishga tushiramiz
tarjimobot.polling()
```

Botni qaytadan ishga tushiramiz va Telegram orqali matnlar yuborib ko'ramiz:



53-RASM. en2uz_bot

Botimiz ishga tushdi va biz xohlagandek ishlayapti. Dasturimizning backend qismi bizning kompyuterimizda joylashgani uchun foydalanuvchilardan kelgan matnlar bizning kompyuterimizga kelib tushmoqda. Ya'ni bizning kompyuterimiz server vazifasini bajarmoqda. Agar Command Prompt oy-

nasini yopsak yoki kompyuterimizni o'chirsak, bot bilan aloqa ham uziladi. Shuning uchun botni ommaga taqdim etishdan avval backendni onlayn serverga ko'chirishimiz kerak. Lekin bu ham alohida katta mavzu va aksar onlayn serverlar pul-lik bo'lgani uchun serverga ko'chirish mavzusi haqida alohida onlayn video darsliklar tayyorlaymiz.

① Katta ehtimol bilan kitob qo'lingizga tekuniga qadar videolarimiz ham www.youtube.com/sariqdev kanalimizga yuklangan bo'ladi.

AMALIYOT

PyPi.org sahifasida qiziqarli modullar juda ham ko'p. Misol uchun, wikipedia moduli yordamida Wikipedia sahifasidan istalgan mavzuga doir matnlarni yuklab olish mumkin: <https://pypi.org/project/wikipedia/>

Modul sahifasiga kiring va undan foydalanishni o'rganing. Sahifani Google Translate, Yandex Tarjimon yoki yuqorida yaratgan botingiz yordamida tarjima qilishingiz mumkin.

wikipedia modulidan foydalanib yangi veb-dastur yoki bot yarating.

Agar boshqa qiziqarli loyihalar yaratsangiz, t.me/sariqdev chat orqali biz bilan bog'laning. Eng qiziqarli loyihalar @sariqdev kanalida ommaga taqdim qilinadi.

SO'NGSO'Z

Tabkrlaymiz! Siz dasturlash asoslarini o'rgandingiz va olgan bilimlaringizni mazmunli loyihalarda qo'llab foydali tajribaga ega bo'ldingiz. Sizda dastur va dasturlash haqida tushuncha hosil bo'ldi. Muhimi, endi siz dasturlash bo'yicha ko'nikmalaringizni rivojlantirish uchun dasturlashning turli yo'nalishlariga ishonch bilan qadam qo'ya olasiz.

Sizning keyingi qadamlaringiz qanday bo'lishi kerak? Avvalo, o'rganishdan to'xtamang. O'zingizni qiziqtirgan loyihalar ustida ishlashni davom eting. Siz o'zingizning o'yiningizni ixtiro qilishingiz yoki mavjud o'yinlarni qayta yaratishingiz mumkin. Siz o'zingizning veb-ilovangizni yaratishingiz yoki sizga yoqqan veb sahifaning muqobilini yaratib ko'rishingiz mumkin. Illoji bo'lsa, boshqa odamlarni dasturlaringizdan foydalanishga taklif eting. Agar siz o'yin yozsangiz, uni boshqa odamlar o'yinlariga imkon bering. Agar siz veb-dastur yaratadigan bo'lsangiz, uni internetda joylashtiring va boshqalarni uni sinab ko'rishga undang. Foydalanuvchilaringizni tinglang va ularning fikr-mulohazalarini loyihalaringsizga qo'shishga harakat qiling.

Loyihalari ustida ishlar ekansiz, albatta, yangi va yangi muammolarga duch kelasiz. Shunday holatlarda tushkunlikka tushmasdan, aksincha, yordam so'rashni, onlayn qidiruv xizmatlaridan foydalanishni o'rganing. Google va StackOverflow sahifalari bilan do'stlashib oling (bu ikki sahifa sizning doimiy hamrohingiz bo'ladi). Boshqa dasturchilar bilan aloqa o'rnatib, dasturlashga oid guruh va kanallarga a'zo bo'ling. Illoji bo'lsa, turli musobagalarda qatnashing.

Sizni qiziqtirgan loyihalar ustida ishlash va, umuman, Python ko'nikmalarini rivojlantirish o'rtasidagi muvozanatni

saqlashga intiling. Har bir yangi loyiha bilan o'sib borayotganingizni his qiling. Bir joyda qotib qolmang. Sizing bilim va ko'nikmalaringiz o'z ustingizda mustaqil ishlashga yetadi. Shuning uchun yana bir bor qaytaraman: o'rganishdan va o'sishdan to'xtamang.

Sizni yana bir bor katta masofani bosib o'tganingiz bilan dasturchilar qatoridan joy olganingiz bilan tabriklayman. Kelajakdagi ishlaringizga omad tilayman!

① Kitob yoki video darsliklar bo'yicha savollaringizni, talab va takliflaringizni Telegram messengeri orqali bevosita muallifga yo'llashingiz mumkin. Muallif manzili: @sariqdevadmi.

MUNDARIJA

Muqaddima 3

1-QISM. ASOSLAR

1-bob. Kompyuter va dastur	13
1.1. Kompyutering ichki tuzilishi	14
1.2. Dastur va dasturlash	15
2-bob. Ilk qadamlar	17
2.1. Python o'rnatamiz	18
2.2. Dasturlash muhiti	19
2.3. Brauzerda dastur yozish	21
2.4. Birinchi dasturimiz	22
2.5. Print() funksiyasi	24
2.6. Sintaksis xatolik (syntax error)	26
2.6. Arifmetik amallar	27
2.7. Izohlar	29
3-bob. O'zgaruvchilar va ma'lumot turlari	31
3.1. O'zgaruvchilar	31
3.2. String – matn	34
3.3. Matnlar bilan ishlash	37
3.4. Input() – foydalannuvchi bilan mulogot	39
3.5. Sonlar	41
3.6. Ro'yxatlar – lists	49
3.7. Ro'yxatlar bilan ishlash	55
3.8. Tuples – o'zgarmas ro'yxatlar	61
3.9. For sikli	64
4-bob. Shartlar va tarmoqlanish	71
4.1. Taqqoslash operatorlari	72
4.2. If-else operatori	74
4.3. Bir nechta shartlarni tekshirish	79
4.4. And va or operatorlari	83
4.5. Boolean ma'lumotlar turi	85
4.6. Shartlarni ketma-ket tekshirish	86
4.7. Ro'yxat tarkibini tekshirish	87

5-bob. Lugʻat va toʻplam.....	92
5.1. Lugʻat nima?.....	92
5.2. Lugʻat bilan ishlash	94
5.2. Lugʻat elementlari bilan ishlash	98
5.3. Toʻplamlar	105
5.4. Nesting	110
6-bob. While sikli.....	117
6.1. While qanday ishlaydi?.....	117
6.2. While, roʻyxatlar va lugʻatlar	123
7-bob. Xatolar bilan ishlash.....	128
7.1. Xatolarning turlari	128
7.2. Exceptions	139
7.3. Xatolarning oldini olish	145
8-bob. Funksiya.....	147
8.1. Funksiya yaratamiz.....	148
8.2. Argument va parametrlar.....	151
8.3. Qiymat qaytaruvchi funksiya	157
8.4. Funksiya va roʻyxat	162
8.5. Moslashuvchan funksiya (*args, **kwargs).....	165
8.6. Modullar	167
8.7. Lambda – nomsiz funksiya.....	174
9-bob. Obyektni yoʻnaltirilgan dasturlash	181
9.1. Oop nima?.....	181
9.2. Oop tamoyillari	188
9.3. Klasslar	191
9.4. Obyektlar bilan ishlash.....	198
9.5. Voriylik va polimorfizm	210
9.6. Inkapsulyatsiya, klassga xos xususiyat va metodlar	217
9.7. Dunder metodlar	224
10-bob. Fayllar bilan ishlash.....	240
10.1. Fayl bilan maʼlumot oʻqish	240
10.3. Oʻzgaruvchilarni faylda saqlash	248
10.4. Json.....	250
11-bob. Dasturni tekshirish.....	261
11.1. Funksiyani tekshirish.....	261
11.2. Klasslarni tekshirish	268

12-Bob. Kutubxonalar.....	276
12.1. Python standart kutubxonasi	276
12.2. Tashqi kutubxonalar	286

2-QISM. LOYIHALAR

13-bob. "Son topish" va "soʻz topish" oʻyinlari.....	300
13.1. "Son topish" oʻyini	300
13.2. "Soʻz topish" oʻyini.....	310
14-bob. Veb-dastur yaratamiz.....	315
14.1. Backend-frontend	315
14.2. Asosiy dastur.....	320
15-bob. Telegram bot	323
15.1. Yangi bot roʻyxatidan oʻtkazamiz.....	323
15.2. Bot uchun backend	324
Soʻngsoʻz.....	331