**O‘ZBEKISTON RESPUBLIKASI HARBIY XAVFSIZLIK VA MUDOFAA UNIVERSITETINING AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI VA HARBIY ALOQA INSTITUTI**

**“AXBOROT TEXNOLOGIYALARI VA** **SUN’IY INTELLEKT”**

**kafedrasi**

****

**“PYTHON DASTURLASH TILI” FANIDAN**

**O‘QUV-USLUBIY MAJMUA**



**Toshkent 2025**

“TASDIQLAYMAN”

O‘R HXvaM UNIVERSITETINING AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI VA HARBIY ALOQA INSTITUTI BOSHLIG‘INING O‘QUV VA ILMIY ISHLAR BO‘YICHA BIRINCHI O‘RINBOSARI

polkovnik

O. Mirjalolov

2025-yil “\_\_\_\_” \_\_\_\_\_\_\_\_\_

“KELISHILGAN”

AXBOROT TEXNOLOGIYALARI VA SUN’IY

INTELLEKT KAFEDRASI BOSHLIG’I

kapitan

B. Yusupov

AXBOROT TEXNOLOGIYALARI VA SUN’IY

INTELLEKT KAFEDRASI SUN’IY INTELLEKT   
SIKLI BOSHLIG’I

kapitan

M. Isakov

AKTvaHAI Ilmiy-uslubiy Kengashining 2025-yil \_\_\_ \_\_\_\_\_\_\_dagi yig‘ilishida muhokama qilindi, \_\_\_\_-son bayonnoma.

AKTvaHAI “Axborot texnologiyalari va sun’iy intellekti” kafedrasida 2025-yil \_\_\_ \_\_\_\_\_\_dagi yig‘ilishidagi \_\_\_-sonli bayonnomada muhokama qilindi.

**Tuzuvchilаr:**

|  |  |
| --- | --- |
| kapitan  B.K. Yusupov | − Axborot kommunikatsiya texnologiyalari va harbiy aloqa instituti “Axborot texnologiyalari va sun’iy intellekt” kafedrasi boshlig‘i, PhD, professori; |
| kapitan  M.M. Isakov | − Axborot kommunikatsiya texnologiyalari va harbiy aloqa instituti “Axborot texnologiyalari va sun’iy intellekt” kafedrasi sun’iy intellekt sikli boshlig‘i; |
| katta leytenant  R.O’. Qodomboyev | − Axborot kommunikatsiya texnologiyalari va harbiy aloqa instituti “Axborot texnologiyalari va sun’iy intellekt” kafedrasi sun’iy intellekt sikli katta o’qituvchisi; |

**Taqrizchilar:**

|  |  |  |
| --- | --- | --- |
| podpolkovnik  O. Temirov |  | O‘R QK BOSH SHTABI AAT va AH BB telekommunikatsiya xavfsizligi va kriptografik himoya boshqarmasi boshlig‘i |
| podpolkovnik  O. Abdiroziqov |  | O‘R HX VA MU “Istiqbolli harbiy texnologiyalar” kafedrasi boshlig‘i, PhD, dotsent. |

**M U N D A R I J A**

[KIRISH 5](file:///D:\Kafedra%20hujjatlari\2025-2026\Fanlar%202025-2026\Python%20dasturlash%20tili%202025-2026\5-kurs\UMK%20-%20PYTHON%20DASTURLASH%20TILI.docx#_Toc206592468)

[Fan bo'yicha o'qitiladigan materiallar to'plami 8](file:///D:\Kafedra%20hujjatlari\2025-2026\Fanlar%202025-2026\Python%20dasturlash%20tili%202025-2026\5-kurs\UMK%20-%20PYTHON%20DASTURLASH%20TILI.docx#_Toc206592470)

[316](#_Toc206592471)

[“SWOT-tahlil” metodi 316](#_Toc206592472)

KIRISH

Ushbu o‘quv-uslubiy majmuada “Python dasturlash tili” faniga tegishli bo‘lgan dasturlash qismiga tegishli bo‘lgan barcha mavzular bo‘yicha kursantlarga Davlat ta’lim standartlari asosida yetkazilishi shart bo‘lgan minimum bilimlar va ko‘nikmalarni to‘la qamrab olingan.

Fanning asosiy maqsadi - yuqori malakali aloqa qo‘shinlari ofitserlarini tayyorlashda axborot-kommunikatsiya texnologiyalari vositalarining dasturiy ta’minoti tuzilishi va ishlash jarayonini hamda yaratilish bosqichlarini nazariy va amaliy jihatdan o‘rgatishdan iborat.

Fanni o‘zlashtirish kursantlarning “Informatika”, “Kompyuter tizimlarini ekspluatasiya qilish”, “Dasturlash texnologiyalari” fanlaridan olgan bilimlariga tayanadi. Fanni o‘zlashtirish quyidagi mashg‘ulot turlarini o‘z ichiga oladi: ma’ruza, seminar, guruhli va amaliy mashg‘ulotlar, shuningdek kursantlarga mustaqil tayyorgarlik vaqtida maslahatlar berish. Ma’ruza materiallari bayoni mustaqil va tugallangan hususiyatga ega bo‘lib, avval bayon qilingan materiallarga mantiqiy bog‘langan hamda boshqa fanlarda, hamda amaliyotda qo‘llanishga yo‘naltirilgan bo‘lishi kerak. Amaliy mashg‘ulotlarda kursantlar olgan nazariy bilimlarini qo‘llay olishni o‘rganishlari kerak.

**Fanning maqsad va vazifalari**

**Fanni o‘qitishdan maqsad** – harbiy dasturiy mahsulotlarni ishlab chiqish usullarini, loyihalash usullarini, dasturiy ta’minotni ishlab chiqishni, dasturiy mahsulotning effektivligini, yo‘nalishning profiliga mos bilim, ko‘nikma va malakani shakllantirishdan iborat.

**Fanning vazifasi** – harbiy dasturiy mahsulotlarni yaratish bosqichlari, tamoyillari, testlash usullari, arxitekturasi va loyihalash usullaridan foydalanishni o‘rgatishdan iborat.

Fan bo‘yicha kursantlarning bilimiga, ko‘nikma va malakasiga qo‘yiladigan talablar:

“Harbiy maqsadlarga yo‘naltirilgan dasturlash” o‘quv fanini uzlashtirish jarayonida amalga oshiriladigan masalalar doirasida kursantlar:

- shaxsiy kompyuterlardan foydalanishni; dasturiy mahsulotlarnini yaratishning eng zamonaviy dasturlash texnologiyalari to‘g‘risida tasavvurga ega bo‘lishi;

- masalalarni algoritmlash va modellashtirish usullarini, optimal dasturlash tilini tanlashni; tayyor dasturiy mahsulotlardan foydalanish, testdan o‘tkazishhamda ularga xizmat ko‘rsatish usullarini bilishi va ishlata olishni, dasturlarni funksional xarakteristikalariga talablar qo‘yishni;

- programmani ishlab chiqish bosqichlarini aniqlashni; algoritmlarni yozish usullarini bilishi kerak.

- dastur algoritmini, blok-sxemasini ishlab chiqish, tanlab olingan algoritmiga test misollarini ishlab chiqish, Dasturdagi xatoliklar topish va tuzatish ko‘nikmalariga ega bo‘lishi kerak.

**Fan bo‘yicha kursantlarning tasavvur, bilim, ko‘nikma va**

**malakalariga qo‘yiladigan talablar**

***Fanni o‘rganish natijasida kursantlar quyidagi tushunchalarga ega bo‘lishi lozim:***

O‘zbekiston Respublikasi Qurolli Kuchlari aloqa tizimida dasturlash asoslarining o‘rni va ro‘li haqida;

* dasturlash asoslarida ishlanayotgan harbiy dasturlarni hamda ularni qo‘llay olish tadbirlarini;
* dasturlash asoslari algoritmlarini.

***quyidagilarni bilishi va qo‘llay olishi lozim:***

* Python C, C++, C# dasturlash tillarini;
* kompyuter tizimlarini muhofazasini ta’minlashda qo‘llaniladigan dasturiy vositalarni;
* O‘zbekiston Respublikasi Mudofaa Vazirligi qo‘shinlari axborot tizimlarida dasturiy ta’minotni ta’minlash bo‘yicha me’yoriy hujjatlarni.

***quyidagi ko‘nikmalarga ega bo‘lishi lozim:***

* dasturlash tillarida ishlanayotgan dasturlarga kirish huquqlarini chegaralash tizimini yaratish bo‘yicha;
* ma’lumotlarni muhofazalashning zamonaviy dasturiy vositalarini sozlash bo‘yicha bilim va ko‘nikmalarga ega bo‘lishi kerak.

Shuning uchun harbiy sohada dasturiy ta’minotni ta’minlash uchun dasturlash asoslarini yaxshi bilish, loyihalash va takomillashtirish talab qilinadi.

**Fanni o‘qitishda zamonaviy axborot va pedagogik texnologiyalar**

O‘quv jarayoni bilan bog‘liq ta’lim sifatini belgilovchi holatlar quyidagilar: yuqori ilmiy-pedagogik darajada dars berish, muammoli ma’ruzalar o‘qish, darslarni savol-javob tarzida qiziqarli tashkil qilish, ilg‘or pedagogik texnologiyalardan va mul’timedia vositalaridan foydalanish, kursantlarni undaydigan, o‘ylantiradigan muammolarni, ular oldiga qo‘yish, talabchanlik, kursantlar bilan individual ishlash, erkin muloqot yuritishga, ilmiy izlanishga jalb qilish. Kursantlarning “Python dasturlash tili” fanini o‘zlashtirishlari uchun o‘qitishning zamonaviy va ilg‘or usullaridan foydalanish, yangi axborot−pedogogik texnologiyalarini tadbiq qilish muhim ahamiyatga egadir. Fanni o‘zlashtirishda o‘quv qo‘llanmalar, ma’ruza matnlari, tarqatma materiallar, elektron materiallar, virtual stendlardan foydalaniladi. Ma’ruza, guruhli va amaliy mashg‘ulotlarida mos ravishdagi zamonaviy pedagogik va axborot texnologiyalaridan foydalaniladi.

Mashg‘ulotlarni asosiy qismini guruh va amaliy mashg‘ulotlardan iborat bo‘lib, ularda o‘quv elementlarni mazmunlari va mavzularni bir-birlariga mantiqiy bog‘liqligi o‘rganiladi. Mashg‘ulotlarda kursantlar har bir element va unga qo‘yilgan o‘quv savollari bo‘yicha mustaqil o‘rganib, o‘qituvchi yordamida bilim va ko‘nikmalarini takomillashtirishlari lozim. Mashg‘ulotlarni asosiy turlari zamonaviy pedagogik va innovatsion texnologiya usullari va interaktiv shakllarini qo‘llagan holda olib borilishi va unda kursantlarni faol qatnashuvi fanni o‘rganishning asosiy omillaridan biri hisoblanadi.

**O‘ZBEKISTON RESPUBLIKASI HARBIY XAVFSIZLIK VA MUDOFAA UNIVERSITETINING AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI VA HARBIY ALOQA INSTITUTI**

**“AXBOROT TEXNOLOGIYALARI VA SUN’IY INTELLEKTI” kafedrasi**

****

**«PYTHON DASTURLASH TILI»**

**FANIDAN**

Fan bo'yicha o'qitiladigan   
materiallar to'plami

**Toshkent – 2025**

**MA’RUZA MASHG‘ULOTI UCHUN O‘QUV MATERIALLARI**

**1-mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**1-mashg‘ulot.** Python dasturlash tilining klassifikatsiyasi va rivojlanish tarixi. Python dasturlash tilining asosiy tushunchalari.

**O‘quv savollari:**

1. Fanning mazmuni, maqsadi, vazifalari;
2. Pythonni o‘rnatish. PyCharm dasturini o‘rnatish;
3. Pythonda “Hello world!” dasturini tuzish.
4. Python tilining asosiy operatorlari bilan tanishish

**1. O‘quv fanining maqsad va vazifalari .**

“Python dasturlash tili” fanining maqsadi kursantlarga dasturlash texnologiyalari haqida tushuncha berish va O‘zbekiston Respublikasi Qurolli Kuchlari tizimida qo‘llaniladigan dasturlash tillarini o‘rgatish, ularga operatsion tizim tomonidan qo‘llaniladigan tayyor dasturiy ta’minotni yaratish uchun zarur bo‘lgan dasturiy vositalardan foydalanishni o‘rgatish.

Kredit-moduli tizimi asosida fan bo‘yicha o‘quv kursi ma’ruza, amaliy mashg‘ulotlar, shuningdek, mavzu bo‘yicha topshiriq va mustaqil topshiriqlarni o‘z ichiga oladi. Ma’ruza o‘quv materiali, amaliy ishlar ko‘rsatilgan mavzular bo‘yicha nazariy va amaliy ma’lumotlar beradi, amaliy ish, mustaqil ish va ishlarni bajarish tartibini tushuntiradi. Kursda belgilangan o‘quv materiali kursantlar tomonidan mustaqil ravishda o‘rganiladi, testlar, amaliy ishlar individual ravishda amalga oshiriladi.

Kurs 1 semestrga mo‘ljallangan bo‘lib, umumiy auditoriya 60 soatni tashkil qiladi. 7 Semestrda, 6 soat nazariya, 54 soat amaliyot, jami 60 soat auditoriya soati mavjud.

Sinfdagi yuklamadan tashqari mustaqil ishlarga ham vaqt ajratiladi. Shunga ko‘ra, 6-semestrda 60 soat mustaqil ish.

Mustaqil tayyorgarlik mashg‘ulotlatlarida kursantdan mustaqil ravishda turli xil dasturlarni yaratishni va yaratgan dasturi haqida gapirib berishi talab qilinadi.

**Kursantlarni baholash**

Kursantlarning bilimini baholash semestr davomida va yakuniy nazorat jarayonida o‘quv materialini o‘zlashtirish ko‘rsatkichi asosida amalga oshiriladi.

7-semestr davomida “Python dasturlash tili” fanini o‘rganishda kursantlar 100 ballik tizim bo‘yicha baholanadi. Shundan joriy va oraliq nazoratga 20 ball, yakuniy nazoratga esa 40 ball beriladi. Joriy va oraliq nazorat ballarining umumiy natijasi 30 balldan past boʻlgan kursantlar yakuniy nazorat imtihoniga qoʻyilmaydi. Yakuniy nazoratga kiritilgan, unda 30 va undan ortiq ball to‘plagan kursantlar o‘tgan semestrda fanni o‘zlashtirgan deb hisoblanadi.

Joriy, oraliq va yakuniy nazorat ballari quyidagicha taqsimlanadi:

|  |  |
| --- | --- |
| Joriy nazorat | 40 ball |
| Oraliq nazorat | 20 ball |
| Yakuniy nazorat | 40 ball |
| Fan bo‘yicha jami: | 100 ball |

**Python dasturlash tili va uning rivojlanish tarixi.**

**Python dasturlash tili.** Python yuqori darajadagi dasturlash tili boʻlib, u AT ning turli sohalarida, masalan, mashinani oʻrganish, turli ilovalarni ishlab chiqish, web illovalari ishlab chiqish, tahlil qilish va boshqalarda keng qoʻllaniladi.

2019-yilda Python Java tilini 10 foizga ortda qoldirib, eng ommabop dasturlash tiliga aylandi. Bu juda ko‘p sabablarga bog‘liq bo‘lib, ulardan biri malakali mutaxassislarga to‘lanadigan ish haqining yuqoriligidir (yiliga taxminan 100 ming dollar).

Hozirgi kunda Turli xil dasturlash tillari mavjud bo‘lib, odatda ular ma’lum bir sohada (yoki bir nechta) liderlik qilishadi. Python dasturlash tili bo‘lsa, turli xil sohalarda ilovalar yaratish imkomiyatiga egaligi, dasturchilarni o‘ziga jalb qilmasdan qolmaydi.

Web ilovalar ishlab chiqish bozorining kichik qismini Python tili yaratilgan web ilovalari egallagan. Python tilini desktop ilovalarini yozish uchun ham ishlatilsa, mashinani o‘rganish (machine learning) sohasida to‘liq yetakchilik qiladi.

**Python nomining kelib chiqish tarixi.**

Gvido van Rossum python dasturlash tilini yaratganidan keyin, uni nomlamasdan turib tarqatishni hohlamadi. Ammo mukammal nom tanlash uchun vaqt sarflash Gvido uchun vaqtni behuda o‘tkazish hisoblanardi. Shunda uning xayoliga kelgan birinchi fikr “Python” bo‘ldi va dasturlash tilining nomini “Python”deb nomlab qo‘yaqoldi. Chunki 1970-yillarning boshida mashhur bo‘lgan, “Monty Python’s Flying Circus” komediya seryali uning sevimli shoularidan biri edi.

Yangi til nomi bilan bir qatorda logotip o‘ylab topish kerak edi va Guido tasodifiy shrift tanladi va “Python” so‘zini shu shriftda yozdi. Gvido bu nomning ilonlarga aloqasi yo‘qligini qayta-qayta ta’kidlagan bo‘lsada, ommaning fikriga ta’sir qilishning ilojisi yo‘q edi.

Ko‘pchilik dasturchilar Python tilini ilonlar bilan bog‘lab, uning nomini emas turli turdagi ilonlar rasmlari bilan kitoblar chop qilishar edi. Bu holat 2006 yilgacha davom etdi.

**Payton** (python) **yoki Piton** (питон) **– bu qanday talaffuz qilinadi?**

Bu ingliz teleko‘rsatuvining nomi bo‘ladimi yoki “ilon” so‘zining inglizcha tarjimasi bo‘ladimi,  “Python” nomi Pyton kabi to‘g‘ri talaffuz qilinadi . Biroq, rus dasturchilarining taxminan 80% “Piton” (питон)deb talaffuz qilishga o‘rganib qolishgan.

 Ushbu variantlardan birini aniq to‘g‘ri ikkinchisini xato deb aytish qiyin. Biroq, “Piton” deb talaffuz qilish varianti faqat rus tilida so‘zlashuvchi suhbatdoshlar bilan suhbatda foydalanish uchun mos keladi, chunki har qanday xalqaro konferentsiyada “Piton” so‘zi tushunarsiz bo‘ladi. Chunki ingliz tilida bunday so‘z mavjud emas.

Bunday konferensiyalarda faqat “Python (Python)” deb talaffuz qilish kerak bo‘ladi.

**Logotip.** Logotipda ikkita ilon kvadrat shaklda tasvirlangan, bu esa ko‘pincha foydalanuvchilarni til nomini sudralib yuruvchi bilan bog‘lashga undaydi.



*1-rasm. Python dasturlash tili logotipi*

Bu logotip muallifning ukasi, dasturchi va tip dizayneri Joost van Rossum tomonidan yaratilgan. U logotip dizaynini ishlab chiqshda kvadrat shaklini egallagan ikkita ilon va Flux Regular matn shriftida yozilgan Python so‘zidan foydalandi.

**Yaratilish tarixi.** Til 1980-yillarning oxirida dasturchi Guido van Rossum tomonidan ishlab chiqilgan. O‘sha paytda u Niderlandiyadagi matematika va informatika markazida ishlayotgan edi.

Gvido van Rossum maktab yillaridanoq apparat vositalari bilan ishlashni yaxshi ko‘rardi va u tengdoshlari tomonidan qo‘llab-quvvatlanmasa ham, ma’qullamasa ham, bu uning mustaqil ravishda dasturlash tilini rivojlantirishiga to‘sqinlik qilmadi.

Rossum bo‘sh vaqtlarida Pythonda ishlagan va u bir paytlar rivojlanishiga yordam bergan ABC dasturlash tiliga asoslanib ishlagan.

**Python dasturlash tili tarixidagi bosqichlar** :

* **1991-yilning fevralda** python tilining kodlari *alt.sources* saytida chop etildi.  Til ob’ektga yo‘naltirilgan yondashuvga amal qildi va obyekt, class, metodlar, merosxorlik, funktsiyalar, istisnolarni qayta ishlash va barcha asosiy ma’lumotlar tuzilmalari bilan ishlash imkoniyatiga ega edi.
* **2000 yilda Pythonning ikkinchi versiyasi chiqdi**. Unga ko‘plab muhim vositalar, jumladan Unicode yordami va qoldiqlarni yig‘uvchi funksiyasi qo‘shildi.
* **2008-yil 3-dekabrda Pythonning uchinchi versiyasi ishlab chiqildi.** Bu versiyasi hozircha oxirgi versiya hisoblanadi. Bu versiyada tilning ko‘pgina xususiyatlari qayta ishlangan va oldingi versiyalar bilan mos kelmaydigan holga keltirildi. Uchinchi versiyaning funksionalligi ikkinchisidan kam bo‘lmasada, tilning rivojlanishi ikki tarmoqqa bo‘lingan. Kimdir eski loyihalarni qo‘llab-quvvatlash uchun Pythonning 2-versiyasida foydalanishni davom ettirsa, kimdir butunlay uchinchi versiyada ishlashni tanladi.

Ikkinchi versiyaning tugatilish vaqti 2015 yilga belgilangan edi. Biroq barcha mavjud kodlarni Python 3 ga o‘tkazishga ulgurmaslikdan qo‘rqib,  Python 2 ning ishlash muddati 2020 yilgacha uzaytirilgan.

Pythonning sintaksisi uni har doim boshqa dasturlash tillaridan ajratib turadi. U ortiqcha operatorlardan xalos qilingan. Oddiy ingliz tili bilan sintaksisning o‘xshashligi hatto oddiy foydalanuvchiga ham kodni tushunishga imkon beradi. Bundan tashqari, dasturchi kamroq kod yozadi, chunki ortiqcha belgilarni ishlatishning hojati bo‘lmaydi. Masalan ; { }….

Pythonning oddiyligi qisman tilning ABC tiliga asoslanganligi bilan bog‘liq bo‘lib, u dasturlashni va dasturchi bo‘lmaganlarning kundalik ishlarini o‘rgatish uchun ishlatilgan.

**Python kod yozishni soddalashtiradi va rivojlanishni tezlashtiradi, chunki u quyidagi xususiyatlarga ega:**

* **Dinamik tip.** Dasturchi o‘zgaruvchilar tipini ko‘rsatishi shart emas, tilning o‘zi o‘zgaruvchiga qanday ma’lumot yuklanganiga qarab tipini aniqlab oladi.
* **Funktsiya orqali bir nechta qiymatlarni qaytarish.** Ushbu qiymatlarni vergul bilan ajratilgan holda to‘rtburchak qavslar ichiga yoziladi va avtomatik ravishda ro‘yxatga aylantiriladi. Funktsiyadan [massivni](https://all-python.ru/osnovy/massivy.html) qaytarish uchun “return ro‘yxat\_nomi” yozish kifoya.
* **Avtomatik xotiradan joy ajratish.** Dasturchi o‘zgaruvchilarga xotiradan joy ajratish uchun ortiqcha kod yozishi talab qilinmaydi. Bu, bir tomondan, dasturchining dastur ustidan nazoratini kamaytiradi, ikkinchi tomondan, rivojlanish sezilarli darajada tezlashadi.
* **Chiqindilarni yig‘uvchi**. Agar ob’ekt keraksiz bo‘lib qolsa, u chiqindi yig‘uvchi tomonidan avtomatik ravishda o‘chiriladi. Chiqindi yig‘uvchi sizga xotiradan foydalanishni optimallashtirish va keraksiz narsalarni qo‘lda o‘chirmaslik imkonini beradi.
* **a, b = b, a**  **ifodasi**. Bu ifoda o‘zgaruvchilarning qiymatlarini o‘zaro almashtiradi. Endi **a** o‘zgaruvchining qiymati **b** o‘zgaruvchisiga yuklandi va aksincha. Shunday qilib, siz nafaqat ikkita o‘zgaruvchining, balki uch va undan ortiq ob’yektlarning qiymatlarini mos ravishda bir-biriga almashtirishingiz mumkin. Faqat taraflardagi o‘zgaruvchilar soni teng bo‘lishi talab qilinadi.
* **O‘zgaruvchi tipining qiymatga bog‘liqligi**.  O‘zgaruvchining qiymati - bu uning turini va boshqa xususiyatlarini belgilaydigan atributlarga ega bo‘lgan qandaydir ob’ekt. O‘zgaruvchi esa bu ob’ektning nomidir. Ushbu yondashuv qat’iy tipdagi ta’riflarga bo‘lgan ehtiyojni yo‘qga chiqaradi va o‘zgaruvchilarga boshqa turdagi qiymatni qayta tayinlashni sezilarli darajada soddalashtirdi.
* **For tsikli** . Pythonda massivlar, ro‘yxatlar va boshqa konteynerlar bilan ishlash oson va qulay. Uning barcha elementlarini takrorlash zarur bo‘lganda, konstruktsiya quyidagicha ko‘rinadi: **for x in konteyner:** *(iteratsiya 0 dan oxirgi -1 elementgacha elementlarni x ga qaytaradi)*. Agar siz tsiklni aniq bir miqdorda takrorlanishini amalga oshirish uchun **range**() funksiyasidan foydalanish kerak bo‘ladi: **for x in range(1,9):** *(tsikl  1 dan 8 gacha bo‘lgan qiymatlarini x ga uzatadi)*.
* **Machine learning sohasidagi yetakchiligi.** Python dasturlash tili “Mashina o‘rganish” (Machine learning) sohasida yaqqol yetakchilik qilib kelmoqda. Ilm-fan bilan u yoki bu tarzda shug‘ullanadigan odamlar kod yozish kabi narsalarga ko‘p vaqt sarflamaslikni afzal ko‘radilar, shuning uchun Python ularga yuklangan vazifalarni amalga oshirish uchun juda mos keladi.

**Python ham soddalik, ham kuchli vositalarni o‘zaro birlashtiradi. U deyarli har qanday dasturning prototipini yaratish uchun ishlatilishi mumkin.**

**Kod misoli** :

def kattasi(a, b):

if a > b:

print(a, “katta”, b, “dan”)

else:

print(b, “kichik”, a, “dan”)

def max\_mass (massiv):

max = 0

for x in massiv:

if massiv[x-1] > max:

max = massiv[x]

return max

def 2mass(massiv):

massiv = massiv \* 2

return massiv

print(“Oddiy Python dasturi”)

a = [1,2,3,6,1,6]

kattasi (1,5)

r1 = max\_mass(a)

r2 = 2mass (a)

print(“Massiv max elementi –”, r1)

print(“Ikkilangan massiv – ”, r2)

Dastur natijasi:

Oddiy Python dasturi

5 katta 1 dan

Massiv max elementi -  6

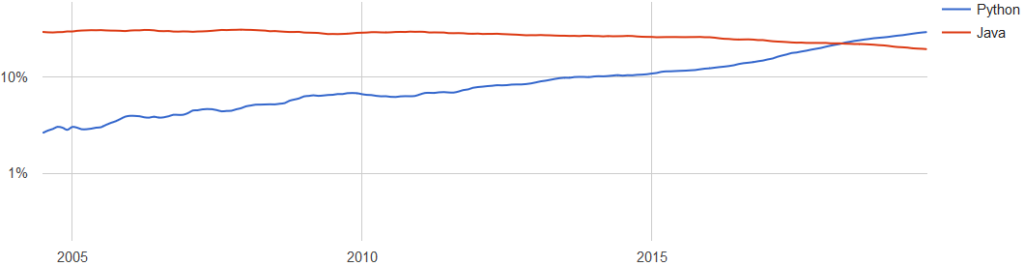
Ikkilangan massiv - [1, 2, 3, 6, 1, 6, 1, 2,3,6,1,6]

**Ommalashganligi.** Til 30 yoshdan oshgan bo‘lsada, butun dunyodagi dasturchilar orasida juda keng tarqalgan. Python deyarli har bir o‘rta yoki katta loyihada, asosiy ishlab chiqish vositasi sifatida bo‘lmasada, prototiplash yoki uning bir qismini yozish uchun vosita sifatida ishlatiladi.

U o‘z atrofida juda katta dasturchilar guruhini to‘pladi.

Stackoverflow saytida o‘tkazilgan so‘rov natijalariga ko‘ra, Python deyarli 39% ovoz bilan 7-o‘rinni egalladi.

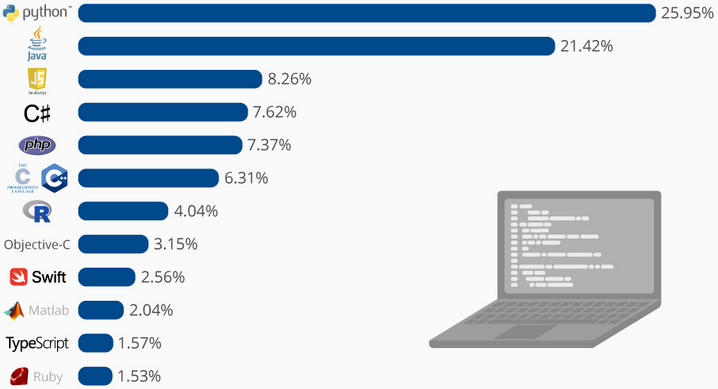
**PYPL.** Ushbu indeks tilni o‘rganish materiallari bilan bog‘liq qidiruv so‘rovlari soniga asoslanadi.



*2-rasm. PYPl indekmsi bo‘yicha solishtirma sxemasi*

PYPL ma’lumotlariga ko‘ra, Python mashhurligi bo‘yicha Javadan 10% ga oldinda.

**statista.com.** Xizmat turli xil statistik ma’lumotlarni taqdim etadi, ular orasida dasturlash tillarining mashhurligi bo‘yicha statistikalar ham mavjud.



*3-rasm.* ***Statista.com*** *saytidagi solishtirish diagrammasi*

**Ish tezligi.** Dasturchilar ko‘pincha o‘zlariga savol berishadi: "Python-dan foydalanish ishlashning pasayishiga olib keladimi?". Batafsil tekshiruvsiz hech qanday xulosa chiqarmang.

Agar biz faqat kodni bajarish tezligini hisobga olsak, Python C kabi boshqa dasturlash tillaridan past ekanligi ayon bo‘ladi. Darhaqiqat, dinamik terish, izohlash va dasturchi ishini osonlashtiradigan boshqa xususiyatlar ishlashning pasayishiga olib keladi. Biroq, zamonaviy IT-da nafaqat dasturlarning tezligi, balki ularni ishlab chiqish tezligi ham muhimdir. Ishlab chiqish, sinovdan o‘tkazish, disk raskadrovka va qo‘llab-quvvatlash - bularning barchasi juda ko‘p pul talab qiladi. Agar Python dasturlarining tezligi bo‘yicha past bo‘lishi mukin, ammo rivojlanish tezligida bo‘yicha unga teng keladigani yo‘q.

**Dasturchilar Pythonda yaratilgan dasturlarni bajarilish tezligini oshirish uchun turli yo‘llardan foydalanadilar:**

* **C kodini o‘rnatish** . Ushbu texnikadan foydalanib, siz ishlashni sezilarli darajada yaxshilashingiz mumkin, odatda vaqt birligida ko‘p so‘rovlarni qayta ishlaydigan kod bo‘limlari C tilida yoziladi. Masalan, bitta ma’lumotlar bazasidan ma’lumotlarni qabul qiladigan, uni qayta ishlaydigan va boshqasiga yuboradigan funksiyalar, agar o‘tadigan ma’lumotlarning miqdori yetarlicha katta bo‘lsa, C tilida yozilgani afzal xisoblanasi.
* **ifodalarni optimallashtirish.** Python dasturlarining tezligi ifodalarning ishlash tezligiga juda bog‘liq. Quyidagi jadvalda tezroq va sekinroq ishlovshi ifodalar keltirilgan.

1-jadval

| **Tezroq** | **Sekinroq** |
| --- | --- |
| a, b = c, d | a = c; b = d |
| a < b < c | a < b and b < c |
| not not a | bool(a) |
| a = 5 | a = 2 + 3 |

* **Dasturni testlash uchun tayyor modullar.** Kodning qaysi bo‘limlari umumiy ish faoliyatini sezilarli darajada kamaytirishini aniqlash uchun dasturchi testlash uchun maxsus modullardan foydalanishi mumkin. Shunday qilib, bu modullar yordamida qaysi kodni optimallashtirish yoki C kodi bilan almashtirish kerakligini bilib olishi mumkin.
* **Tayyor asboblar.** Aksariyat vazifalar uchun samarali echimlar allaqachon ishlab chiqilgan. O‘z yechimingizni noldan yozishdan ko‘ra, ba’zi kutubxonaning tayyor, tuzatilgan kodini ishlatish yaxshiroqdir, bu 100% samarali bo‘lmaydi.

**Pythonda qanday dasturlar yozish mumkin?**

Python tilida turli sferalarda dasturlar tuzish mumkin.

**Back-end – saytning dasturiy qismi.** Saytning server tomonini rivojlantirish uchun turli fremworklar qo‘llaniladi: **Django** va **Flask**. Bu fremworklar Pythonni boshqa mashhur vositalar bilan raqobatlashadigan imkoniyatlarga ega server tomonidagi dasturlash tiliga aylantiradilar.

PHP tili server tomonidagi web ishlab chiqish bozorining ko‘p qismini nazorat qilsada, tobora ko‘proq dasturchilar Pythonda ishlab chiqishni afzal ko‘rishmoqda.

**Blokcheyn.** Blokcheyn - bu ketma-ket bloklar zanjiri bo‘lib, unda har bir blok ma’lumotni o‘z ichiga oladi va har doim oldingisiga ulanadi. Texnologiya har qanday sohada qo‘llanilishi mumkin. Ayniqsa moliya sektorida va bitcoin kriptovalyutasi sohasida mashhurdir.

Blokcheyn axborot xavfsizligi va ochiqligini o‘zida mujassamlashtiradi. U foydalanuvchiga dunyoning istalgan nuqtasidan ma’lumotlarga kirish imkonini beradi.

**Bot.** Bu ma’lum bir vaqtda yoki berilgan signalga javoban ba’zi harakatlarni avtomatik ravishda bajaradigan dasturdir. Botlar inson xatti harakatlarini bevosita taqlid qilishi mumkin. Shuning uchun ular ko‘pincha texnik yordam ko‘rsatishda (chat botlarida), Internetda ma’lumot qidirishda (qidiruv botlarida), virtual dunyoda odam yoki boshqa mavjudotning harakatlarini taqlid qilishda (kompyuter o‘yinlari) zarur bo‘ladi.

Python sizga tezkor xususiyatga ega va nisbatan aqlli botlarni yaratish imkonini beradi. Shuni tushunish kerakki, botlar 500 qatorli kodlardan iborat oddiy dastur emas. Biznes uchun bot yaratish buyurtmasi bir necha millionga tushishi mumkin. Narx insondan farqlash qiyin bo‘ladigan botni loyihalash juda qiyin ekanligi bilan bog‘liq. Muloqotning ko‘plab variantlarini ta’minlash, inson xatti-harakatlarining omillarini tahlil qilish va ularni dasturda amalga oshirish kerak. Oddiy qilib aytganda, faqat nol va birlarni tushunadigan mashinadan siz ibtidoiy "miya" qilishingiz kerak bo‘ladi.

**Malumotlar bazasi.** Ma’lumotlar bazasi umumiy xususiyatlar va maxsus qoidalarga muvofiq tizimlashtirilgan ma’lumotlardir. Har qanday yirik loyihada ma’lumotlar bazalaridan foydalaniladi. Ular foydalanuvchilar haqidagi ma’lumotlarni, dasturdagi o‘zgarishlarni va hokazolarni saqlaydi.

Ma’lumotlar bazasini boshqarish tizimi Pythonda yozilishi mumkin.

**Kengaytirilgan haqiqat (Дополненная реальность).** “To‘ldirilgan reallik” virtual texnologiyalar yordamida jismoniy dunyoni to‘ldiradi. Ya’ni, virtual ob’ektlar real muhitga proyeksiya qilinadi va oddiy jismoniy ob’ektlarning xususiyatlari va xatti harakatlariga taqlid qiladi.

“Kengaytirilgan haqiqat”ni turli 3D filmlarda guvohi bo‘lish mumkin. Haqiqiy dunyoda u, masalan, jangovar jangchilarda (maqsad tizimi) ishlatiladi.

“Kengaytirilgan haqiqat” ishi teglar bilan o‘zaro ta’sirga asoslangan. Elektron qurilma ma’lumot oladi va atrofdagi makonni tahlil qiladi, kompyuter ko‘rish yordamida u odamning oldida nimani ko‘rayotganini “Tushunadi”. Keyin qurilma real dunyoda “Virtual qatlam”ni qoplaydi.

“Kengaytirilgan haqiqat” sferasida yaratilgan mukammal dasturlar taxminan 7-10 ming AKSH dollarini tashkil qiladi. Ularni loyihalash va yozish oson emas, 3D-dizaynerlardan dasturchilargacha turli mutaxassislar ishlab chiqish jarayonida tinmasdan mehnat qilishadi.

Python – bunday sferadagi loyihalarni yaratish uchun ajoyib tanlov.

**BitTorrent mijozi.** BitTorrent - bu Internet orqali katta hajmdagi ma’lumotlarni tezlikda uzatish, qabul qilish imkonini beruvchi noyob texnologiya.

BitTorrentning 6-versiyasidan keyingi versiyalari butunlay Pythonda yozilgan. Keyinchalik u C++ da butunlay qayta yozilgan bo‘lsa-da, bu Pyton tilidan xuddi shunga o‘xshash vazifalarni bajarish uchun foydalanish mumkinligini anglatadi.

**Neyron tarmoq.** “Neyron tarmoq” tushunchasi dasturlashga biologiyadan kirib kelgan. Biologiyada neyron tarmoq bir-biriga bog‘langan neyronlar ketma-ketligidir. Dasturiy ravishda yaratilgan neyron tarmoqlar nafaqat ma’lumotni tahlil qilish va saqlash, balki uni xotira qayta ishlab chiqishga qodir.

Ular inson miyasi tomonidan bajariladigan hisob-kitoblarni talab qiladigan murakkab masalalarni hal qilish uchun ishlatiladi. Odatda, neyron tarmoqlar biror narsani xususiyatlari bo‘yicha tasniflash, bashorat qilish, masalan, fotosurat yoki videodan odamni tanib olish kabi funksiyalarni amalga oshirish uchun foydalaniladi.

**Python neyron tarmoqlarni rivojlantirishda yaqqol yetakchi hisoblanadi. Standart vositalardan tashqari, u *“Mashinani o‘rganish”* sohasida juda ko‘p kutubxonalar mavjud. Buning yordamida hatto katta va murakkab loyihalar nisbatan tez yozilishi mumkin.**

**Parser -** bu ma’lumotlarni yig‘ish va qayta ishlash uchun mo‘ljallangan dastur. Foydalanuvchi valyuta kursi kabi ma’lumotlarni tahlil qilishi yoki turli kompaniyalar aktsiyalaridagi o‘zgarishlarni kuzatishi va tahlil qilishi mumkin.

Parser turli xil tillarda yozilishi mumkin. Ya’ni parser dasturlar boshqa dasturlash tillari yordamida ham yaratish mumkin. Lekin parser (ma’lumot yig‘uvchi) dasturlarni python tilida boshqa dasturlash tillariga qaraganda tez va samarali yozish mumkin.

**O‘yin yaratish.** Pythonda katta o‘yinlar yaratilmaydi. U prototipni (demo versiyasini) ishlab chiqish yoki ba’zi bir qismini amalga oshirish uchun ishlatiladi (masalan, o‘yinning server qismidagi logik qismi).

**Kichik loyihani yozish uchun siz kichik 2D o‘yinni yaratish uchun barcha kerakli vositalarni taqdim etadigan Pygame kutubxonasidan foydalanishingiz mumkin.**

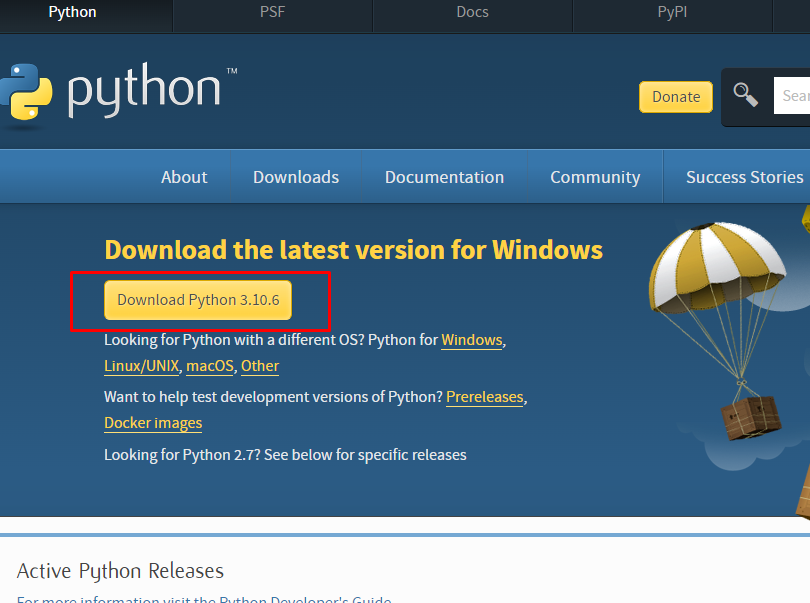
**2. Pythonni o‘rnatish. PyCharm dasturini o‘rnatish**

**PyCharm** - JetBrains tomonidan ishlab chiqilgan kross-platforma muharriri. Pycharm samarali Python rivojlanishi uchun zarur bo‘lgan barcha vositalarni taqdim etadi . Quyida Windows-da Python va PyCharm-ni o‘rnatish bo‘yicha batafsil ko‘rsatmalar mavjud.

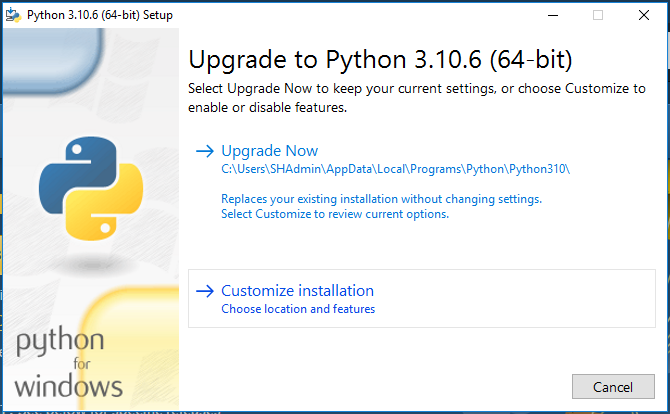
**Python o‘rnatilmoqda**

**1-qadam**. Python-ni yuklab olish va o‘rnatish uchun rasmiy Python veb-saytiga tashrif buyuring  //www.python.org/downloads/  va tegishli versiyani tanlang. Biz Python 3.10.6 versiyasini tanladik

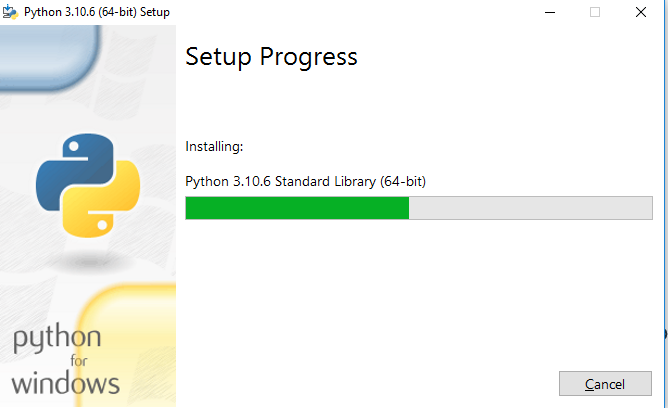




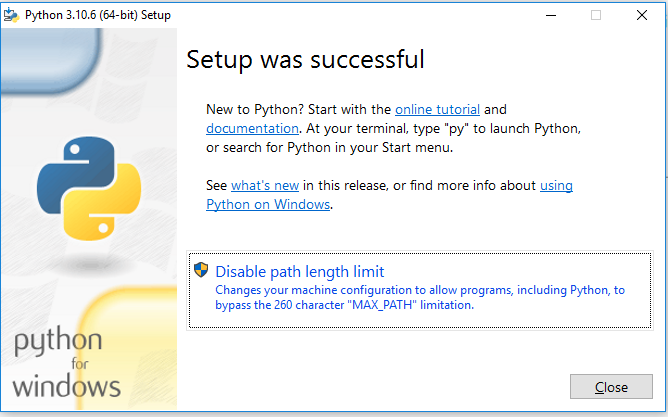
**2-qadam**  Yuklab olish tugallangach, Python-ni o‘rnatish uchun .exe faylini ishga tushiring. Keyin "Hozir o‘rnatish" tugmasini bosing.



**3-qadam**  Keyingi bosqichda Python o‘rnatilishi jarayonini ko‘rishingiz mumkin.

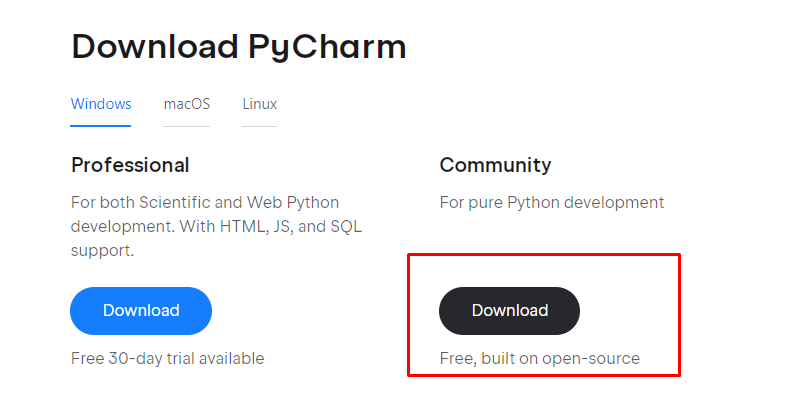


**4-qadam**  O‘rnatish tugallangach, o‘rnatish muvaffaqiyatli bo‘lganligi haqida panelni ko‘rasiz. Endi "Yopish" tugmasini bosing.

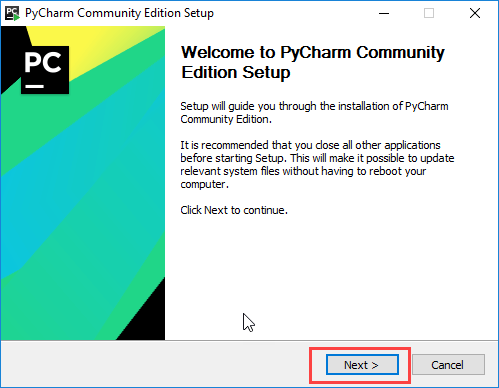


**Pycharm o‘rnatilmoqda**

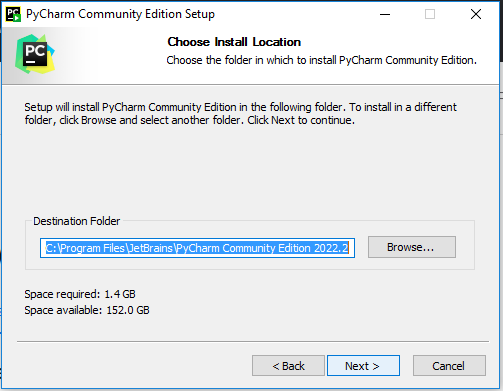
**1-qadam**  PyCharm-ni yuklab olish uchun //www.jetbrains.com/pycharm/download/ veb-saytiga tashrif buyuring va jamoat bo‘limidagi YUKLASH havolasini bosing.



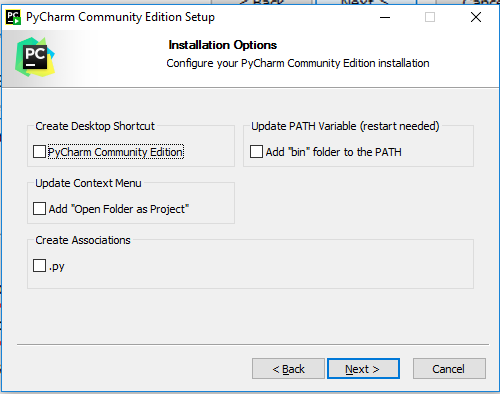
**2-qadam**  Yuklab olish tugallangach, PyCharm-ni o‘rnatish uchun .exe faylini ishga tushiring. O‘rnatish ustasi ishga tushishi kerak. "Keyingi" tugmasini bosing.



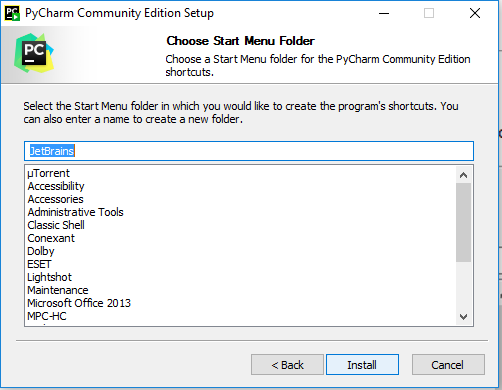
**3-qadam**  Keyingi bosqichda, agar kerak bo‘lsa, o‘rnatish yo‘lini o‘zgartiring. "Keyingi" tugmasini bosing.



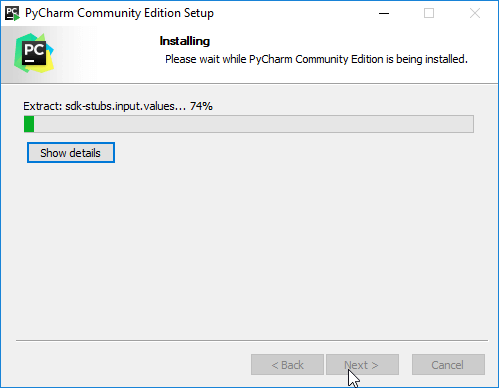
**4 -qadam)** Keyingi bosqichda, agar xohlasangiz, ish stolida yorliq yaratishingiz mumkin, so‘ngra "Next" tugmasini bosing..



**5-qadam)**  Boshlash menyusi papkasini tanlang. JetBrains-ni sukut bo‘yicha qoldiring va "O‘rnatish" tugmasini bosing.



**6-qadam)**  O‘rnatish tugashini kuting.



**7-qadam)**  O‘rnatish tugallangandan so‘ng siz PyCharm o‘rnatilganligi haqida xabar olasiz. Agar siz uni ishga tushirishni xohlasangiz, avval "PyCharm Community Edition-ni ishga tushirish" katagiga belgi qo‘ying va "Finish" tugmasini bosing.

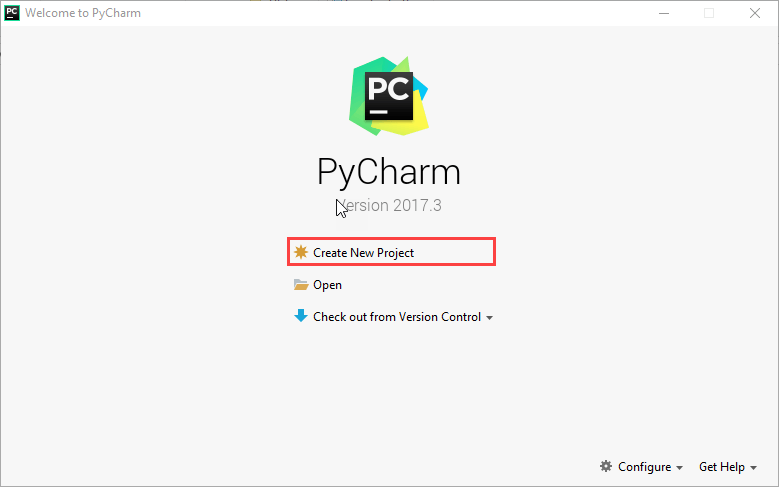


**8-qadam)**  Finish tugmasini bosganingizdan so‘ng quyidagi ekran paydo bo‘ladi.



**3. Pythonda “ Salom dunyo !” dasturini yaratish.**

**1-qadam)**  PyCharm muharririni oching. PyCharm xush kelibsiz panelini ko‘rasiz. Yangi loyiha yaratish uchun "Yangi loyiha yaratish" tugmasini bosing.

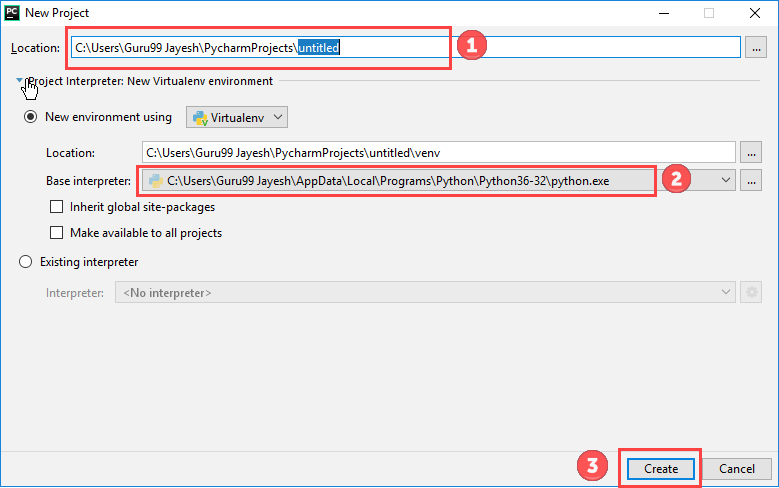


**2-qadam)**  Siz o‘rindiqni tanlashingiz kerak.

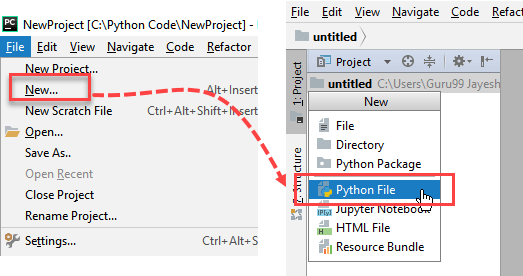
Siz loyihani yaratmoqchi bo‘lgan joyni tanlashingiz mumkin. Agar siz joylashuvni o‘zgartirishni xohlamasangiz, uni avvalgidek qoldiring, lekin hech bo‘lmaganda "nomsiz" nomini "Birinchi loyiha" kabi mazmunliroq narsaga o‘zgartiring.

PyCharm siz avval o‘rnatgan Python tarjimonini topgan bo‘lishi kerak edi.

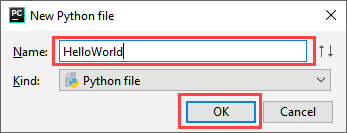
Keyin, "Yaratish" tugmasini bosing.



**3-qadam)**  Endi "Fayl" menyusiga o‘ting va "Yangi" ni tanlang. Keyin, "Python fayli" ni tanlang.



**4-qadam)**  Yangi qalqib chiquvchi oyna paydo bo‘ladi. Endi fayl nomini kiriting (bu erda biz "HelloWorld" ni o‘rnatamiz) va "OK" tugmasini bosing.



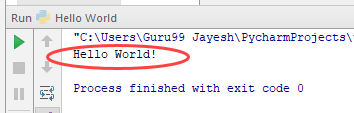
**5-qadam)**  Endi oddiy dastur yozing - print("Salom Dunyo!").



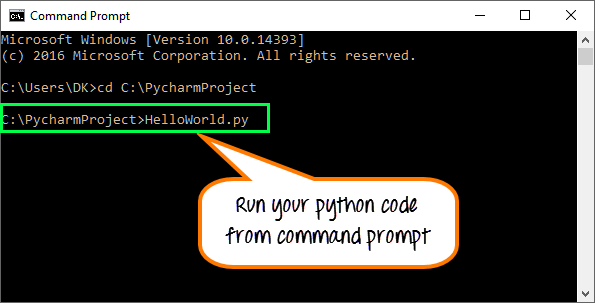
**6-qadam)**  Endi Run menyusiga o‘ting va dasturingizni ishga tushirish uchun Run-ni tanlang.



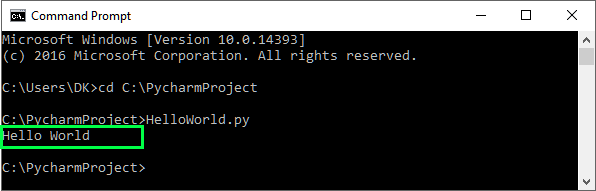
**7-qadam)**  Dasturingizning chiqishini ekranning pastki qismida ko‘rishingiz mumkin.



**8-qadam)**  Agar sizda Pycharm Editor o‘rnatilmagan bo‘lsa, tashvishlanmang, kodni buyruq satridan ishga tushirishingiz mumkin. Dasturni ishga tushirish uchun buyruq satriga to‘g‘ri fayl yo‘lini kiriting.



Kodning chiqishi quyidagicha bo‘ladi:



**9-qadam)**  Agar siz hali ham dasturni ishga tushira olmasangiz, bizda siz uchun Python muharriri mavjud. Iltimos, ushbu kodni   ishga tushiring.

**4. Python tilining asosiy operatorlari bilan tanishish**

Python dasturlash tilida yozilgan dastur ifodalar jamlanmasidan iborat. Har bir ifoda yangi qatorga joylashtiriladi. Masalan:

print(2 + 3)

print("Hello")

Python dasturlash tili kodida ifodalarning oldidan qoldirilgan oraliq masofalar katta ro‘l o‘ynaydi. Agar dasturchi tomonidan bilmasdan yoki chiroy uchun qoldirilgan har qanday oraliq masofa xatolik kelib chiqishiga sabab bo‘lishi mumkin. Masalan, garchi kod yuqoridagi bilan deyarli bir xil bo‘lsada, quyidagi holatda dasturda oraliq masofalar bilan bog‘liq xatolik sodir bo‘ladi:

print(2 + 3)

print("Hello")

Shuning uchun, python dasturlash tilida dastur kodining boshi satrning boshidan boshlanishi kerak. Bu Python bilan C# yoki Java kabi boshqa dasturlash tillari o‘rtasidagi muhim farqlardan biridir.

Ammo shuni yodda tutish kerakki, tilning ba’zi operatorlarining sintaksisi bir necha qatordan iborat bo‘ladi. Masalan, shart operatori if:

if 1 < 2:

print("salom")

Bu dasturda, agar 1 soni 2 dan kichik bo‘lsa, u holda “**salom**” so‘zi ekranga chiqariladi. Bu dasturda print(“Salom”) ifodasining oldida bo‘sh joy bo‘lishi kerak, chunki bu ifoda if shartli konstruktsiyasining bir qismi sifatida ishlatiladi.

Bundan tashqari, kodni yozish mobaynida qoldirib ketiladigan bo‘shliqlar 4 ga karrali probellar soniga (ya’ni 4, 8, 16 va boshqalar) teng bo‘lishi maqsadga muvofiqdir. Ammo 4 ga karrali bo‘lmay qolishi xatolik keltirib chiqarmaydi. Ifoda oldida qoldiriladigan masofalarning 4 ta probelga yoki 4ga karrali probellar soniga teng bo‘lishining sababi shundaki, 1 ta tab 4 ta probelga tengligida.

**Harflar katta-kichikligini farqlanishi**

**(Регистрозависимость)**

Python katta-kichik harflarni turli belgilan sifatida qabul qiladigan tildir. Shuning uchun **print** va **Print** yoki **PRINT** ifodalari turli ifodalarni anglatadi. Shunday ekan agar konsolga “Salom dunyo” xabarini Print funksiyasi orqali chiqarmoqchi bo‘lsak, hech qanday narsa chiqara olmaymiz, aksincha xatolik sodir bo‘ladi:

Print("Hello World")

**Izohlar**

Izohlar – kodning ma’lum bir qismida nima amalga oshirilayotganligini ifodalab foydalanuvchi uchun tushunchalar berib ketish uchun foydalaniladi. Dastur interpritatsiya qilinganda (ishga tushirilganda) interpritator izohlarni e’tiborsiz qoldiradi, shuning uchun ular dasturning ishlashiga ta’sir qilmaydi. Pythondagi izohlar blok va inline izohlarda keladi.

Bitta satrda joylashuvchi izohlardan oldin panjara belgisi- # qo‘yiladi:

# Konsolga chiqish

# Salom Dunyo xabarlari

print("Salom Dunyo ")

# belgisidan keyingi har qanday belgilar to‘plami izohni bildiradi. Ya’ni, yuqoridagi misolda kodning birinchi ikki qatori izohlar hisoblanadi.

Shuningdek izohlar ifodalardan so‘ng, til ifodalari bilan bir qatorda joylashgan bo‘lishi ham mumkin:

print("Salom dunyo")   # Xabarni konsolga chop etish

Blok izohlarida sharh matnidan oldin va keyin uchta bitta tirnoq qo‘yiladi: *‘‘‘sharh matni’’’*. Masalan:

‘‘‘

    Konsol chiqishi

    Salom dunyo xabarlari

‘‘‘

print ("Salom dunyo")

**Asosiy funksiyalari**

Python ko‘plab o‘rnatilgan, standart funktsiyalarni taqdim etadi. Ulardan ba’zilari, ayniqsa, til o‘rganishning dastlabki bosqichlarida juda tez-tez qo‘llaniladi, shuning uchun ularni ko‘rib chiqaylik.

  Konsolga ma’lumot chiqarishning asosiy funksiyasi - bu print() funksiyasi hisoblanadi. Biz chiqarmoqchi bo‘lgan satr ushbu funktsiyaga argument sifatida uzatiladi:

print ("Salom Python")

Agar konsolga bir nechta qiymatlarni chop etish kerak bo‘lsa, ularni vergul bilan ajratib, print() funktsiyasiga beriladi:

print("To‘liq ismi:", "Tom", "Smit")

Natijada, vergul bilan ajratilgan joylarga probel bilan ajratilgan holatda, barcha berilgan qiymatlar, bir qatorga joylashtirilib konsolga chiqariladi:

To‘liq ismi: Tom Smit

Agar **print()** funksiyasi chiqarish uchun javobgar bo‘lsa, u holda **input()** funktsiyasi axborotni kiritish uchun xizmat qiladi. **input()** funksiyasi kiritilgan ma’lumotni satr tipida qaytaradi va bu satrni o‘zgaruvchiga saqlanadi va shu qiymatga o‘zaruvchi orqali murojaat qilinadi:

name = input("Ismni kiriting:")

print ("Salom", ism)

Konsol chiqishi:

Ismni kiriting: Eugene

Salom Eugene

**Nazorat savollari:**

1. Python paketini qanday o‘rnatish mumkin ?

2. Python kodini oddiy matnli faylga saqlash orqali ishga tushirish uchun qanday qadamlarni bajarishim kerak ?

3. Python dasturlash tili bilan ishlash uchun qanday tarjimonlardan foydalanish mumkin ?

4. Python dasturlash tilining nomi qayerdan kelib chiqqan?

5. Python dasturlash tilining tarixi haqida gapirib bering.

6. Python dasturlash tili logotipini kim ixtiro qilgan.

7. Pythonni oddiy va tushunarli dasturlash tili nima qiladi?

8. Pythonning boshqa dasturlash tillaridan qanday afzalliklari bor?

9. Python dasturlash tili yordamida qanday dasturlar yaratish mumkin?

10.Pythonda dastur strukturasi qanday tuzilgan?

11.Satrning oldida qoldirilgan oraliq masofalar (indents) nimani anglatadi?

12.Pythonning sintaksisi boshqa dasturlash tillarining sintaksisidan eng katta farqli jihati nimada?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**2-mashg‘ulot.** Pythonda arifmetik operatorlar.

**O‘quv savollari:**

1. Arifmetik operatorlar.
2. Sonlar bilan ishlash. O‘zgaruvchilar.
3. Bool tipli ma’lumotlar bilan ishlash.

**1. Arifmetik operatorlar.**

Python barcha turdagi arifmetik operatsiyalarni qo‘llab-quvvatlaydi :

**Qo‘shish (+):** print(6 + 2)   # 8

**Ayirish (-):** print(6 - 2)   # 4

**Ko‘paytirish (\*):** print(6 \* 2)   # 12

**Bo‘linish (/):** print(6/2)   # 3.0

**Bo‘lib butunni olish (//):** print(7 // 2)  # 3

**Darajaga oshirish (\*\*):** print(6 \*\* 2)  # 62 natija 36

**Bo‘lib qoldig‘ini olish (%):** print(7 % 2) # 1

Agar bitta arifmetik ifodada bir nechta arifmetik amallar ketma-ket foydalanilsa, amallarning prioritetiga muvofiq ketma-ketlikda bajariladi. Birinchi navbatda yuqori ustuvorlikdagi operatsiyalar amalga oshiriladi. Arifmetik amallarning prioritetlarining kamayish tartibida quyidagi 2-jadvalda ko‘rsatilgan.

Operatsiya   yo‘nalishi:

2-jadval

|  |  |  |
| --- | --- | --- |
| 1 | \*\* | o‘ngdan chapga |
| 2 | \*, /, //, % | Chapdan o‘ngga |
| 3 | +, - | Chapdan o‘ngga |

Keling, quyidagi ifodani ko‘rib chiqamiz:

raqam = 3 + 4 \* 5 \*\* 2 + 7

print(raqam)   # 110

Bu yerda birinchi navbatda darajaga oshirish amali (5 \*\* 2) bajariladi, keyin natija 4 ga (25 \* 4) ko‘paytiriladi, keyin qo‘shish (3 + 100) amali bajariladi va keyin yana qo‘shish (103 + 7) amalga oshiriladi. Shunday qilib natija 110 ga teng bo‘ladi.

 Qavslar amallar tartibini qayta aniqlash uchun ishlatilishi mumkin:

raqam = (3 + 4) \* (5 \*\* 2 + 7)

print(raqam)   # 224

Shuni ta’kidlash kerakki, arifmetik amallarda ham butun, ham kasr sonlar ishtirok etishi mumkin. Agar butun son (int) va vergulli (kasr) son (float) bir xil amalda ishtirok etsa, u holda butun son float turiga o‘tkaziladi.

**Arifmetik amallarning tenglik belgisi bilan ishlatilishi**

Bu maxsus operatsiyalar amal bajarilgandan keyin natijani birinchi operandga yuklash imkonini beradi:

3-jadval

|  |  |
| --- | --- |
| **+=** | Qo‘shishdan hosil bo‘lgan natijani o‘zlashtirish |
| **-=** | Ayirishdan hosil bo‘lgan natijani o‘zlashtirish |
| **\*=** | Ko‘paytirishdan hosil bo‘lgan natijani o‘zlashtirish |
| **/=** | Bo‘linishdan hosil bo‘lgan natijani o‘zlashtirish |
| **//=** | Bo‘lib butunni olishdan hosil bo‘lgan natijani o‘zlashtirish |
| **\*\*=** | Raqam darajaga oshirishdan hosil bo‘lgan natijani o‘zlashtirish |
| **%=** | Bo‘lib qoldig‘ini oshishdan hosil bo‘lgan natijani o‘zlashtirish |

**Operatsiyalarni dasturda qo‘llanilishi:**

raqam = 10

raqam += 5

print(raqam)   # 10 + 5 = 15

raqam -= 3

print(raqam)   # 15 - 3 = 12

raqam \*= 4

print(raqam)   # 12 \* 4 = 48

raqam /= 2

print(raqam)   # 48 / 2 = 24.0

raqam //= 5

print(raqam)   # 24 // 5 = 4

raqam \*\*= 3

print(raqam)   # 4 \*\* 3 = 64

raqam %= 6

print(raqam)   # 64 % 6 = 4

**Yaxlitlash va “round” funksiyasi**

Dasturlashda float tipli raqamlar bilan operatsiyalar bajarayotganda, natijalar aniq biz o‘ylaganimiz kabi bo‘lmasligi mumkin. Masalan :

bir\_son = 2.0001

ikki\_son = 5

uch\_son = bir\_son / ikki\_son

print(uch\_son) # 0.40002000000000004

Bunday holda, dasturchi 0,40002 kabi natijani olishni kutdi, lekin oxirida bir qator nollar orqali, qandaydir to‘rt paydo bo‘lmoqda. Yoki boshqa ifoda:

print(2.0001 + 0.2)   # 2.1002000000000003

Yuqoridagi holatda, natijani yaxlitlash uchun o‘rnatilgan round() funktsiyasidan foydalanishimiz mumkin:

bir\_son = 2.0001

ikki\_son = 0.1

uch\_son = bir\_son + ikki\_son

print(round(uch\_son))  # 2

Yaxlitlanadigan raqam round() funktsiyasiga o‘tkaziladi. Agar yuqoridagi misoldagi kabi funksiyaga bitta raqam uzatilsa, u butun songa yaxlitlanadi.

round() funktsiyasi ikkinchi raqamni ham olishi mumkin, natijada olingan sonda nechta o‘nli kasr bo‘lishi kerakligini ko‘rsatadi:

bir\_son = 2.0001

ikki\_son = 0.1

uch\_son = bir\_son + ikki\_son

print(round(uch\_son, 4))  # 2.1001

Bu holda uch\_son verguldan keying 4 ta raqamgacha yaxlitlanadi.

Agar funktsiyaga faqat bitta qiymat o‘tkazilsa - faqat yaxlitlanadigan raqam, u eng yaqin butun songa yaxlitlanadi.

Yaxlitlash misollari:

# butun songacha yaxlitlash

print(round(2.49))

print(round(2.51))

Biroq, agar yaxlitlanadigan qism ikkita butun sondan bir xil masofada bo‘lsa, yaxlitlash eng yaqin juft songa o‘tadi:

print(round(2.5))    # 2

print(round(3.5))    # 4

**round()** funksiyasining ikkinchi parametrini beradigan bo‘lsak, verguldan keyin shuncha xonagacha yaxlitlaydi. Yaxlitlagandayam tushirib qoldiriladigan raqam 5 dan katta yoki teng bo‘lsa, unda oldingisiga birni qo‘shib yaxlitlaydi:

# verguldan keyingi 2 ta raqamgacha yaxlitlash

print(round (2.554, 2))       # 2.55

print(round (2.5551, 2))       # 2.56

print(round (2.554999, 2))    # 2.55

print(round (2.499, 2))       # 2.5

Biroq, round() funksiyasi ideal vosita emasligini yodda tuting. Masalan, yuqorida, butun sonlarga yaxlitlashda, agar yaxlitlanadigan qism ikki qiymatdan bir xil masofada bo‘lsa, yaxlitlash eng yaqin juft qiymatgacha amalga oshiriladi, degan qoida qo‘llaniladi. Pythonda, sonning kasr qismini suzuvchi raqam sifatida to‘g‘ri ko‘rsatib bo‘lmasligi sababli, bu butunlay kutilmagan natijalarga olib kelishi mumkin. Masalan:

# verguldan keyingi 2 ta raqamgacha yaxlitlash

print(dumaloq (2.545, 2))    # 2.54

print(dumaloq(2.555, 2))  # 2.56

print(dumaloq (2.565, 2))    # 2.56

print(dumaloq (2.575, 2))    # 2.58

print( dumaloq(2.655, 2))    # 2.65

print(dumaloq (2.665, 2))    # 2.67

print(dumaloq (2.675, 2))    # 2.67

**2. Sonlar bilan ishlash. O‘zgaruvchilar**

**O‘zgaruvchilar**

O‘zgaruvchilar qiymatlarni saqlash uchun mo‘ljallangan dastur elementi. Python tilidagi o‘zgaruvchi nomi alifbo belgisi yoki pastki chiziq bilan boshlanishi kerak va alfanumerik belgilar va pastki chiziqni o‘z ichiga olishi mumkin. Bundan tashqari, o‘zgaruvchining nomi Python tilidagi kalit so‘zlarning nomi bilan bir xil bo‘lmasligi kerak. Kalit so‘zlar ko‘p emas, ularni eslab qolish mumkin:

*False await else import pass*

*None break except in raise*

*True class finally is return*

*and continue for lambda try*

*as def from nonlocal while*

*assert del global not with*

*async elif if or yield*

O‘zgaruvchilarni e’lon qilish quyidagicha amalga oshiriladi:

foydalanuvchi = "Tomson"

Bu yerda “foydalanuvchi” nomli o‘zgaruvchiga "Tom" satri o‘zlashtirilmoqda.

Dasturlash tillarida o‘zgaruvchilarni nomlashning ikki xil usuli mavjud: ***camel case*** va ***underscore notation***.

**Camel case** o‘zgaruvchi nomidagi har bir keying so‘z bosh harf bilan boshlanadi va oldingi so‘zga biriktirib yoziladi. Masalan:

userName = "Tomson"

**Underscore notation** – buusulda o‘zgaruvchi nomidagi so‘zlar pastki chiziq bilan ajratiladi. Masalan:

user\_name = "Tomson"

Shuningdek, siz katta-kichik registr sezgirligini hisobga olishingiz kerak, shuning uchun **ism** va **Ism** o‘zgaruvchilari turli ob’ektlarni ifodalaydi.

# ikki xil oʻzgaruvchi

ism = "Jenifer"

Ism = "Loktfud"

O‘zgaruvchilarning o‘ziga xos xususiyati shundagi dastur davomida uning qiymati va tipi o‘zgartirlishi mumkin.

Quyida o‘zgaruvchini aniqlab, uni dastur davomida qiymatidan foydalanishni ko‘rib chiqamiz:

name = "Tomson"  # o‘zgaruvchi nomi "Tomson" ga teng

print(name)     # print: Tomson

name = "Bobbi"   # qiymatni "Bobbi" ga o‘zgartiring

print(name)     # chiqish: Bobbi

**Ma’lumotlar turlari**

O‘zgaruvchi ma’lumotlar turlaridan birining ma’lumotlarini saqlaydi. Pythonda juda ko‘p turli xil ma’lumotlar turlari mavjud. Shulardan eng asosiy turlarni ko‘rib chiqamiz: **bool**, **int**, **float** va **str**.

**bool (Mantiqiy qiymatlar).** bool turli o‘zgaruvchilar ikkita mantiqiy qiymatni qabul qiladi: True (rost yoki 1) yoki False (yolg‘on yoki 0). True qiymati biror narsaning haqiqat ekanligini ko‘rsatish uchun ishlatiladi. False qiymati esa, aksincha biror fikrning yoki narsa noto‘g‘ri ekanligini ko‘rsatadi. Ushbu turdagi o‘zgaruvchilardan foydalanishga misol:

isMarried = False

print(isMarried)     # False

isAlive = True

print(isAlive)       # True

**int (Butun sonlar).** int turi 1, 4, 8, 50 kabi butun sonlar to‘plamidagi sonlarni ifodalaydi. Ya’ni minus cheksizlikdan plus cheksizlikkacha bo‘lgan barcha butun sonlar to‘plamiga tushuvchi biror bir sonni qabul qiladi. dasturda foydalanishga misol:

age = 21

print("Yosh:", age)     # Yosh: 21

count = 15

print("Hisob:", count) # Hisob: 15

Odatiy bo‘lib, standart raqamlar o‘nlik sonlar sifatida qabul qilinadi. Ammo Python ikkilik, sakkizlik va o‘n oltilik tizimlardagi raqamlarni ham qo‘llab-quvvatlaydi.

Raqam ***ikkilik sanoq sistemasi***da ekanligini ko‘rsatish uchun raqam oldiga **0b** prefiksi qo‘yiladi:

a = 0b11

b = 0b1011

c = 0b100001

print(a)     # 3 o‘nlik sanoq sistemasida

print(b)     # 11 o‘nlik sanoq sistemasida

print(c)     # 33 o‘nlik sanoq sistemasida

Raqam ***sakkizlik sanoq sistemasi***ni ifodalashini ko‘rsatish uchun raqamga **00** prefiksi qo‘yiladi:

a = 007

b = 0011

c = 0017

print(a)     # 7 o‘nlik sanoq sistemasida

print(b)     # 9 o‘nlik sanoq sistemasida

print (c)    # 15 o‘nlik sanoq sistemasida

Raqam ***o‘n oltilik sanoq sistemasi***ni ifodalashini ko‘rsatish uchun raqamga **0x** prefiksi qo‘yiladi:

a = 0x0A

b = 0xFF

c = 0xA1

print(a)     # 10 o‘nlik sanoq sistemasida

print(b)     # 255 o‘nlik sanoq sistemasida

print(c)     # 161 o‘nlik sanoq sistemasida

Shuni ta’kidlash kerakki, qaysi tizimda biz konsolga chiqarish uchun bosib chiqarish funktsiyasiga raqam o‘tkazsak, u sukut bo‘yicha o‘nli kasrda chiqariladi.

**float (Kasr sonlar).** float turi vergulli sonini ifodalaydi va haqiqiy sonlar to‘pamiga kiruvchi biror-bir sonni qabul qiladi. masalan: 1,2 yoki 34,76 va hokazo. Butun va kasr qismlar orasidagi ajratuvchi sifatida nuqta ishlatiladi.

uzunligi = 1.68

p = 3.14

vazn = 68.

print (uzunligi)    # 1.68

print(p)        # 3.14

print(vazn)    # 68.0

kasr sonini eksponensial belgida aniqlash mumkin:

x = 3.9e3

print(x)   # 3900.0

x = 3.9e-3

print(x)   # 0.0039

Bu yerda x = 3.9e3 ifodadagi e3 = 103 ni anglatadi va 3.9\*103  ko‘payma x o‘zgaruvchisiga yuklanmoqda. O‘z navbatida e-3 = 10-3

**str (satrli tip).** str turi satrlarni ifodalaydi. Satr “salom” va ‘dunyo‘ kabi bitta yoki qo‘sh tirnoq ichiga olingan belgilar ketma-ketligini ifodalaydi. Python 3.x da satrlar Unicode belgilar to‘plamini ifodalaydi

hi = "Salom dunyo!"

print (hi )   # Salom dunyo!

ism = "Tomson"

print (ism)   # Tomson

**O‘zgaruvchan tip.** Python dasturlash tilida o‘zgaruvchilarning tipi dinamik tip hisoblanadi. Bu o‘zgaruvchining tiplari dastur davomida qiymatning tipiga bog‘liq tarzda o‘zgarishi mimkinligini anglatadi.

Shunday qilib, ikki yoki bitta tirnoq ichida satr tayinlanganda, o‘zgaruvchi str turida bo‘ladi. Butun sonni o‘zlashtirlganda Python avtomatik ravishda o‘zgaruvchining turini int sifatida aniqlaydi. O‘zgaruvchini float ob’ekti sifatida aniqlash uchun unga kasr son beriladi, bu yerda kasr va butun qismlarni ajratuvchi sifatida nuqtadan foydalaniladi. Quyida bunga misol keltirilgan:

f\_Id = "abc"   # satr turiga mansub

print (f\_Id)

f\_Id = 234   # butun sonlar turiga mansub

print (f\_Id)

Standart **type()** funksiyasidan foydalanib, o‘zgaruvchining joriy tipini bilib olishingiz mumkin:

f\_Id = "abc"       # tip str

print(type(f\_Id)) # <class 'str'>

f\_Id = 234         # tip  int

print(type(f\_Id)) # <class 'int'>

**3. Bool tipli ma’lumotlar bilan ishlash**

Python dasturlash tilida ham bir qancha mantiqiy ifodalar mavjud. Ushbu mantiqiy ifodalarning barchasi ikkita o‘zgaruvchi bilan ishlatiladi va umumiy natija sifatida bool turidagi mantiqiy qiymat qaytaradi. Mantiqiy qiymatlar ikkita bo‘lib, quyidagilar: **True** (rost) va **False** (yolg‘on).

**Solishtirish operatorlari.** Eng oddiy shartli ifodalar ikki qiymatni taqqoslaydigan taqqoslash amallaridir. Python quyidagi taqqoslash operatsiyalarini qo‘llab-quvvatlaydi:

4-jadval

|  |  |
| --- | --- |
| **==** | Ikkala operand teng bo‘lsa, True qiymatini qaytaradi. Aks holda False qaytaradi. |
| **!=** | Ikkala operand teng bo‘lmasa, True qiymatini qaytaradi. Aks holda False qaytaradi. |
| **>** (katta) | Agar birinchi operand ikkinchisidan katta bo‘lsa, True qiymatini qaytaradi. |
| **<** (kichik) | Agar birinchi operand ikkinchidan kichik bo‘lsa, True qiymatini qaytaradi. |
| **>=** (katta yoki teng) | Birinchi operand ikkinchidan katta yoki teng bo‘lsa, True qiymatini qaytaradi. |
| **<=** (kichik yoki teng) | Birinchi operand ikkinchidan kichik yoki teng bo‘lsa, True qiymatini qaytaradi. |

 Taqqoslash operatsiyalariga misollar:

a = 6

b = 9

res = 6 == 9  # natijani o‘zgaruvchiga yuklash

print(res)  # False

print(a > b)  # False

print(a != b)  # True

print(a < b)  # True

boolean1 = True

boolean2 = False

print(boolean1 == boolean2) # False

Taqqoslash operatsiyalari turli ob’ektlarni - satrlarni, raqamlarni, mantiqiy amallarni solishtirishi mumkin, ammo operatsiyaning ikkala operandlari ham bir xil turni ifodalashi kerak.

**Mantiqiy amallar.** Turli murakkablikka ega bo‘lgan mantiqiy ifodalarni yaratish uchun mantiqiy operatorlar ishlatiladi. Python quyidagi mantiqiy operatorlarni qo‘llab quvvatlaydi:

**and** (mantiqiy ko‘paytirish) operatori ikkita operandga qo‘llaniladi:

x and y

Birinchidan, and operatori x ifodasini baholaydi va agar u False bo‘lsa, uning qiymati qaytariladi. Agar u True bo‘lsa, ikkinchi operand y baholanadi va y ning qiymati qaytariladi.

yosh = 20

vazn= 65

res = yosh > 19 and vazn == 65

print(res) # True

Bu holda **and** operatori ikkita ifoda natijalarini taqqoslaydi: yosh > 19 hamda vazn == 65. Agar bu ifodalarning ikkalasi ham True qiymatini qaytarsa, unda **and** operatori ham res o‘zgaruvchisiga True qiymatini qaytaradi (rasmiy ravishda oxirgi operandning qiymati qaytariladi). .

Lekin **and** operatorining operandlari True va False bo‘lishi shart emas. Bu har qanday qiymat bo‘lishi mumkin. Masalan:

natija = 14 va "s"

print(natija)   # s, chunki 14 soni True ga ekvivalent,

#shuning uchun oxirgi operandning qiymati qaytariladi

natija = 0 va "s"

print(natija)   # 0 , chunki 0 False ga ekvivalent

Bunday holda, 0 raqami va bo‘sh "" qatori noto‘g‘ri deb hisoblanadi, qolgan barcha raqamlar va bo‘sh bo‘lmagan satrlar "True" ga ekvivalentdir.

**or (mantiqiy qo‘shish)** ikkita operandga ham tegishli:

x or y

**Or** operatori avval x ifodasini baholaydi va agar u True bo‘lsa, uning qiymati qaytariladi. Agar u False bo‘lsa, ikkinchi operand y baholanadi va y ning qiymati qaytariladi. Masalan

yosh = 22

natija = yosh > 21 or False

print(natija) # True

Shuningdek, **OR** operatori har qanday qiymatlarga qo‘llanilishi mumkin. Masalan:

res = 4 or "f"

print(res)   # 4, chunki 4 rost ga teng, shuning uchun birinchi #operandning qiymati qaytariladi

natija = 0 or "f"

print(natija)   # f, chunki 0 False ga ekvivalent, shuning uchun #oxirgi operandning qiymati qaytariladi

**not (mantiqiy inkor - !)**

Ifoda False bo‘lsa, True qiymatini qaytaradi

yosh = 25

isSingle = False

print(not yosh > 21)   # False

print(not isSingle)   # True

print(not 45)   # False

print(not 0)   # True

**in operatori**

Agar ba’zi qiymatlar to‘plamida ma’lum qiymat mavjud bo‘lsa, **in** operatori "True" ni qaytaradi. U quyidagi shaklga ega:

qiymat **in** [qiymat\_to‘plamidagi]

Masalan, satr belgilar to‘plamini ifodalaydi. Va in operatori bilan biz unda biron bir qiymat mos kelishini va mavjudligini tekshirishimiz mumkin:

xabar = "hello world!"

hi = "hello"

**print**(hi in xabar)

zol= "golden"

**print**(zol in xabar)

Agar qiymatlar to‘plamida qiymat mavjud masligini tekshirishimiz kerak bo‘lsa, unda biz **not in** modifikatsiyasidan foydalanishimiz kerak bo‘ladi. Agar qiymat [qiymatlar to‘plami] da mavjud bo‘lmasa, u True qiymatini qaytaradi:

xabar = "hello world!"

hi = "hello"

**print**(hi not in xabar)  # False

golden = " golden "

**print**(golden not in xabar)  # True

**Nazorat savollari:**

1. pythonda qanday arifmetik amallar mavjud?
2. % - qanday amalni bajarish uchun ishlatiladi?
3. // - qanday amalni bajarish uchun ishlatiladi?
4. \*\* - qanday amalni bajarish uchun ishlatiladi?
5. Kvadrat ildiz qanday ishlatiladi?
6. Oddiy bo‘lish amalini bajarilganda qanday tipli qiymat qaytaradi?
7. += , -=, \*=, /=, \*\*= va %= - operatorlarining vazifasi qanday?
8. Round() funksiyasining ishlash prinsipini tushuntirib bering.

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**3-mashg‘ulot.** Satrlar bilan ishlovchi operatorlar va metodlar.

**O‘quv savollari:**

1. Satrlar bilan ishlovchi operatorlar va metodlar.
2. str.format() metodi yordamida satrlarni formatlash.

**1. Satrlar bilan ishlovchi operatorlar va metodlar.**

Satr deb mavjud belgilar ketma-ketligiga aytiladi. Satrlarni pythonda aniqlash uchun ham bittalik, ham qo‘sh tirnoqlardan foydalaniladi. Masalan:

msg = ‘Salom Dunyo!’

**print**(msg)   # Salom Dunyo!

nomi="Jeyson"

**print**(nomi)   # Jeyson

Agar satrdagi simvollar soni ko‘p bo‘lsa, uni qismlarga bo‘lish va ularni turli xil kod satrlariga joylashtirish mumkin. Bunday holda, butun satr qavs ichiga olinadi va uning alohida qismlari tirnoq ichiga joylashtiriladi:

text = ("Salom hurmatli kursantlar"

"Bugun biz sizlar bilan"

"Satrlar ustida amallar bajarib ko‘ramiz"

)

**print**(text)

Agar ko‘p qatorli matnni hosil qilish kerak bo‘lsa, unda bunday matn ikki tomondan uch qo‘sh (""") yoki bittalik (‘‘‘) tirnoq ichiga olinib yoziladi:

‘‘‘

Bu izoh

‘‘‘

text = ‘‘‘Salom hurmatli kursantlar

Bugun biz sizlar bilan

Satrlar ustida amallar

bajarib ko‘ramiz

‘‘‘

print(text)

Faqat shuni esda tutish kerakki uchta qo‘shtirnoqdan izohlar uchun ham foydalaniladi. Agar uchta qo‘shtirnoq ichidagi matn o‘zgaruvchiga tayinlangan bo‘lsa, unda bu izoh emas, satr hisoblanadi. Izoh qoldirish uchun esa izoh bo‘lishi kerak bo‘lgan matnni uchta qo‘shtirnoqqa o‘rab olinadi holos.

**Satrdagi qo‘llaniladigan maxsus simvollar.** Satr bir qator maxsus belgilarni o‘z ichiga olishi mumkin. Ulardan ba’zilari quyida keltirilgan:

* **\:** satr tarkibiga slesh simbolini qo‘shish imkonini beradi
* **\’:** satr ichiga bitta tirnoq qo‘shish imkonini beradi
* **\":** qatorga qoʻsh tirnoq qoʻshish imkonini beradi
* **\n:** Yangi qatorga oʻtadi
* **\t:** Tabulyatsiya qo‘shadi (4 ta probelga teng bo‘sh joy)

Quyida ba’zi maxsus belgilardan satr ichida foydalanish bo‘yicha misollar keltirilgan:

text = "Xabar:\n\"Salom Dunyo\""

**print**(text)

Dasturning konsoldagi ko‘rinishi:

Xabar:

"Salom Dunyo"

Garchi bunday maxsus belgilar satrga bittalik yoki qo‘sh tirnoq qo‘yish, jadval tuzish, boshqa qatorga o‘tkazish kabi ko‘plab amallarni bajarishga imkon bersada, ba’zi hollarda ular dasturda xatolik keltirib chiqaradi. Masalan:

URL = "D:\python\_course\name.txt"

**print**(URL)

Bu yerda URL o‘zgaruvchisi faylga yo‘lni o‘zlashtirib oladi. Biroq, satr ichida "\n" maxsus belgisi mavjud bo‘lib, garchi dasturchi bu belgini faylga yo‘lni ko‘rsatish uchun foydalanayotgan bo‘lsada, python interpritatori uni maxsus operator sifatida tarjima qiladi. Shunday qilib satrni ekranga chiqarishda natija quyidagicha bo‘ladi:

c:\python\_course

ame.txt

Bunday vaziyatni oldini olish uchun satr oldiga r belgisi qo‘yiladi:

URL = r"C:\python\_course\name.txt"

**print**(URL)

c:\python\_course\name.txt

**Qatorga qiymatlarni kiritish**

Python boshqa o‘zgaruvchilarning qiymatlarini satrga joylashtirish imkonini beradi. Buning uchun satr ichida oʻzgaruvchilar jingalak qavslarga {} joylashtiriladi va satr oldidagi qo‘shtirnoq belgisi oldiga f harfi qoʻyiladi:

user\_name = "Jeyson"

user\_age = 22

user = f"Ismi: { user\_name } Yoshi: { user\_age }"

**print**(user\_age) # Ismi: Jeyson Yoshi: 22

Bunday holda, user\_name o‘zgaruvchisining qiymati {user\_name} o‘rniga kiritiladi. Xuddi shunday, {user\_age} oʻrniga user\_age oʻzgaruvchisining qiymati kiritiladi.

**Satr tarkibidagi belgilarga murojaat qilish.** Python dasturlash tilida ham boshqa dasturlash tillaridagi kabi satrning tarkibidagi simvollarga murojaat qilib ular ustida turli amallar bajarish mumkin. Satr tarkibidagi belgilarga murojaat qilish uchun satr\_nomi va kvadrat qavslar ichida simvolning indeksi kiritiladi. Indeks bu yerda simvolning satrdagi joylashgan o‘rni bo‘lib, simvollar satrda 0 dan boshlab indexlangan bo‘ladi. Demak satrdagi simvollar soni oxirgi simvolning indeksidan bittaga ortiq bo‘ladi. Quyida satr tarkibidagi belgilarga murojaat qanday amalga oshirilishiga misol keltirilgan:

str = "hello world!"

belgi = str[0] # h

**print**(belgi)

belgi = str[6] # w

**print**(belgi)

belgi = str[11] # error: IndexError: string index out of range

**print**(belgi)

Agar satrda mavjud bo‘lmagan indeksdagi elementga murojaat qilinsa, IndexError xatoligi kelib chiqadi. Misol uchun, yuqoridagi holatda, str nomli satr 11 ta belgidan iborat va belgilari 0 dan 10 gacha indekslarga ega. Shuning uchun str[11] ifodasi xatolik qaytardi. Chunki satrda 11 ta element va oxirgi elementning indeksi 10 ga teng.

  Satr oxiridan boshlab elementlarga murojjat qilish uchun manfiy indekslardan foydalanish mumkin. Shunday qilib, indeks -1 oxirgi belgini va indeks -2 oxirgi belgidan bitta oldingi ya’ni oxiridan 2- belgini ifodalaydi va hokazo. Masalan:

str = "salom dunyo!"

belgi = str[-1]

**print**(belgi) #!

belgi2 = str[-5]

**print**(belgi2) #u

Belgilar bilan ishlashda satr o‘zgarmas (immutable) tip ekanligini hisobga olish kerak, shuning uchun satrning har qanday individual belgisini o‘zgartirishga harakat qilsak, quyidagi holatda bo‘lgani kabi xatoga duch kelamiz:

str = "Salom DASTURCHILAR!"

str[1] = "R"

TypeError:'str' object does not support item assignment

Faqat satr qiymatini unga boshqa qiymat belgilash orqali butunlay qayta o‘rnatishimiz mumkin.

**Satrning barcha elementlarini sikl yordamida olish.** Satrdagibarcha belgilarni for tsiklini qo‘llash orqali olish mumkin:

str = "salom dunyo"

**for** belgi **in** str:

    print(belgi)

s

a

l

o

m

d

u

n

y

o

**Matndan belgilar ketma-ketligini olish.** Satr tarkibidan nafaqat bitta belgini, balki belgilar ketma-ketligini ham olish mumkin. Buning uchun quyidagi sintaksis qo‘llaniladi:

**str[:end]** – ushbu ifoda 0 - indeksdan to end - indeksgacha belgilar ketma-ketligini chiqaradi. end – elementni o‘zini olmaydi. End o‘rnida son beriladi. str[:10]

**str[start:end]** – belgilar ketma-ketligi **start** dan to **end** - indeksigacha bo‘lgan belgilar ketma-ketligini qaytaradi. Ammo ikkinchi index – end xisobga olinmaydi.

**string[start:end:step]** – belgilar ketma-ketligini **start** dan to **end** - indeksigacha bo‘lgan belgilar ketma-ketligini **step** – qadami bilan qaytaradi.

Quyida str satrining tarkibidan belgilar ketma-ketligini olish uchun barcha variantlardan foydalanilgan dastur keltirilgan:

str = "salom dunyo"

# 0-indeksdan 5 - indeksgacha

sub\_str1 = **str[:5]**

**print**(sub\_str1)       # salom

# 2-indeksdan 5-indeksgacha

sub\_str2 = **str[2:5]**

**print**(sub\_str2)       #llo

# 2 - indeksdan 9-indeksgacha oraliqda 2 qadam bilan

sub\_str3 = **str[2:9:2]**

**print**(sub\_str3)       # lmdn

**Satrlarni birlashtirish.** Satrlar ustida eng keng tarqalgan amallardan biri ularni birlashtirish yoki qo‘shish (konkatinatsiya) qilish. Starlar ustida arifmetik qo‘shish amalini bajaradigan bo‘lsak, satrlarni birlashtirish amalini bajargan bo‘lamiz:

name = "Jeyson"

surname = "Stethom"

full\_name = name + " " + surname

**print**(full\_name) # Jeyson Stethom

Ikki qatorni birlashtirish oson, lekin agar satr va raqam qo‘shishi kerak bo‘lsa-chi? Bunday holda str() funksiyasidan foydalanib raqamni satrga o‘tkazish talab qilinadi:

name=" Jeyson "

age = 38

info\_per = "Ism: " + name + " Yoshi: " + **str**(age)

**print**(info\_per)   # Ism: Jeyson Yoshi: 38

**Satrni takrorlash.** Satrlarga arifmetik ko‘paytirish amalini qo‘llash, ularning shuncha marta takrorlaydi:

print("char" \* 3)   # charcharchar

print("u" \* 4)   # uuuu

**Satrlarni taqqoslash.** Satrlarni taqqoslashda asosiy diqqatni belgiga va ularning katta yoki kichik harfligiga qaratiladi. Demak, raqamli belgi shartli ravishda har qanday alifbo belgisidan kichik hisoblanadi. Katta harfdagi alifbo belgilari shartli ravishda kichik alifbo belgilaridan kichik hisoblanadi. Masalan:

str1 = "1a"

str2 = "aa"

str3 = "aa"

print(str1>str2) #False, chunki str1 ni 1- belgisi raqam

print(str2>str3) #Rost, chunki str2 ni 1- belgisi kichik harf

Shuning uchun "1a" qatori shartli ravishda "aa" qatoridan kichikdir. Avval birinchi belgi solishtiriladi. Agar ikkala satrning bosh belgilari raqamlarni ifodalasa, u holda kichikroq raqam kichikroq hisoblanadi, masalan, "1a" satri "2a" dan kichik.

Agar boshlang‘ich belgilar bir xil holatda alifbo belgilarini ifodalasa, unda alifbo bo‘yicha tartibiga qaraladi. Shunday qilib, "aa" "da" dan kichik va "da" "fa" dan kichikdir.

Agar birinchi belgilar bir xil bo‘lsa, ikkinchi belgilar solishtiriladi va hokazo.

Satrlarni katta kichik harflarini bir biri bilan solishtirishdan qochish maqsadida ularning harflarini bir xilga keltirib olish mumkin. Ya’ni katta yoki kichik registrga o‘tkazib olish mumkin.

**lower**() funksiyasi satrdagi barcha harflarni kichik harfga, **upper**() funksiyasi esa satrdagi barcha harflarni bosh harfga o‘zgartiradi. Bu funkmsiyalar satrning o‘ziga o‘zgartirish kiritadi.

str1 = "Tomas"

str2 = "TOMAS"

**print**(str1 == str2)   # False - satrlar teng emas

**print**(str1.**lower**() == str2.**lower**())   # Rost

**ord** va **len** funktsiyalari. Satr Unicode belgilarni o‘z ichiga olganligi sababli, Unicode belgisining raqamli qiymatini olish uchun **ord()** funktsiyasidan foydalanish mumkin:

print(ord("B"))      # 66

String uzunligini olish uchun **len**() funksiyasidan foydalaniladi:

str = "salom dunyo!"

uzunlik = **len**(str)

print(uzunlik)    # 12

**Satrda qidirish.** text in str ifodasidan foydalanib, str qatoridan text satrini qidirib topish mumkin. Agar str satrida text topilsa, u holda ifoda "True" ni qaytaradi, aks holda "False" ni qaytaradi:

str = "salom dunyo"

is\_true = "salom" **in** str

print(is\_true)     # rost

is\_true = "sdunyo" **in** string

print(is\_true)     # False

**Satr bilan ishlovchi operatorlar va metodlardan foydalanish**

Satrlar bilan ishlash uchun ko‘p qo‘llanladigan asosiy metodlarni ko‘rib chiqaylik:

***isalpha* ():** agar satr faqat alfavit belgilaridan iborat bo‘lsa, True qaytaradi;

***islower* ():** Agar satr faqat kichik harflardan iborat bo‘lsa, True qiymatini qaytaradi;

***isupper* ():** Agar satrdagi barcha belgilar katta harflardan iborat bo‘lsa, True qiymatini qaytaradi;

***isdigit* ():** agar satrning barcha belgilari raqam bo‘lsa, True qiymatini qaytaradi;

***isnumeric():*** agar satr raqam bo‘lsa, True qiymatini qaytaradi;

***startswith(str):*** agar satr pastki qator str bilan boshlansa, True qiymatini qaytaradi;

***endswith(str):*** agar satr oxiri **str** bilan tugasa, True qiymatini qaytaradi;

***low():*** satrni kichik harfga o‘zgartiradi;

***upper():*** satrni katta harfga aylantirish;

***title():*** Satrdagi barcha soʻzlarning birinchi harflarini bosh harfga oʻzgartiradi;

***capitalize():*** satrning faqat birinchi so‘zining birinchi harfini bosh harf bilan yozadi;

***lstrip():*** satr boshidagi probellarni o‘chirib, satrni tozalaydi;

***rstrip():*** satr oxiridagi probellarni o‘chirib, satrni tozalaydi;

***strip():*** satrdan oldingi va keyingi boʻshliqlarni olib tashlaydi;

***ljust(width):*** agar satr uzunligi width parametridan kichik bo‘lsa, kenglik qiymatini to‘ldirish uchun satrning o‘ng tomoniga bo‘shliqlar qo‘shiladi va satrning o‘zi oqlanadi;

***rjust(width):*** agar satr uzunligi width parametridan kichik bo‘lsa, kenglik qiymatini to‘ldirish uchun satrning chap tomoniga bo‘shliqlar qo‘shiladi va satrning o‘zi o‘ng tomonga asoslanadi;

***center(width):*** agar satrning uzunligi kenglik parametridan kichik bo‘lsa, kenglik qiymatini to‘ldirish uchun satrning chap va o‘ng tomoniga bo‘shliqlar teng ravishda qo‘shiladi va satrning o‘zi markazlashtiriladi;

***find(str[, start [, end]):*** satrning ko‘rsatilgan intervalidan qism qatorini qidirish. Agar satrdan qism qator topilsa indeksini qaytaradi. Agar qator topilmasa, -1 raqami qaytariladi;

***replace(old, new[, num]):*** satrdagi bir qism qatorni boshqasiga almashtiradi;

***split([delimeter[, num]]):*** ajratuvchiga qarab qatorni qism qatorlarga ajratadi;

***join(strs):*** bir nechta satrni ular orasiga maʼlum ajratuvchi qoʻyib, bir qatorga birlashtiradi.

  Yuqorida keltirilgan metodlarni dasturda misollar bilan ko‘rib chiqaylik. Masalan, agar raqam klaviaturadan kiritilishi kerak bo‘lsa, kiritilgan satrni raqamga aylantirishdan oldin isnumeric() usuli yordamida haqiqatda raqam kiritilganligini tekshirishimiz mumkin va agar shunday bo‘lsa, raqamga o‘girish amalini bajariladi (**isnumeric()** metodi):

str = input(" Raqamni kiriting :")

**if** str.**isnumeric**():

    raqam = **int**(str)

    print(raqam)

Satr ma’lum bir qism qator bilan boshlanishi yoki tugashini tekshirish (**startswith()** metodi):

f = "hello.py"

startsWithHello = f.**startswith**("hello") #True

endsWithExe = f.**endswith**("exe") # False

Satrning boshida va oxiridagi bo‘shliqlarni olib tashlash (**strip()** metodi):

str = "    Salom   dunyo!   "

str = str.**strip**()

print(str)            # salom   dunyo!

Bo‘shliqlar va hizalamalar bilan satrni to‘ldirish (**rjust()** metodi)::

print("Redmi 7:", "500".**rjust**(10))

print("Nokia P10:", "360".**rjust**(10))

Konsol chiqishi:

  Redmi 7:       500

Nokia N10:     360

**Satrda qidirish**

Satrdagi qism qatorni topish uchun Pythonda **find()** metodi mavjud, u satrda qism qatorning birinchi paydo bo‘lish indeksini qaytaradi. **find()** metodining uchta ko‘rinishi mavjud:

***find(str):***  str qism satrini satr boshidan oxirigacha qidiriladi;

***find(str, start):*** start parametri qidiriladigan boshlang‘ich indeksni belgilaydi va start -indeksdan oxirigacha bo‘lgan simvollar ketma-ketligidan qidiradi;

***find(str, start, end):*** bu holda str qism satri satrning start va end indekslari orasidan qidiradi.

Agar qism qator topilmasa, find metodi -1 qiymatni qaytaradi:

Hi\_bye = "Salom dunyo! Xayr dunyo!"

indeks = Hi\_bye.**find**("dun")

**print**(indeks)        #6

# 10 indeksdan qidirish

indeks = Hi\_bye.**find**("dun",10)

**print**(indeks)        #21

# 10 dan 15 gacha indekslarni qidirish

indeks = Hi\_bye.**find**("dun",10,15)

**print**(indeks)        # -1 – topilmaganini anglatadi

**Satrda almashtirish (*replace*).** Satrdagi bitta str satrini boshqasiga almashtirish uchun ***replace***() metodidan foydalaning. Replace metodini bir nechta turda qo‘llash mumkin:

***replace* (**old, new**)*:*** satrdagi **old** satrlarni **new** satr bilan almashtiradi;

***replace*(**old, new, num**):** **num** parametri **old** satrlarning nechtasini **new** satri bilan almashtirishni belgilab beradi. Odatda **num** parametridan foydalanilmasa, u -1 qiymatni qabul qiladi, bu esa barcha **old** satrlarni **new** satri bilan almashtirilishini anglatadi.

tel\_num = "+99-899-667-89-10"

*# defislarni bo‘sh joylar bilan almashtirish*

tahrirlangan\_telefon = tel\_num.**replace**("-", " ")

**print**(tahrirlangan\_telefon)      *# +99 899 667 89 10*

*# defislarni olib tashlash*

tahrirlangan\_telefon = tel\_num.**replace**("-", "")

**print**(tahrirlangan\_telefon)      *# +998996678910*

*# faqat birinchi defisni almashtirish*

tahrirlangan\_telefon = tel\_num.**replace**("-", "", 1)

**print**(tahrirlangan\_telefon)      *# +99899-667-89-10*

**Qism qatorlarga ajratish. *Split()*** metodi chegaralovchiga qarab qatorni qism qatorlar ro‘yxatiga ajratadi. Ajratuvchi har qanday belgi yoki belgilar ketma-ketligi bo‘lishi mumkin. Ushbu metod quyidagi ko‘rinishlarda qo‘llanilishi mumkin:

***split():*** ajratuvchi sifatida bo‘sh joy qilinadi. Va probeldan probelgacha bo‘lgan joylarni element qilib oladi.

***split(delimeter):*** cheklovchi sifatida delimeter dan foydalaniladi.

***split(delimeter, num):*** num parametri bo‘linish uchun delimeter (ajratuvchi) ning necha marta ishlatilishini belgilaydi. Qatorning qolgan qismi qismlarga ajratilmasdan ro‘yxatga qo‘shiladi.

text = "Bu katta, ikki bo‘yli eman, shoxlari singan va po‘stlog‘i singan"

# probellardan ajratiladi

splitted\_text = text.**split**()

print(splitted\_text)

print(splitted\_text[6]) #singan

# vergul ustida tanaffus

splitted\_text = text.**split**(",")

print(splitted\_text)

print(splitted\_text[1]) # ikki bo‘yli eman

# birinchi besh boʻshliqqa boʻlingan

splitted\_text = text.**split**(" ", 5)

**print**(bo‘lingan\_matn)

**print**(bo‘lingan\_matn[5])

# shoxlari singan va po‘stlog‘i singan

**Satrlarni birlashtirish.** Satrlar ustidagi eng oddiy amallarni ko‘rib chiqa turib, qo‘shish amali yordamida qatorlarni birlashtirishni ko‘rsatdik. Satrlarni birlashtirishning yana bir imkoniyati - ***join() usuli:*** u bir nechta satrlarni birlashtirish uchun ishlatiladi.

Bundan tashqari, ushbu metod chaqirilgan joriy satr ajratuvchi sifatida ishlatiladi:

strings = ["Let’s", "go", "speaking", "from", "my", "heart", "into", " My box"]

*# ajratuvchi - bo‘sh joy*

sentence = " ".**join**(strings)

**print**(sentence)

*# Let’s go speaking from my heart into My box*

*# ajratuvchi  - vertikal chiziq*

sentence = " | ".**join**(strings)

**print**(sentence)

*# Let’s | go | speaking | from | my | heart | into | My box*

Ro‘yxat o‘rniga oddiy satr qo‘shilish usuliga o‘tkazilishi mumkin, keyin ajratuvchi ushbu satr belgilari orasiga kiritiladi:

str = "hello"

joined\_str = "\".**join**(str)

**print**(joined\_str) # h\e\l\l\o

**2. str.format() metodi yordamida satrlarni formatlash**

Oldingi mavzularda satr oldiga f belgisini qo‘yish orqali ba’zi qiymatlarni qanday kiritish mumkinligi ko‘rib chiqilgan edi. Masalan:

first\_name="Jeyson"

text = f"Hello, {first\_name}."

print(text) # Hello, Jeyson.

name="Bobo"

age=73

info = f"Ismi: {name}\t Yoshi: {age}"

**print**(info) # Ismi: Bobo Yoshi: 73

Satrga o‘zgaruvchining qiymatini kiritish uchun maxsus parametrlardan foydalaniladi, ular jingalak qavslar ({}) bilan o‘rab olinadi.

**Nomlangan parametrlar.** Formatlanayotgan satrga chaqirilayotgam o‘zgaruvchilarni **format()** metodiga parametr sifatida beriladi. Format metodi esa ushbu parametrlarning qiymatlarini satr tarkibida jingalak qavslarda ko‘rsatilgan mos nomlarga uzatadi:

txt = "Salom, {first\_name}.".format(first\_name="Jeyson")

print(txt) # Salom, Jeyson.

info = "Ismi: {name}\t Yoshi: {age}".format(name="Bobo", age=73)

print(info) # Ismi: Bobo Yoshi: 73

Demak, formatlash metodida parametrlar qatordagi parametrlar bilan satr tarkibidagi jingalak qavslar ichidagi nomlar ham bir xil bo‘llishi talab qilinadi. Shunday qilib, agar parametr yuqorida ko‘rsatilga dasturda bo‘lgani kabi first\_name deb ataladigan bo‘lsa, u holda qiymat tayinlangan argument ham first\_name deb ataladi.

**Joylashgan o‘rni bo‘yicha parametrlar.** Shuningdek, ketma-ket argumentlar to‘plamini formatlash usuliga o‘tkazish va bu argumentlarni format satrining o‘ziga, ularning indeksini jingalak qavslar ichida ko‘rsatish mumkin (indekslash noldan boshlanadi):

info = "Ismi: {0}\t Age: {1}".format("Bobo", 73)

print(info) # Ismi: Bobo Age: 73

Bunday holda, argumentlar satrga bir necha marta chaqirilishi ham mumkin:

txt = "Salom, {0} {0} {0}.".format("Jeyson")

Salom, Jeyson Jeyson Jeyson

**Almashtirishlar.** Formatlangan qiymatlarni satrga o‘tkazishning yana bir usuli - o‘rniga ma’lum qiymatlar qo‘shiladigan almashtirishlar yoki maxsus to‘ldiruvchilardan foydalanish. Formatlash uchun quyidagi to‘ldiruvchilardan foydalanish mumkin:

**s:** String tipli ma’lumot kiritish uchun;

**d:** Double tipli sonlarni kiritish uchun;

**f:** Float tipli sonlarni kiritish uchun. Bu tur uchun nuqta orqali kasr sonini aniqlash ham mumkin;

**%:** qiymat 100 ga ko‘paytiriladi va foiz belgisini qo‘shib qo‘yiladi.

 To‘ldiruvchining umumiy sintaksisi quyidagicha:

{:[maxsus to‘ldiruvchi] }

To‘ldiruvchiga qarab qo‘shimcha parametrlar qo‘shilishi mumkin. Masalan, float raqamlarini formatlash uchun siz quyidagi variantlardan foydalanishingiz mumkin

{:[belgilar\_soni][vergul][.sonning\_verguldan\_keyingi\_qismi] to‘ldiruvchi}

Format metodini chaqirganda, qiymatlar unga argumentlar sifatida uzatiladi, ular to‘ldiruvchilar o‘rniga kiritiladi. String almashtirish ({:s}) ga misol:

hi = "Salom sizga {:s}"

name = "Jeyson"

formatted\_txt = hi.**format(name)**

print(formatted\_txt) # Salom sizga Jeyson

Format() usuli natijada yangi formatlangan qatorni qaytaradi.

 {:d} - butun son formatlashga misol:

txt = "{:d} ta simvol"

number = 12

formatted\_txt = txt**.format(number)**

print(formatted\_txt) # 12 ta simvol

Agar formatlash kerak bo‘lgan raqam 999 dan katta bo‘lsa, vergulni minglik ajratuvchi sifatida ishlatmoqchi ekanligimizni to‘ldiruvchi ta’rifida belgilash ({:,d}) mumkin:

txt = "{:,d} belgi"

print(txt.**format(300)**) # 300 belgi

Bundan tashqari, to‘ldiruvchilar f-satrlarida ham ishlatilishi mumkin:

n = 300

txt = f"{n:,d} belgi"

print(txt) # 200 belgi

Kasr sonlar uchun, sonning kasr qismida qancha belgi ko‘rinishini aniqlash mumkin. Buning uchun verguldan keyin ko‘rinishi kerak bo‘lgan sonlar soni, nuqtadan keyin to‘ldiruvchidan oldin kiritiladi. Masalan:

num = 115.2548695

print("{:.2f}".**format**(num)) *# 115.25*

print("{:.3f}".**format**(num)) *# 115.254*

print("{:.4f}".**format**(num)) *# 115.2548*

print("{:,.2f}".**format**(12582.23664)) *# 12,582.23*

Yana bir parametr formatlangan qiymatning minimal kengligini belgilar bilan belgilash imkonini beradi:

print("{:10.2f}".**format**(115.2548695)) *# 115.25*

print("{:8d}".**format**(25)) *# 25*

**format** metodini o‘rnini bosuvchi **f**  usuli:

n1 = 115.2548695

print(f"{n1:10.2f}") *# 115.25*

n2 = 26

print(f"{n2:8d}") *# 26*

**Foizlarni** ekranga chiqarish uchun **"%"** kodini ishlatish yaxshiroqdir :

num = .123456

print("{:%}".**format**(num)) *# 12.3456000%*

print("{:.0%}".**format**(num)) *# 12%*

print("{:.1%}".**format**(num)) *# 12.3%*

print(f"{num:%}") *# 12.3456000%*

print(f"{num:.0%}") *# 12%*

print(f"{num:.1%}") *# 12.3%*

**Format metodini ishlatmasdan formatlash.** Quyidagi sintaksis bilan formatlashning yana bir usuli ham mavjud:

string**%**(param1, param2,..paramN)

Ya’ni, boshida yuqorida ko‘rib chiqilgan bir xil to‘ldiruvchilarni o‘z ichiga olgan qator mavjud (% to‘ldiruvchidan tashqari) satrdan keyin foiz - % belgisi qo‘yiladi va keyin qatorga kiritiladigan qiymatlar ro‘yxati keltiriladi. Aslida, foiz belgisi yangi qatorga olib keladigan operatsiyani anglatadi:

info = " Ism : **%s** \t Yoshi : **%d**" **%** ("Jeck", 32)

print(info)    *# Ism : Jeck      Yoshi : 32*

Toʻldiruvchidan keyin foiz belgisi qoʻyiladi va format funksiyasidan farqli oʻlaroq, bu yerda jingalak qavslar kerak emas.

Bundan tashqari, bu yerda raqamlarni formatlash usullari ham qo‘llaniladi:

raqam = 23.8689578

print("**%0.2f**   - **%e**" **%** (raqam, raqam))

*# 23.87   - 2.386896 e +01*

**Nazorat savollari:**

1. Format metodining qo‘llanilishi qanday?
2. Format metodining “Nomlangan parametrlar” usulini tushuntirib bering.
3. Format metodining “Pozitsiya bo‘yicha parametrlar” usulini tushuntirib bering
4. Almashtirishlar – bu qanday vazifani bajaradi?
5. Format metodisiz formatlash qanday amalga oshiriladi?
6. isalpha () - metodining vazifasi qanday?
7. islower () - metodining vazifasi qanday?
8. isupper () - metodining vazifasi qanday?
9. isdigit  () - metodining vazifasi qanday?
10. startswith () - metodining vazifasi qanday?
11. endswith () - metodining vazifasi qanday?
12. replace () - metodining vazifasi qanday?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**4-mashg‘ulot.** Shart operatori. Shart operatoriga doir dasturlari tuzish.

**O‘quv savollari:**

1. IF, IF-ELSE va IF-ELIF-ELSE operatorlari.

**1. IF, IF-ELSE va IF-ELIF-ELSE operatorlari.**

Shartli operatorlari shartli ifodalardan foydalanadi va ularning qiymatiga qarab, dasturning bajarilishini yo‘llardan biri bo‘ylab yo‘naltiradi. Shunday operatorlardan biri **if** operatoridir. U quyidagi quyidagicha sintaksisga ega:

**if** mantiqiy\_amal:

[if bloki]

**elif** mantiqiy\_amal:

[elif bloki]

**else**:

[else bloki]

Eng oddiy shaklda if kalit so‘zidan keyin mantiqiy ifoda keladi. Va agar bu mantiqiy ifoda "True" ni qaytarsa, u holda ko‘rsatmalarning keyingi bloki bajariladi, ularning har biri blokga tegishliligini oldiga 4 ta probel yoki 1 ta tab tashlanishi bilan aniqlanadi (4 bo‘sh joy yoki bo‘shliqlar sonini cheklash maqsadga muvofiqdir) bu 4 ga karrali):

til = "uzbekcha"

**if** til == "uzbekcha":

**print**("Assalomu alaykum")

**print**("Xayr")

Bu holda til o‘zgaruvchisining qiymati "uzbekcha" bo‘lganligi sababli, if bloki bajariladi, unda faqat bitta operator mavjud - print("Assalomu alaykum"). Natijada, konsol quyidagi qatorlarni ko‘rsatadi:

Assalomu alaykum

Xayr

Kodning oxirgi qatoriga e’tibor bering, unda " Xayr" xabari ko‘rsatiladi. Chunki print(“Xayr”) oldidan tab tashlanmagan, shuning uchun u if blokiga tegishli emas va if konstruktsiyasidagi ifoda False qiymatini qaytarsa ​​ham bari bir print(“Xayr”) amali bajariladi.

Ammo agar biz uni cheklasak, u if konstruktsiyasiga ham tegishli bo‘ladi:

til = "uzbekcha"

**if** til == "uzbekcha":

print("Assalomu alaykum")

print("Xayr")

**block else**:

Mabodo if ifodasi False qiymatini qaytarsa, unda else bloki ishga tushadi:

til = "uzbekcha"

**if** til == "uzbekcha":

print("Assalomu alaykum")

**else**:

print("Hi")

print("Xayr")

Agar til == "uzbekcha" iborasi True qiymatini qaytarsa, u holda if bloki bajariladi, aks holda else bloki bajariladi. Va bu holda shart til == "uzbekcha" False qiymatini qaytarganligi sababli, else blokidagi operator bajariladi.

Bundan tashqari, else bloki tarkibiga kiruvchi operatorlarning oldidan ham satr boshidan tab (4ta probelga teng bo‘sh joy) tashlanishi kerak. Misol uchun, yuqoridagi misolda print("Xayr") ning oldidan tab tashlanmagan, shuning uchun u else bloki tarkibiga kiritilmagan va til o‘zgaruvchisining qiymati qaysi tilda bo‘lishidan qat’iy nazar bajariladi. Ya’ni, konsol quyidagi qatorlarni ko‘rsatadi:

Assalomu alaykum

Xayr

**else** bloki tarkibi bir nechta qatorlardan tashkil topgan bo‘lishi ham mumkin:

til = "uzbekcha"

**if** til == "uzbekcha":

print("Salom")

print("Dunyo!")

**else**:

print("Hello")

print("World!")

Agar siz bir nechta aniq shartlarni kiritishingiz kerak bo‘lsa, unda siz qo‘shimcha elif bloklaridan foydalanishingiz kerak bo‘ladi:

til = "uzbekcha"

**if** til == "uzbekcha":

print("Assalomu")

print("Alaykum")

**elif** til == "german":

print("Hallo")

print("Welton")

**else**:

print("What’s")

print("up")

Python bu yerda birinchi, if ning shartini tekshiradi. Agar rost bo‘lsa, if blokidagi operatorlar bajariladi. Agar bu shart False qiymatini qaytarsa, Python elif shartini tekshiradi.

Agar elifning sharli ifodasi True bo‘lsa, u holda elif blokidagi opertorlar bajariladi. Ammo agar u ham False qaytarsa, else blokidaki bajariladi.

Shartlarni elif operatoridan foydalangan holda ixtiyoriy davom ettirishingiz mumkin. Masalan :

til = "Uz"

**if** til == "english":

print("Hello")

**elif** til == "german":

print("Hallo")

**elif** til == "french":

print("Salut")

**else**:

print("Salom")

**Ichma-ich joylashgan if operatori.** if operatorining sharti bajarilganidan keyin yanayam aniqlashtirish uchun ichida yana bir necha shartga tekshiriladi:

til = "uzbekcha"

payt == "ertalabki vaqt"

**if** til == "uzbekcha":

print("Uzbek")

**if** payt == "ertalabki vaqt":

print("Unda hayrli tong!")

**elif** payt == "kechgi vaqt":

print("Unda sizga hayrli kech!")

Bu yerda if konstruksiyasi ichki o‘rnatilgan if/elif konstruktsiyalarini o‘z ichiga oladi. Ya’ni, agar til o‘zgaruvchisi "uzbekcha" ga teng bo‘lsa, u holda ichki o‘rnatilgan if/elif konstruksiyalarisi kunduzgi o‘zgaruvchining qiymatini qo‘shimcha ravishda tekshirishadi - payt o‘zgaruvchisining qiymati "ertalabki vaqt" ga tengmi yoki "kechgi vaqt" ga tengligini tekshiradi va shunga qarab javob qaytariladi. Va bu holda, biz quyidagi konsol chiqishini olamiz:

Uzbek

Unda hayrli tong

Esda tutingki, ichki o‘rnatilgan if ifodalarining oldidan tab qoldirish kerak, ichki o‘rnatilgan konstruktsiyalardagi operatorlarning oldidan ham tab (bo‘sh joy) qoldirilishi kerak. To‘g‘ri joylashtirilmagan tabulyatsiya dastur mantig‘ini o‘zgartirishi mumkin.

Xuddi shunday, ichki o‘rnatilgan if/elif/else konstruksiyalari elif va else bloklariga joylashtirilishi mumkin:

til = "uzbek"

payt = "ertalab"

**if** til == "uzbek":

**if** payt == "ertalab":

print("Hayrli tong")

**else**:

print("Hayrli kech")

**else**:

**if** payt == "ertalab":

print("Good morning")

**else**:

print("Good evening")

**Nazorat savollari :**

1. If operatori qanday vazifasi bajaradi?
2. If operatorining strukturasi qanday?
3. Elif operatorining vazifasi qanday?
4. Ichma – ich foydalaniladigan shart operatoriga misollar keltiring.
5. Bloklarning if ifodaga tegishliligini qanday bilish mumkin?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**5-mashg‘ulot.** Pythonda takrorlanuvchi jarayonlarni dasturlash.

**O‘quv savollari:**

1. Sikl operatorlari – For va while bilan ishlash.
2. Break, continue va else operatorlarining qo‘llanilishi.

**1. Sikl operatorlari – For va while bilan ishlash.**

Sikllar biror bir shartning bajarilishiga qarab ba’zi harakatlarni bajarishga imkon beradi. Pythonda quyidagi turdagi sikllar mavjud: **while**va **for.**

**WHILE sikli.** while sikli qandaydir bir shartning to‘g‘riligini tekshiradi va agar shart True bo‘lsa, u holda sikl tanasidagi blokni bajaradi. U quyidagicha sintaksisga ega:

**while** <shartli\_ifoda>:

[sikl tanasi]

**while** kalit so‘zidan keyin shartli ifoda keladi va bu ifoda True qiymat qaytarsa, sikl tanasiga kirib u yerdagi operatorlar ketma-ket bajariladi.

**while** sikliga tegishli barcha bloklar oldidan bo‘sh joy qoldirilib ketishi va boshlanish qismlari bir ustunga joylashgan bo‘lishi kerak va while kalit so‘zidan bir xilda chekingan bo‘lishi kerak.

son = 1

**while** son < 5:

print(f"son = {son}")

son += 1

print("Dastur tugatildi")

Bunday holda, son o‘zgaruvchisi 5 dan kichik bo‘lsa, while sikli ishlaydi. Aks holda sikldan keyingi amal bajariladi.

Bu sikl tanasi ikkita ifodadan iborat:

print(f"son = {son}")

son += 1

Shuni ham yodda tutingki, oxirgi print(“Dastur tugallandi”) satr boshidan chekinmaydi, shuning uchun u while siklining bir qismi emas.

*Siklning butun jarayonini quyidagicha ifodalash mumkin:* Birinchi, **son** o‘zgaruvchisining qiymati 5 dan kichik yoki kichik emasligini tekshiriladi. Va o‘zgaruvchi dastlab 1 ga teng bo‘lganligi sababli, bu shart True ni qaytaradi va shuning uchun sikl operatorlari bajariladi.

Sikl ko‘rsatmalari satr **son**=1 ni konsolga chiqaradi. Va keyin raqam o‘zgaruvchisining qiymati bittaga oshiriladi - endi u 2 ga teng bo‘ladi. sikl ko‘rsatmalari blokini bir marta bajarish iteratsiya deb ataladi. Ya’ni, shu tarzda, birinchi takrorlash siklda amalga oshiriladi.

**son** **< 5** sharti yana tekshiriladi. Yana True qaytariladi, chunki son 2 ga teng, shuning uchun sikl operatorlari yana bir marta bajariladi.

Sikl ko‘rsatmalari konsolga satr **son** = 2 ni chiqaradi. Va keyin **son** o‘zgaruvchisining qiymati yana bittaga oshiriladi - endi u 3 ga teng bo‘ldi. Shunday qilib, ikkinchi takrorlash amalga oshiriladi.

**son < 5** shart yana tekshiriladi. Yana True qiymat qaytariladi, chunki son = 3, shuning uchun sikl operatorlari bajariladi.

Sikl ko‘rsatmalari konsolga satr **son** = 3 ni chiqaradi. Va keyin **son** o‘zgaruvchisining qiymati yana bittaga oshiriladi - endi u 4 ga teng. Ya’ni uchinchi iteratsiya amalga oshiriladi.

**son < 5** sharti yana tekshiriladi. Yana True qiymat qaytariladi, chunki son = 4, shuning uchun sikl ko‘rsatmalari bajariladi.

Sikl ko‘rsatmalari konsolga son= 4 qator raqamini chiqaradi.Va keyin son o‘zgaruvchisining qiymati yana bittaga oshiriladi - endi u 5 ga teng. Ya’ni to‘rtinchi takrorlash amalga oshiriladi.

  Va shart son < 5 yana tekshiriladi. Lekin endi u False qiymat qaytaradi, chunki son o‘zgaruvchisining qiymati endi 5 ga teng, shuning uchun sikl tugatiladi. Shundan so‘ng sikldan keyin operatorlar bajariladi. Shunday qilib, bu sikl to‘rtta o‘tish yoki to‘rtta takrorlashni amalga oshiradi

 Natijada, kodni bajarishda biz quyidagi konsol chiqishini olamiz:

son = 1

son = 2

son = 3

son = 4

Dastur tugatildi

While sikli bilan yana bir oprator – else ham ishlatiladi. U while siklidagi shart False qiymat qaytarganida ishga tushadi va tarkibidagi blok bajariladi:

raqam = 1

**while** raqam <5:

print(f"raqam = {raqam}")

raqam += 1

**else**:

print(f"raqam = {raqam}. sikl tugallandi")

print("Dastur tugatildi")

Ya’ni, bu holda birinchi navbatda shart tekshiriladi va while operatorlari bajariladi. Keyin shart False ga aylanganda else blokidagi gaplar bajariladi. E’tibor bering, else blokidagi iboralar ham sikl konstruktsiyasining boshidan chekinadi. Natijada, bu holda biz quyidagi konsol chiqishini olamiz:

raqam = 1

raqam = 2

raqam = 3

raqam = 4

raqam = 5. sikl tugallandi

Dastur tugadi

**For sikli.** Sikl ning yana bir turi for konstruktsiyasidir. Ushbu sikl qiymatlar to‘plamini takrorlaydi, har bir qiymatni o‘zgaruvchiga qo‘yadi va keyin siklda biz ushbu o‘zgaruvchi bilan turli harakatlarni bajarishimiz mumkin. For siklining strukturasi quyidagicha:

**for** [o‘zgaruvchi] in [qiymatlar\_to‘plami]:

[sikl tanasi]

For kalit so‘zidan keyin qiymatlar joylashtiriladigan o‘zgaruvchining nomi keladi. Keyin in operatoridan keyin qiymatlar to‘plami va ikki nuqta qo‘yiladi.

Va keyingi qatordan sikl tanasidagi operatorlar boshlanadi. Bu operatorlar siklning bir qismi ekanligi uchun ular for kalit so‘zi boshidan ichkariga kiritilishi kerak.

Siklni bajarishda Python ketma-ket barcha qiymatlarni to‘plamdan oladi va ularni o‘zgaruvchiga uzatadi. To‘plamdagi barcha qiymatlar takrorlangandan keyin sikl tugaydi.

Masalan, qiymatlar to‘plami sifatida siz aslida belgilar to‘plamini ifodalovchi satrni ko‘rib chiqishingiz mumkin. Keling, bir misolni ko‘rib chiqaylik :

string = "Hello"

**for** char **in** string:

print(char)

**char** o‘zgaruvchisi siklda aniqlanadi, **in** operatoridan keyin **string** o‘zgaruvchisi "Hello" satrini saqlaydigan takrorlangan to‘plam sifatida ko‘rsatiladi. Natijada, for sikli string tarkibidagi barcha belgilarni ketma-ketlikda takrorlaydi va ularni **char** o‘zgaruvchisiga o‘zlashtiradi. Sikl blokining o‘zi **char** o‘zgaruvchining qiymatini konsolga chop etadigan bitta funksiyadan iborat. Dasturning konsol chiqishi:

H

e

l

l

o

For siklida sikl tugagandan keyin bajariladigan qo‘shimcha else bloki ham bo‘lishi mumkin:

string = "Hello"

**for** char **in** string:

print(char)

**else**:

print(f"oxirgi simvol: {char}

print("Dastur yakunlandi")

Bunday holda, biz quyidagi konsol chiqishini olamiz:

H

e

l

l

o

oxirgi simvol: o

Dastur yakunlandi

Shuni ta’kidlash kerakki, else bloki for siklida aniqlangan barcha o‘zgaruvchilarga kirish huquqiga ega.

**Ichma-ich sikllar.** Ba’zi sikllar o‘zlarida boshqa sikllarni o‘z ichiga olishi mumkin. Ko‘paytirish jadvalining chiqishi misolini ko‘rib chiqing:

a = 1

b = 1

while a < 10:

while b < 10:

print(a \* b, end="\t")

b += 1

print("\n")

b = 1

a += 1

Tashqi sikl while a < 10: a o‘zgaruvchisi 10 ga teng bo‘lgunga qadar 9 marta ishlaydi. Bu sikl ichida ichki sikl while b < 10: ishlaydi. b o‘zgaruvchisi 10 ga teng bo‘lgunga qadar ichki sikl ham 9 marta ishlaydi. Bundan tashqari, ichki siklning barcha 9 ta takrorlanishi tashqi siklning bir iteratsiyasi ichida ishga tushiriladi.

Ichki halqaning har bir iteratsiyasida konsolda a va b raqamlarining mahsuloti ko‘rsatiladi. Keyin b o‘zgaruvchining qiymati bittaga oshiriladi. Ichki sikl o‘z ishini tugatgandan so‘ng, b o‘zgaruvchining qiymati 1 ga o‘rnatiladi va a o‘zgaruvchining qiymati bittaga oshiriladi va tashqi siklning keyingi takrorlanishiga o‘tish sodir bo‘ladi. Va a o‘zgaruvchisi 10 ga teng bo‘lgunga qadar hamma narsa takrorlanadi. Shunga ko‘ra, ichki sikl tashqi siklning barcha iteratsiyasi uchun faqat 81 marta ishlaydi. Natijada, biz quyidagi konsol chiqishini olamiz:

1          2        3        4        5        6        7        8        9

2          4        6        8        10      12      14      16      18

3          6        9        12      15      18      21      24      27

4          8        12      16      20      24      28      32      36

5          10      15      20      25      30      35      40      45

6          12      18      24      30      36      42      48      54

7          14      21      28      35      42      49      56      63

8          16      24      32      40      48      56      64      72

9          18      27      36      45      54      63      72      81

for siklini xuddi shunday aniqlanishi mumkin:

**for** c1 **in**  "ab":

**for** c2 **in** "ba":

        print(f"{c1}{c2}")

Bunday holda, tashqi sikl "ab" qatoridan o‘tadi va har bir belgini c1 o‘zgaruvchisiga qo‘yadi. Ichki sikl "ba" satrini takrorlaydi, satrdagi har bir belgini c2 o‘zgaruvchisiga qo‘yadi va ikkala belgi kombinatsiyasini konsolga chiqaradi. Ya’ni, oxirida biz a va b belgilarning barcha mumkin bo‘lgan kombinatsiyalarini olamiz:

ab

aa

bb

ba

**2. Break, continue va else operatorlarining qo‘llanilishi**

Biz siklni boshqarish uchun maxsus break va continue operatorlaridan foydalanishimiz mumkin. Break operatori siklni to‘xtatadi. continue operatori esa siklning keyingi iteratsiyasiga o‘tkazadi.

Agar siklda uning keyingi bajarilishiga mos kelmaydigan shartlar hosil bo‘lsa, break operatoridan foydalanish mumkin. Quyidagi misolni ko‘rib chiqing :

raqam = 0

**while** raqam <5:

raqam += 1

**if** raqam == 3: # agar raqam = 3 bo‘lsa,sikldan chiqamiz

break

print(f"raqam = {raqam}")

Bu yerda while sikli raqam<5 shartini rostlikka tekshiradi. Va agar raqam o‘zgaruvchisining qiymati hozircha 5 ga teng bo‘lmasa, raqamning qiymati konsolda chop etiladi. Ammo ungacha sikl ichida yana bir shart ham tekshiriladi: if raqam == 3. Ya’ni **raqam**ning qiymati 3 ga teng bo‘lsa, u holda break operatori yordamida sikldan chiqamiz. Va oxirida biz quyidagi konsol chiqishini olamiz:

raqam = 1

raqam = 2

**Break** operatoridan farqli o‘laroq, **continue** operatori siklning keyingi iteratsiyasiga uni tugatmasdan o‘tadi. Misol uchun, oldingi misolda break ni continue bilan almashtiramiz:

raqam = 0

**while** raqam <5:

raqam += 1

**if** raqam == 3: # agar raqam = 3 bo‘lsa, siklning yangi iteratsiyasiga o‘ting

continue

print(f" raqam = {raqam}")

Va bu holda, agar raqam o‘zgaruvchisining qiymati 3 ga teng bo‘lsa, continue operatori siklni keyingi aylanishga o‘tkazib yuboradi va navbatdagi qadam amalga oshiriladi:

raqam = 1

raqam = 2

raqam = 4

raqam = 5

**Nazorat savollari :**

1. Dasturlashda takrorlanuvchi jarayonlarni dasturlash qanday ro‘l egallaydi?
2. Takrorlanuvchi jarayonlarni qanday operatorlar yordamida dasturini tuziladi?
3. **for** operatorining ishlash prinsipi qanday?
4. **while** operatorining sintaksisi va ishlash prinsipi qanday?
5. **for** va **while** operatorlarining bir biridan asosiy farqlari nimada?
6. **else** operatorini siklik jarayonlarda nima maqsadda foydalaniladi?
7. **range** operatorining vazifasi va ishlash prinsipi qanday?
8. **break** operatorining vazifasi nima?
9. **continue** operatori qanday holatlarda ishlatiladi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**6-mashg‘ulot.** Pythonda roʻyxatlar bilan ishlash.

**O‘quv savollari:**

1. Ro‘yxatlar va ularning qo‘llanilishi.
2. Ro‘yxatlarni yaratish usullari.
3. Ro‘yxatlar bilan ishlovchi metodlar.

**1. Ro‘yxatlar va ularning qo‘llanilishi.**

Ma’lumotlar to‘plamlari bilan ishlash uchun Python ro‘yxat, kortej va lug‘atlar kabi o‘rnatilgan turlarni taqdim etadi.

**Ro‘yxat (list)** - elementlar to‘plami yoki ketma-ketligini saqlaydigan ma’lumotlar turi. Ko‘pgina dasturlash tillarida massiv deb ataladigan o‘xshash ma’lumotlar tuzilmasi mavjud.

**2. Ro‘yxatlarni yaratish usullari**

Ro‘yxatni yaratish uchun kvadrat qavslar [] ishlatiladi. Ularning ichida ro‘yxat elementlari vergul bilan ajratilgan holda saqlanadi. Masalan, raqamlar ro‘yxatini yarataylik:

raqamlar = [0, 2, 12, 44, 51]

Xuddi shunday, boshqa turdagi ma’lumotlar bilan ro‘yxatlarni belgilash mumkin, masalan, satrlar (ya’ni string turdagi ma’lumotlar) ro‘yxatini yaratish quyidagicha amalga oshiriladi:

odamlar = ["Jasur", "Akmal", "Bobur"]

Roʻyxat yaratish uchun **list()** konstruktor funksiyasidan ham foydalanish mumkin **:**

raqamlar1 = **[]**

raqamlar2 = **list()**

Ushbu ikkala ro‘yxat ta’riflari o‘xshash - ular bo‘sh ro‘yxat yaratadi.

Ro‘yxatda faqat bir xil turdagi ob’ektlar bo‘lishi shart emas. Bir vaqtning o‘zida satrlarni, raqamlarni va boshqa ma’lumotlar turlarining ob’ektlarini bir xil ro‘yxatga joylashtirish mumkin:

obyektlar = [1, 2.26, "Salom dunyo", True, False]

Roʻyxat elementlarini ekranga chiqarish uchun standart print funksiyasidan foydalanish mumkin. Bu roʻyxat tarkibini to‘liq shaklda konsolga chop etadi:

raqamlar = [1, 2, 3, 4, 5]

odamlar = ["Jasur", "Akmal", "Bobur"]

print(raqamlar)*# [1, 2, 3, 4, 5]*

print(odamlar)    *# ["Jasur", "Akmal", "Bobur"]*

list() konstruktori qiymatlar to‘plamini qabul qilishi mumkin, ular asosida ro‘yxat tuziladi:

raqamlar1 = [1, 2, 3, 4, 5]

raqamlar2 = **list**(raqamlar1)

print(raqamlar2)   *# [1, 2, 3, 4, 5]*

harflar = **list**("Salom")

print(harflar)       *# [‘S’, ‘a’, ​​’l’, ‘o‘, ‘m’]*

Agar bir xil qiymat bir necha marta takrorlanadigan ro‘yxatni yaratish kerak bo‘lsa, unda yulduzcha - \* belgisidan foydalanish mumkin, ya’ni aslida mavjud ro‘yxatga ko‘paytirish amalini qo‘llaniladi:

raqamlar = [15]\*6      *# 15 marta 6 takrorlash*

print(raqamlar)       *# [15, 15, 15, 15, 15, 15]*

odamlar = ["Jasur"]**\*3**  *# "Tom" ni 3 marta takrorlash*

print(odamlar)        *# ["Jasur", "Jasur", "Jasur"]*

talabalar = ["Bobur", "Saman"] **\* 2**

*# "Bobur", "Saman" ni 2 marta takrorlash*

print (talabalar)

*# ["Bobur", "Saman", "Bobur", "Saman"]*

**Ro‘yxat elementlariga murojaat qilish.** Ro‘yxat elementlariga murojaat qilish uchun ro‘yxatdagi element sonini ifodalovchi indekslardan foydalanish kerak. Indekslar noldan boshlanadi. Ya’ni, birinchi element indeksi 0, ikkinchi element indeksi 1 ga teng bo‘ladi va hokazo. Elementlarga teskari tartibda ya’ni oxiridan boshiga qarab murojaat qilish uchun manfiy indeks tushunchasi kiritilgan. Bu manfiy indekslar -1 dan boshlanadi. Ya’ni, oxirgi element -1 indeksiga ega bo‘ladi, oxirgidan bitta oldingisi esa -2 indeksiga ega bo‘ladi va hokazo.

odamlar = ["Jasur", "Samat", "Bobur"]

*# ro‘yxat boshidan elementlarni olish*

print(odamlar[0])    # Jasur

print(odamlar[1])    # Samat

print(odamlar[2])    # Bobur

*# roʻyxat oxiridagi elementlarni olish*

print(odamlar[-2])    *# Samat*

print(odamlar[-1])    *# Bobur*

print(odamlar[-3])    *# Jasur*

Ro‘yxat elementini o‘zgartirish uchun unga yangi qiymat yuklash kifoya:

odamlar = ["Jasur", "Samat", "Bobur"]

odamlar[1] = "Akmal"   *# ikkinchi elementni o‘zgartirish*

**print**(odamlar[1])     *# Akmal*

**print**(odamlar)        *# ["Jasur", " Akmal", "Bobur"]*

**Ro‘yxat elementlarini ajratish.** Python sizga ro‘yxatni alohida elementlarga ajratish imkonini beradi:

odamlar = ["Jasur", "Bobur", "Samat"]

jasur, bobur, samat = odamlar

**print** (jasur)       *# Jasur*

**print** (bobur)       *# Bobur*

**print** (samat)       *# Samat*

Bunday holda, jasur, bobur va samat o‘zgaruvchilari odamlar ro‘yxatidan ketma-ket tayinlangan elementlardir. Biroq, o‘zgaruvchilar soni tayinlangan ro‘yxat elementlari soniga teng bo‘lishi talab qilinadi. Aks xolda xatolik kelib chiqadi.

Ro‘yxat elementlarini olish uchun for yoki while tsiklidan foydalanish mumkin.

For tsikli bilan takrorlash:

odamlar = ["Tomson", "Sems", "Bob"]

**for** odam **in** odamlar:

**print** (odam)

Bu yerda odamlar ro‘yxati takrorlanadi va har bir element shaxs o‘zgaruvchisiga joylashtiriladi.

Iteratsiyani while tsikli bilan ham bajarish mumkin:

odamlar = ["Tomson", "Sems", "Bob"]

i = 0

**while** i < **len**(odamlar):

**print**(odamlar[i])

# elementni olish uchun indeksdan foydalanish

    i += 1

**len()** funktsiyasidan foydalanib takrorlash uchun ro‘yxat uzunligini olinadi. Bu dasturda i - hisoblagich qiymati ro‘yxat uzunligiga teng bo‘lguncha chop etadi. i ning qiymati esa har safar birga oshirilib boraveradi.

**Ro‘yxatni taqqoslash.** Ikki ro‘yxat, agar ular bir xil elementlar to‘plamini o‘z ichiga olsa, teng hisoblanadi:

raqamlar1 = [1, 2, 3, 4, 5]

raqamlar2 = list ([1, 2, 3, 4, 5])

**if** raqamlar1 == raqamlar2:

**print**("1 - raqamlar 2 - raqamlarga teng")

**else**:

**print**("1-raqamlar 2-raqamlarga teng emas")

Bunday holda, ikkala ro‘yxat ham teng bo‘ladi.

**3. Ro‘yxatlar bilan ishlovchi metodlar**

Ro‘yxat elementlarini boshqarish uchun bir qancha metodlar mavjud. Ulardan asosan ko‘p marta murojaat qilinadiganlarini quyida keltirilgan:

***append(element):*** **element** - ni roʻyxat oxiriga qoʻshish;

***insert (indeks, element):*** ro‘yxatga ***element***ni ***indeks*** – element qilib qo‘shish;

***extend (items):*** roʻyxat oxiriga ***items*** toʻplamini qoʻshish;

***remove(item):*** ro‘yxatdan ***item*** elementini olib tashlaydi. Ro‘yxatdan birinchi uchragan ***item*** elementi o‘chiriladi. Agar element topilmasa, ValueError xatoligi qaytariladi;

***clear():*** roʻyxatdagi barcha elementlarni oʻchirish;

***index(item):*** ***item*** elementining indeksini qaytaradi. Agar element topilmasa, ValueError xatoligi qaytariladi;

***pop([indeks]):*** ***indeks*** indeksidagi elementni olib tashlaydi va qaytaradi. Agar indeks ko‘rsatilmay faqat pop() holida foydalanilsa, oxirgi elementni olib tashlanadi.

***count(item):*** ro‘yxatdagi ***item*** elementining takrorlanishlar sonini qaytaradi. Agar mavjud bo‘lmas, 0 qaytaradi;

***sort([key]):*** ro‘yxat elementlarini tartiblaydi. Standart bo‘yicha o‘sish tartibida saralanadi. Lekin ***key*** parametr yordamida saralashni qanday tartibda bo‘lishini boshqarish mumkin.

***reverse():*** Roʻyxatdagi barcha elementlarni teskari tartibda qayta joylashtiradi.

***copy():*** roʻyxatni nusxalaydi.

Bundan tashqari, Python ro‘yxatlar bilan ishlash uchun bir qator standart funktsiyalarni taqdim etadi:

***len(list):*** ro‘yxat uzunligini aniqlash;

***sorted(list, [key]):*** tartiblangan roʻyxatni qaytaradi;

***min(list):*** ro‘yxatning eng kichik elementini topish;

***max(ro‘yxat):*** ro‘yxatdagi eng katta elementni topish.

**Ro‘yxatga element qo‘shish uchun append()** , **extension** va **insert** metodlari, ro‘yxatdan elementni olib tashlash uchun **remove()**, **pop()** va **clear()** usullari qo‘llaniladi.

**Foydalanish usullari:**

list1= ["Timur", "Bobur"]

*# roʻyxat oxiriga qoʻshish*

list1.**append**("Elmurod") #["Timur", "Bobur", "Elmurod"]

*# ikkinchi pozitsiyaga qo‘shish*

list1.**insert**(1, "Bilol") #["Timur", " Bilol", "Bobur", "Elmurod"]

*# elementlar to‘plamini qo‘shish ["Malik", "Samat"]*

list1.**extend**(["Malik", "Samat"])

# ["Timur", " Bilol", "Bobur", "Elmurod", "Malik", "Samat"]

*# element indeksini olish*

index\_of\_tom = list1.**index**("Bobur") # 2

*# ushbu indeksda o‘chirish*

removed\_item = list1.**pop**(index\_of\_tom)

# ["Timur", " Bilol", "Elmurod", "Malik", "Samat"]

*# oxirgi elementni olib tashlang*

oxirgi\_element = list1.**pop**()  # "Samat"

print(list1) # ["Timur", " Bilol", "Elmurod", "Malik"]

*# "Elmurod" elementini olib tashlash*

list1.**remove**("Elmurod")

print(list1)         # ["Timur", " Bilol", "Malik"]

# barcha elementlarni o‘chirish

list1.**clear**()

print (list1)        # []

**IN kalit so‘zining ishlatirlishi**

Agar biron bir element topilmasa, olib tashlash va indekslash usullari xatolik qaytaradi. Bunday vaziyatni oldini olish uchun elementni ishlatishdan oldin **in** kalit so‘zidan foydalanib uning mavjudligini tekshirish lozim:

list\_odamlar = ["Timur", "Bobur", "Samat"]

**if** "Ali" **in** list\_odamlar:

    list\_odamlar.**remove**("Ali")

print(list\_odamlar) # ["Timur", "Bobur", "Samat"]

Odamlar ro‘yxatida "Ali" elementi bo‘lsa, **if "Ali" in list\_odamlar** ifodasi "True" qiymati qaytaradi. Shunday ekan bu vaziyatda False qiymati qaytadi va list\_odamlar.remove(“Ali”) funksiyasi bajarilmaydi va list\_odamlar nomli ro‘yxat tarkibi o‘zgarishsiz qoladi.

**DEL metodi**

Pythonda ro‘yxatdan elementni o‘chirish uchun yana bir **del** nomli operatordan ham foydalaniladi. O‘chiriladigan element yoki elementlar to‘plami ushbu operatorga parametr sifatida uzatiladi:

list\_odamlar = ["Timur", " Bilol", "Bobur", "Elmurod", "Malik", "Samat"]

*# ikkinchi elementni olib tashlash*

**del** list\_odamlar [1]

print (list\_odamlar) # ["Timur", "Bobur", "Elmurod", "Malik", "Samat"]

*#   to‘rtinchi elementgacha o‘chirish*

list\_odamlar **[:3]**

print(list\_odamlar) #["Elmurod", "Malik", "Samat"]

**del** list\_odamlar**[1:]**#ikkinchi elementdan oxirigacha

print(list\_odamlar) #["Timur"]

**COUNT() metodi**

Agar element roʻyxatda necha marta borligini bilish kerak boʻlsa, count() metodidan foydalanish maqsadga muvofiq bo‘ladi:

odamlar = ["Bobur", " Bilol", "Bobur", "Elmurod", "Malik", "Bobur"]

odamlar\_soni = odamlar.**count**("Bobur")

print(odamlar\_soni)       # 3

**SORT() metodi**

Sort() metodi standart holatda o‘sish tartibida saralash uchun ishlatiladi:

odamlar = ["Timur", "Bilol", "Bobur", "Elmurod", "Malik", "Samat"]

odamlar.**sort**()

print(odamlar)

#['Bilol','Bobur','Elmurod','Malik','Samat', 'Timur']

Agar ma’lumotlarni teskari tartibda saralash zarur bo‘lsa, sort() metodidan keyin reverse() usulini qo‘llash mumkin:

odamlar = ["Timur", "Bilol", "Bobur", "Elmurod", "Malik", "Samat"]

odamlar.**sort**()

odamlar.**reverse**()

print(odamlar)

# ['Timur', 'Samat', 'Malik', 'Elmurod', 'Bobur', 'Bilol']

Saralash aslida ikkita ob’ektni taqqoslaydi va qaysi biri "kichik" bo‘lsa, "katta" ob’ektdan oldin joylashtiriladi. "katta" va "kichik" tushunchalari albatta nisbiydir. Va agar ro‘yxatning barcha elementlari raqamlardan tarkib topgan bo‘lsa, unda barchasi oddiy - raqamlar o‘sish tartibida joylashtiriladi. Agar ro‘yxatda satrlar va boshqa ob’ektlar mavjud bo‘lsa va saralash kerak bo‘lsa, unda vaziyat yanada murakkabroq bo‘ladi. Bunday holda harflar sonllardan oldin joylashtiriladi. Albatta sonli satrlar haqida gap ketmoqda. Chunki butun tipli ma’lumot bilan string tipli ma’lumot solishtirish xatolik keltirib chiqaradi. Demak, string tiplarni solishtirishda, agar ularning birinchi belgilar teng bo‘lsa, ikkinchi belgilar solishtiriladi va hokazo. Bunda raqamli belgi alifbodagi bosh harfdan "kichik" deb hisoblanadi va o‘z navbatida bosh harf ham kichik harfdan kichik hisoblanadi.

Shunday qilib, agar ro‘yxat tarkibida katta va kichik harflar mavjud bo‘lsa va saralash amali bajarilsa, unda unchalik to‘g‘ri bo‘lmagan natijalarni olinishi mumkin, chunki asosan "bob" matni "Tom" matnidan oldin kelishi kerak. Bunday holda sort metodining standart tartiblashini o‘zgartirish uchun unga parametr sifatida funktsiyani beriladi:

odamlar = ["Timur", "bilol", "Bobur", "elmurod", "Malik", "Samat"]

odamlar.**sort**()   *# standart tartiblash*

print(odamlar) #['Bobur', 'Malik', 'Samat', 'Timur', 'bilol', 'elmurod']

odamlar.**sort(key=str.lower)**

*# kichik katta-kichik tartiblash*

print ( odamlar )

# ['bilol', 'Bobur', 'elmurod', 'Malik', 'Samat', 'Timur']

**SORTED metodi**

Saralash usuliga qo‘shimcha ravishda ikkita shaklga ega bo‘lgan tartiblangan standart funksiyadan foydalanish mumkin:

***sorted(list):*** ***list*** ro‘yxatini tartiblaydi

***sorted(list, key):*** ***list*** ro‘yxati elementlariga ***key*** funksiyasini qoʻllash orqali roʻyxatni tartiblaydi

list\_per = ["Tom", "bob", "alis", "Sem", "Bill"]

sorted\_per = ***sorted***(list\_per, key=str.lower)

print (sorted\_per)

# ["alisa", "Bill", "bob", "Sem", "Tom"]

Ushbu funktsiyadan foydalanganda shuni yodda tutish kerakki, bu funksiya tartiblangan ro‘yxatni o‘zgartirmaydi, ammo u barcha tartiblangan elementlarni yangi ro‘yxatga joylashtiradi va natijani qaytaradi.

**MIN va MAX funksiyalari**

Pythonning **min()** va **max()** funksiyalari mos ravishda minimum va maksimum qiymatlarni topishga imkon beradi:

raqamlar = [9, 221, 12, 10, 3, 125, 18]

print (**min**(raqamlar))      # 3

print (**max**(raqamlar))      # 221

**COPY() funksiyasi**

Ro‘yxatlarni nusxalashda, ro‘yxatlar o‘zgaruvchan tur ekanligini yodda tutish kerak. Shuning uchun ikkala o‘zgaruvchi ham bir xil ro‘yxatga ishora qilsa, bitta o‘zgaruvchini o‘zgartirish boshqa o‘zgaruvchiga ta’sir qiladi:

list\_per = ["Timur", "Bobur", "Alisey"]

list\_per2 = list\_per

list\_per2.**append**("Samat") # elementni ikkinchi ro‘yxatga qo‘shish

# kishi1 va odamlar2 bir xil roʻyxatga ishora qiladi

print(list\_per)    # ["Timur", "Bobur", "Alisey", " Samat"]

print(list\_per2)    # ["Timur", "Bobur", "Alisey", " Samat"]

Bu yuqorida keltirilgan dasturdagi bir o‘zgaruvchining qiymatini ikkinchisiga yuklab qo‘yish holati nusxalash deb atalmaydi. Nushalash maqsadida bunday hatti harakatni qilmagan ma’qul. Elementlarni nusxalash uchun chuqurroq nusxalash talab qilinadi. Shundagina bir vaqtning nusxalangan va aslnusxadagi ro‘yxatlar qiymatlari bir xil bo‘lsada, aslida turli xil ro‘yxatlarni ifodalaydi. Buning uchun oddiygina **copy()** metodidan foydalanib nusxalash kerak bo‘ladi:

list\_per = ["Tom", "Bob", "Alis"]

list\_per2 = list\_per.**copy**() # elementlarni list\_per dan list\_per2 ga nusxalash

# elementni FAQAT ikkinchi ro‘yxatga qo‘shish

list\_per2.**append**("Sem")

# list\_per va list\_per2 turli roʻyxatlarni bildiradi

print(list\_per)    # ["Tom", "Bob", "Alis"]

print(list\_per2)    # ["Tom", "Bob", "Alis", "Sem"]

**Ro‘yxatning bir qismini nusxalash**

Agar butun ro‘yxatni emas, balki uning ma’lum bir qismini nusxalash kerak bo‘lsa, unda quyidagi shakllarni qabul qilishi mumkin bo‘lgan maxsus sintaksisdan foydalanish kerak bo‘ladi:

***list[:end]:*** roʻyxat boshidan ***end*** elementigacha bo‘lgan oraliqni tanlash.

***list[start:end]:*** ro‘yxatning ***start*** va ***end*** elementlari oralig‘ini belgilaydi

***list[start:end:step]:*** ro‘yxatning ***start*** va ***end*** elementlari oralig‘ini ***step*** qadam bilan belgilaydi. Odatiy bo‘lib, bu ***step*** parametr 1 ga teng bo‘ladi.

list\_per = ["Tomson", "Bobi", "Elison", "Semmi", "Timati", "Billi"]

slice\_people1 = list\_per **[:3]** # 0 dan 3 gacha

print(slice\_people1)    # ['Tomson', 'Bobi', 'Elison']

slice\_people2 = list\_per **[1:3]** #1 dan 3 gacha

print(slice\_people2)    # ["Bobi", "Elison"]

slice\_people3 = list\_per**[1:6:2]** #1 dan 6 gacha 2 ga oshib

print(slice\_people3)    #["Bobi", "Semmi", "Billi"]

**Manfiy indeks**

Ro‘yxatning oxirgi elementidan boshlab murojaatni boshlash kerak bo‘lganda manfiy indeksdan foydalanish kerak bo‘ladi. Masalan, -1 – element oxirgi element, -2 – element oxiridan bitta oldingi va hokazo.

list\_per  = ["Tomson", "Bobi", "Elison", "Semmi", "Timati", "Billi"]

*# nolinchidan oxirgisigacha, oxirgi xisobga kirmaydi*

slice\_people1 = list\_per**[:-1]**

print(slice\_people1)

*# ['Tomson', 'Bobi', 'Elison', 'Semmi', 'Timati']*

*# oxirgi uchinchi elementdan oxirgisigacha, oxirgisini olmaydi*

slice\_people2 = list\_per **[-3:-1]**

print(slice\_people2)    # ["Semmi", "Timati"]

**Ro‘yxatlarni ulash**

Qo‘shish (+) operatori ro‘yxatlarni birlashtirish uchun ishlatiladi:

people1 = ["Tomson", "Bobi", "Alison"]

people2 = ["Tomson", "Semmi", "Timati", "Billi"]

people3 = people1 + people2

print (people3)

*# ["Tomson", "Bobi", "Elison", "Tomson", "Semmi", "Timati", "Billi"]*

**Ichma-ich Ro‘yxatlar**

Ro‘yxatlar satrlar, raqamlar kabi standart ma’lumotlardan tashqari, boshqa ro‘yxatlarni ham o‘z ichiga olishi mumkin. Bunday ro‘yxatlarni jadvallar bilan bog‘lash mumkin, bu yerda ichki ro‘yxatlar qatorlar vazifasini bajaradi. Masalan:

odamlar = [

    ["Tomson", 29],

    ["Elison", 33],

    ["Bobi", 27]

]

print(odamlar[0])          *# ["Tomson", 29]*

print(odamlar[0][0])       *# "Tomson"*

print(odamlar[0][1])       *# 29*

Ichki ro‘yxatning elementiga murojaat qilish uchun bir juft indeksdan foydalanish kerak: odamlar[0][1] - birinchi joylashtirilgan ro‘yxatning ikkinchi elementiga murojaat.

Umumiy ro‘yxatni, shuningdek, ichki ro‘yxatlarni qo‘shish, o‘chirish va o‘zgartirish oddiy (bir o‘lchovli) ro‘yxatlar kabi amalga oshiriladi:

odamlar = [

    ["Tomson", 29],

    ["Elison", 33],

    ["Bobi", 27]

]

*# ichki ro‘yxat yaratish*

odam = **list**()

odam.**append**("Billi")

odam.**append**(41)

*# ichki ro‘yxat qo‘shish*

odamlar.**append**(odam)

print(odamlar[-1])   # ["Billi", 41]

*# ichki roʻyxatga qoʻshish*

odamlar[-1].**append**("+79996543210")

print(odamlar[-1])  # ["Billi", 41, "+79996543210"]

*# oxirgi elementni o‘rnatilgan ro‘yxatdan olib tashlash*

odamlar**[-1].pop**()

print(odamlar[-1])          # ["Billi", 41]

*# oxirgi kiritilgan roʻyxatni butunlay oʻchirish*

odamlar.**pop(-1)**

*# birinchi elementni o‘zgartirish*

odamlar[0] = ["Semmi", 18]

print (odamlar)

*# [ ["Semmi", 18], ["Alison", 33], ["Bobi", 27]]*

Ichki ro‘yxatlar ustida takrorlash:

odamlar = [

    ["Tomson", 29],

    ["Elison", 33],

    ["Bobi", 27]

]

**for** odam **in** odamlar:

**for** item **in** odam:

print(item, end=" | ")

Konsol chiqishi :

Tomson | 29 | Elison | 33 | Bobi | 27 |

**Nazorat savollari:**

1. Ro‘yxat deb nimaga aytiladi?
2. Ro‘yxatlarni qanday yaratiladi?
3. Ro‘yxatlarning elementlari bir turga mansub bo‘lishi shartmi?
4. ro‘yxat elementlariga murojaatlar qanday amalga oshiriladi?
5. Ro‘yxat uzunligini qanday aniqlanadi?
6. Ro‘yxat elementini biro songa ko‘paytirish nimani beradi?
7. Manfiy index tushunchasi nimani anglatadi?
8. tomson, bobi, semmi = odamlar – ushbu ifodaning ma’nosi nima?
9. Ro‘yxatga ma’lumotni sikl orqali kiritish qanday amalga oshiriladi?
10. Ro‘yxat elementlarini saralash uchun qaysi metodni ishlatiladi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**7-mashg‘ulot.** Pythonda Kortejlar (Tupllar) bilan ishlash.

**O‘quv savollari:**

1. Kortejlar va ularning qo‘llanilishi.
2. Kortejlarni yaratish usullari.
3. Kortejlar bilan ishlovchi metodlar.

**1. Kortejlar va ularning qo‘llanilishi.**

Kortej xuddi ro‘yxat kabi elementlar ketma-ketligini ifodalaydi. Bundan tashqari kortej o‘zgarmas tur hisoblanadi. Shuning uchun kortejdagi elementlarni qo‘shish, olib tashlash yoki uni o‘zgartirishning ilojisi yo‘q.

**2. Kortejlarni yaratish usullari.**

Tuple yaratish uchun turli qiymatlarni vergul bilan ajratib oddiy qavslar ichiga joylab qo‘yish kifoya:

tommi= ("Tomson",23)

print (tommi)      # ("Tomson", 23)

Shuningdek, kortejni yaratish uchun qavslarsiz vergul bilan ajratilgan qiymatlarni ro‘yxatga olish kerak bo‘ladi:

tommi = "Tomson", 23

print (tommi)      # ("Tomson", 23)

Agar mabodo kortej bitta elementdan iborat bo‘lib qolsa, u holda kortejning yagona elementidan keyin vergul qo‘yish kerak:

tommi = ("Tomson")

Roʻyxat kabi boshqa elementlar toʻplamidan kortej yaratish uchun roʻyxatni **tuple()** funksiyasiga uzatiladi, bu esa ro‘yxatni kortejni aylantiradi va kortej qaytaradi:

data = ["Tomson", 37, "Google"]

tommi = tuple (data)

print(tommi)       # ("Tomson", 37, "Google")

**3. Kortejlar bilan ishlovchi metodlar**

**LEN() funksiyasi**

Pythonning standart len() funktsiyasidan foydalanib, kortej uzunligini aniqlash mumkin:

tommi = ("Tomson", 37, "Google")

print (**len**(tommi))      # 3

**Kortej elementlariga murojaat**

Kortejdagi elementlarga murojaat roʻyxatdagi kabi indeks boʻyicha amalga oshiriladi. Indekslash, shuningdek, ro‘yxat boshidan elementlarni olishda noldan va ro‘yxat oxiridagi elementlarni olishda -1 dan boshlanadi:

tommi = ("Tomson", 37, "Google", "software enginireeng")

print(tommi[0]) # "Tomson"

print(tommi[1]) # 37

print(tommi[-1]) # "software enginireeng"

Lekin kortej o‘zgarmas tur bo‘lgani uchun uning elementlarini o‘zgartirishning ilojisi yo‘q. Ya’ni, quyidagi yozuv ishlamaydi:

tommi[1] = "Tomsonjon"

Agar zarurat tug‘ilsa, kortejni alohida o‘zgaruvchilarga ajratish mumkin:

name, age, company, position = ("Tomson", 35, "Google", "software enginireeng")

print(name) # Tomson

print(age) # 35

print(position) # software enginireeng

print(company) # Google

**Subtuplelarni olish**

Ro‘yxatlarda bo‘lgani kabi, kortejning bir qismini boshqa kortej sifatida olish mumkin:

tommi = ("Tomson", 35, "Google", "software enginireeng")

# 1 dan 3 gacha bo‘lgan elementni olish, 3-element xisoblanmaydi

print(tommi[1:3])      # (35, “Google”)

# 0 dan 3 gacha bo‘lgan elementni olish, 3-element xisoblanmaydi

print(tommi[:3])      # ("Tomson", 35, "Google")

# 1 dan oxirgi elementgacha subtuple olish

print (tommi[1:])

# (35, "Google", "software enginireeng")

**Kortejni funksiyaga parametr sifati berish va natijasi sifatida qabul qilish**

Funktsiyadan bir vaqtning o‘zida bir nechta qiymatlarni qaytarish kerak bo‘lganda, kortejlardan foydalanish qulaydir. Funktsiya bir nechta qiymatlarni qaytarsa, u aslida kortejni qaytaradi:

**def** get\_user():

    name="Tomson"

    yosh = 25

    kompaniya = "Google"

    return name, yosh, kompaniya

foydalanuvchi = get\_user()

print(foydalanuvchi)      # ("Tomson", 25, "Google")

**\*** operatori yordamida funktsiyaga kortejni uzatishda uni funktsiya parametrlariga o‘tkaziladigan individual qiymatlarga ajratish mumkin:

**def** print\_person(ism, yosh, kompaniya):

print(f"Ism:{ism}Yosh:{yosh} Kompaniya: {kompaniya}")

tommi = ("Tomson", 25)

print\_person(\*tommi, "Microsoft")

# Ism: Tomson   Yosh: 25   Kompaniya: Microsoft

bob = ("Bobi", 41, "Apple")

print\_person(\*bob)

# Ism: Bobi   Yoshi: 41   Kompaniya: Apple

**Kortejda sikldan foydalanish**

To‘plam elementlariga murojaat qilishda ularning har bir elementiga alohida murojaat qilish, o‘ta noto‘g‘ri bo‘ladi. Buning uchun standart for va while sikllaridan foydalaniladi. **for** tsikli orqali kortej elementlarini olish quyidagicha amalga oshiriladi:

tom = ("Tomson", 22, "Google")

**for** element **in** tom:

    print (element)

While siklidan foydalanib kortej elementlariga murojaat qilish:

tom = ("Tomson", 22, "Google")

i = 0

**while** i <**len**(tom):

    print (tom[i])

    i += 1

**Kortej tarkibida element mavjudligini tekshirish**

Ba’zi masalalarda elementga murojaat qilish kerak bo‘ladi va murojjat natijasi xatolik bilan tugashi mumkin. Bunday hollarda elementni mavjudligini birinchi tekshirib olish kerak bo‘ladi va keyin biror amal bajarilsa, maqsadga muvofiq bo‘ladi. Ushbu murojaat amalga oshiriladigan elementni mavjudligini tekshirish uchun in operatoridan foydalaniladi va u quyidagicha foydalaniladi:

foydalanuvchi = ("Tomson", 22, "Google")

nomi="Tomson"

**if** nomi **in** foydalanuvchi:

**print**("Foydalanuvchi nomi Tomson")

**else**:

**print**("Foydalanuvchining nomi boshqacha ekan!")

**Nazorat savollari:**

1. Kortej deb nimaga aytiladi?
2. Kortejlarning ro‘yxatlardan farqi nimada?
3. Kortejlarning yaratilishi usullari qanday?
4. Kortejdagi ma’lumotlarga qanday murojaat qilinadi?
5. Kortejdagi ma’lumotlarni qanday qilib o‘zgartirish mumkin?
6. Parametr va funksiya natijasi sifatida kortejni qanday qo‘llash mumkin?
7. Kortej elementlarini tsikl orqali chiqarish qanday amalga oshiriladi?
8. Kortejni parametr sifatida uzatishda “\*” (yulduzcha) belgisining ro‘li qanday?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**8-mashg‘ulot.** Pythonda Setlar ro‘yxati bilan ishlash.

**O‘quv savollari:**

1. Setlar va ularning qo‘llanilishi.
2. Setlarni yaratish usullari.
3. Setlar bilan ishlovchi metodlar.

**1. Setlar va ularning qo‘llanilishi.**

**To‘plam (set)** shunday to‘plamki, uning har bir elementi to‘plamda yagona, takrorlanmas hisoblanadi.Bunday turdagi to‘plamnianiqlash uchun jingalak qavslardan foydalaniladi:

set\_list= {"Tomson", "Bobbi", "Alisey", "Tomson"}

print(set\_list)  # {"Tomson", "Bobbi", "Alisey"}

E’tibor berish kerakki, print funktsiyasi yordamida to‘plamni ekranga chiqarilganda "Tomson" elementini bir marta chiqarildi. Ammo e’lon qilinganda "Tomson" ham, to‘plam ta’rifi ushbu elementni ikki marta o‘z ichiga oladi.

**2. Setlarni yaratish usullari**

Buning sababi shundaki, to‘plam faqat yagona qiymatlarni o‘z ichiga oladi.

Bundan tashqari, to‘plamni aniqlash uchun elementlar ro‘yxati yoki to‘plami uzatiladigan **set()** funktsiyasidan foydalanish mumkin**:**

set\_persons = ["Maya", "Billi", "Teddi"]

set\_users = **set**(set\_persons)

**print**(set\_users)     # {"Maya", "Billi", "Teddi"}

**set()** funktsiyasi bo‘sh to‘plam yaratish uchun quyidagicha foydalaniladi:

set\_users= **set**()

**3. Setlar bilan ishlovchi metodlar**

Boshqa turdagi toplamlar singari ***set()*** to‘plamida ham uzunligini aniqlovchi ***len()*** funksiyasi mavjud:

foydalanuvchilar = {"Tomson", "Bobi", "Alison"}

**print** (**len**(foydalanuvchilar))        # 3

**add() metodi**

To‘plamga bitta element qo‘shish uchun **add()** metodidan foydalaniladi:

set\_users = **set**()

set\_users.**add**("Sammi")

**print** (set\_users) # "Sammi"

**remove() metodi**

**Element olib tashlash.** Bitta elementni olib tashlash **remove()** metodi yordamida amalga oshiriladi va olib tashlanadigan element qaytariladi. Ammo shuni yodda tutish kerakki, agar bunday element to‘plamda bo‘lmasa, unda xatolik yuzaga keladi. Shuning uchun, o‘chirishdan oldin, in operatori yordamida element mavjudligini tekshirish kerak:

set\_users = {"Tomson", "Bobi", "Alison"}

set\_user = "Tomson"

**if** set\_user **in** set\_users:

    set\_users.**remove**(set\_user)

print(set\_users)     # {"Bobi", "Alison"}

**discard** () metodi

**Elementni olib tashlash uchun discard** () usulini ham qo‘llash mumkin. Ammo element mavjud bo‘lmasa, xatolik qaytariladi:

set\_users = {"Tomson", "Bobi", "Alison"}

set\_users.**discard**("Tim") *# element "Tim" mavjud emas va metod hech narsa qilmaydi*

print(set\_users)     *# {"Tomson", "Bobi", "Alison"}*

set\_users.**discard**("Tomson")  *# "Tomson" elementi mavjud va usul elementni olib tashlaydi*

print(set\_users)     *#   {"Bobi", "Alison"}*

**clear()**  metodi

Barcha elementlarni olib tashlash uchun **clear()** usuli chaqiriladi :

set\_users.**clear**()

**To‘plam elementlariga sikl yordamida murojaat qilish**

Elementlarga murojaat qilish uchun for siklidan foydalanish mumkin :

set\_users = {"Tomson", "Bobi", "Alison"}

**for** user **in** set\_users:

**print** (user)

Takrorlash paytida har bir element ***user*** o‘zgaruvchisiga uzatiladi va u o‘z navbatida ushbu elementlarni konsolga chiqaradi.

**copy** **() metodi**

**To‘plamlarni nusxalash. Copy** () usulidan foydalanib, bir to‘plam elementlarini boshqa o‘zgaruvchiga nusxalash mumkin:

set\_users = {"Tomson", "Bobi", "Alison"}

talabalar = set\_users.**copy**()

print (talabalar)      # {"Tomson", "Bobi", "Alison"}

**union() metodi**

**To‘plamlarni birlashtirish. union()** usuli ikkita to‘plamni birlashtiradi va yangi to‘plamni qaytaradi:

set\_users = {"Tomson", "Bobbi", "Alison"}

users2 = {"Semmi", "Keyt", "Bobbi"}

users3 = set\_users.**union** (users2)

print (users3)

# {"Tomson", "Bobi", "Alison","Semmi", "Keyt"}

**intersection() metodi**

**To‘plamlarning kesishishi.** To‘plamlarning kesishishi faqat ikkala to‘plamda bir vaqtning o‘zida bo‘lgan elementlarni olish imkonini beradi. **Intersection()** metodi to‘plamlar kesishish funksiyasini bajaradi va kesishmadan hosil bo‘lgan yangi to‘plamni qaytaradi:

set\_users = {"Tomson", "Bobbi", "Alison"}

users2 = {"Sammi", "Keyt", "Bobbi"}

users3 = set\_users.**intersection**(users2)

print (users3)    # {"Bobbi"}

**and** operatori

Kesishish usuli o‘rniga mantiqiy ko‘paytirish operatsiyasi (***and***) dan foydalanish ham mumkin:

set\_users = {"Tomson", "Bobbi", "Alison"}

users2 = {"Sammi", "Keyt", "Bobbi"}

print (set\_users and users2)    # {"Bobbi"}

Bu ikkala holda ham, bir xil natijaga erishiladi.

**intersection\_update()** metodi

**intersection\_update()** metodi birinchi to‘plamni kesishish elementlari bilan almashtiradi:

set\_users = {"Tomson", "Bobbi", "Alison"}

users2 = {"Sammi", "Keyt", "Bobbi"}

set\_users.**intersection\_update**(users2)

print(set\_users)    # {"Bobbi"}

**DIFFERENCE() metodi.**

Ushbu metod to‘plamlar orasidagi farqlanuvchi elementlarni qaytaradi. Ya’ni birinchi to‘plamda bo‘lgan, ammo ikkinchisida yo‘q bo‘lgan elementlarni qaytaradi. To‘plamlar farqini olish uchun difference usuli yoki ayirish operatsiyasidan foydalaniladi mumkin:

set\_users = {"Tomson", "Bobbi", "Alison"}

users2 = {"Sammi", "Keyt", "Bobbi"}

users3 = set\_users.**difference**(users2)

print(users3)            *# {"Tomson", "Alison"}*

print (set\_users - users2)    *# {"Tomson", "Alison"}*

**symmetric\_difference()**  metodi

To‘plam farqining alohida turi - simmetrik farq **symmetric\_difference()** usuli yoki ^ operatsiyasi yordamida amalga oshiriladi. U ikkala to‘plamning umumiy elementlardan tashqari barcha elementlarini qaytaradi:

set\_users = {"Tomson", "Bobbi", "Alison"}

users2 = {"Sammi", "Keyt", "Bobbi"}

users3 = set\_users.**symmetric\_difference**(users2)

print(users3)    *# {"Tomson", "Elison", "Sammi", "Keyt"}*

users4 = **set\_users ^ users2**

print(users4)    *# {"Tomson", "Alison", "Sammi", "Keyt"}*

**issubset** metodi

**issubset** metodi joriy toʻplam boshqa toʻplamining kichik toʻplami (yaʼni qismi) ekanligini aniqlash imkonini beradi:

set\_users = {"Tomson", "Bobbi", "Alison"}

users2={"Sammi", "Keyt", "Bobbi", "Alison", "Tomson"}

print (set\_users.**issubset**(users2))    # True

print(users2.**issubset**(set\_users))  # False

**issuperset** metodi

Agar joriy to‘plam boshqa to‘plamning super to‘plami (ya’ni o‘z ichiga olgan) bo‘lsa, **issuperset** usuli **True** qiymatini qaytaradi:

set\_users = {"Tomson", "Bobbi", "Alison"}

users2={"Sammi", "Keyt", "Bobbi", "Alison", "Tomson"}

print(set\_users.**issuperset**(users2))    *# False*

print (users2.**issuperset**(set\_users))    *# True*

**frozenset** matodi

**Muzlatilgan to‘plam** (*frozen set*) turidagi to‘plamlar o‘zgartirib bo‘lmaydigan to‘plamlar turiga kiradi. Uni yaratish uchun **frozenset** funksiyasidan foydalaniladi:

set\_user= **frozenset**({"Tommi","Bobbi","Elison"})

**Frozenset** funktsiyasiga elementlar to‘plami - ro‘yxat, kortej boshqa to‘plamlar uzatiladi.

  Bunday to‘plamga yangi elementlarni qo‘shish, undagi mavjud elementlarini o‘chirib tashlash kabi amallarni bajarishning ilojisi yo‘q. Aynana shuning uchun **frozenset** to‘plami  operatsiyalarning cheklangan to‘plamini qo‘llab-quvvatlaydi:

***len(s):*** to‘plam uzunligini aniqlash;

***x in s*** : agar ***s*** to‘plamida ***x*** elementi mavjud bo‘lsa, True qiymatini qaytaradi

***x not in s:*** agar ***x*** ***s*** to‘plamining tarkibida bo‘lmasa, True qaytaradi

***s.issubset(t):*** agar ***t s*** toʻplamini oʻz ichiga olgan boʻlsa, True qiymatini qaytaradi

***s.issuperset(t):*** agar ***t s*** ichida bo‘lsa, True qiymatini qaytaradi

***s.union(t):*** ***s*** va ***t*** to‘plamlarning birlashuvini qaytaradi

***s.intersection(t):*** ***s*** va ***t*** ning kesishishini qaytaradi

***s.difference(t):*** ***s*** va ***t*** to‘plamlari orasidagi farqni qaytaradi

***s.copy():*** ***s*** to‘plamining nusxasini qaytaradi

**Nazorat savollari:**

1. To‘plamlarni yaratishning qanday usullarini bilasiz?
2. set() metodining vazifasi qanday?
3. To‘plamlarni lug‘atlardan farqi nimada?
4. To‘plamga element qo‘shish qanday amalga oshiriladi?
5. To‘plamga element olib tashlash qanday amalga oshiriladi?
6. **discard**() metodining vazifasi qanday?
7. **clear**() metodining vazifasi qanday?
8. **copy**() metodi nima maqsadda ishlatiladi va uning afzallik hamda kamchiliklari qanday?
9. **union()** metodi haqida gapirib bering.
10. **intersection()** qanday metod?
11. **intersection\_update()** metodining vazifasi va **intersection()** metodidan farqi haqida gapirib bering.
12. **symmetric\_difference()**  metodi haqida gapirib bering.
13. **issubset()** – metodining ishlash prinsipi qanday?
14. **issuperset()** – metodidan qanday foydalaniladi?
15. **frozenset()** funksiyasining vazifasi nima?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**9-mashg‘ulot.** Pythonda “Lug‘at” bilan ishlash.

O‘quv savollari:

1. Lug‘atlar va ularning qo‘llanilishi.
2. Lug‘atlarni yaratish usullari.
3. Lug‘atlar bilan ishlovchi metodlar.

**1. Lug‘atlar va ularning qo‘llanilishi.**

**Pythondagi lug‘at (dictionary)**  elementlar to‘plamini saqlaydi. Lug‘at tarkibidagi har bir element o‘ziga xos, yagona kalitga va u bilan bog‘liq bo‘lgan qiymatga ega bo‘ladi**.**

Lug‘at ta’rifi quyidagi sintaksisga ega:

dictionary={kalit1: qiymat1, kalit2: qiymat2, ....}

Lug‘atning barcha elementlari jingalak qavslar ichiga joylanadi. Har bir element bir-biri bilan vergul orqali ajratildi. Har bir elementning yagona kaliti va unga bog‘langan qiymat mavjud bo‘ladi. Elementning kaliti birinchi keladi va undan keyin qiymat keladi. Kalit va qiymatini ikki nuqta ajratib turadi.

**2. Lug‘atlarni yaratish usullari.**

**Lug‘atni aniqlash:**

users = {1: "Tomson", 2: "Bobi", 3: "Billi"}

users lug‘ati raqamlarni kalit sifatida va satrlarni qiymat sifatida ishlatadi. Ya’ni, 1-kalitga ega bo‘lgan element "Tomson" qiymatiga teng, 2-kalitga ega element "Bobi" qiymatiga ega va hokazo.

Lug‘atdagi elementlar tarkibi bir xil tipli ma’lumotlar bo‘lishi ham mumkin. Masalan:

e\_mails = {"tomson@gmail.com": "Tomson", "bobi@gmail.com": "Bobi", "semmi@gmail.com": "Semmi"}

e\_mails lug‘atida kalit sifatida satrlardan foydalanildi - foydalanuvchi elektron pochta manzillari va qiymat sifatida ham satrlardan - foydalanuvchi nomlaridan foydalanildi.

Lekin kalitlar va satrlar bir xil turdagi bo‘lishi shart emas. Ular turli xil turlarni ko‘rsatishi mumkin:

objects = {1: "Tomson", "2": False, 3: 150.64}

Bundan tashqari, umuman elementsiz, bo‘sh lug‘atni aniqlash ham mumkin:

dict\_object = {}

yoki shunga o‘xshash:

dict\_object = dict()

**Ro‘yxatlar va kortejlarni lug‘atga aylantirish**

Lug‘at va ro‘yxat tuzilmaviy jihatdan bir-biriga o‘xshamaydigan turlar bo‘lsa-da, standart **dict()** funksiyasi yordamida ma’lum turdagi ro‘yxatlarni lug‘atga aylantirish mumkin. Buning uchun ro‘yxat ichki ro‘yxatlar to‘plamini saqlashi kerak. Har bir ichki ro‘yxat ikkita elementdan iborat bo‘lishi kerak - lug‘atga aylantirilganda birinchi element kalitga, ikkinchisi esa qiymatga aylanadi:

user\_list=[

    ["+554513455", "Tomson"],

    ["+002549557", "Bobi"],

    ["+445454512", "Elison"]

]

user\_dict = dict(user\_list)

print(user\_dict)  *# {"+554513455", "Tomson:", "+002549557", #"Bobi", "+445454512": "Elison"}*

Xuddi shunday, ikkita kortej lug‘atga aylantirilishi mumkin, o‘z navbatida lug‘at ikkita kortejni o‘z ichiga oladi:

user\_tuple = (

    ("+554513455", "Tomson"),

    ("+002549557", "Bobi"),

    ("+445454512", "Elison")

)

user\_dict = **dict** **(user\_tuple)**

print (user\_dict)

**Lug‘at elementlariga murojaat qilish va o‘zgartirish**

Lug‘at elementlariga murojaat qilish uchun uning nomidan keyin element kaliti kvadrat qavs ichida ko‘rsatiladi:

Dictionary\_name[key]

Lug‘atdagi elementlarni olish va o‘zgartirish uchun:

dict\_users = {

    "+11002255": "Tomson",

    "+15245999": "Bobi",

    "+15415154": "Elison"

}

# "+11002255" kaliti bilan elementni olish

print (dict\_users ["+11002255"])       # Tomson

# element qiymatini "+15245999" tugmasi bilan belgilash

dict\_users["+15245999"] = "Bob Smit"

print (dict\_users["+15245999"])    # "Bob Smit"

Agar bunday kalit bilan elementning qiymatini belgilashda lug‘atda hech qanday element bo‘lmasa, u qo‘shiladi:

 dict\_users ["+5555555"] = "Semmi"

Ammo agar lug‘atda mavjud bo‘lmagan kalit bilan qiymat olishga harakat qilinsa, Python KeyError xatoligini chiqaradi:

foydalanuvchi=foydalanuvchilar["+4444444"] # KeyError

Bunday holatni oldini olish maqsadida elementga murojaat qilishdan oldin "key in dict" ifodasi yordamida lug‘atda element mavjudligini tekshirish mumkin. Agar kalit lug‘atda bo‘lsa, u holda bu ifoda "True" ni qaytaradi:

key = "+00000044"

**if** key **in** dict\_users:

    dict\_user = dict\_users[key]

    print(dict\_user)

**else**:

    print ("Element topilmadi")

Shuningdek lug‘at elementlariga murojaat qilish uchun ***get*** metodidan ham foydalaniladi.

**3. Lug‘atlar bilan ishlovchi metodlar**

***get(key):*** lug‘atdan kalit tugmasi bo‘lgan elementni qaytaradi. Agar ushbu kalitga ega bo‘lgan element mavjud bo‘lmasa, u holda "False" qiymati qaytariladi;

***get(key, default):*** lug‘atdan key kaliti bo‘lgan elementni qaytaradi. Agar bunday kalitga ega element bo‘lmasa, u ***default*** qiymatni qaytaradi.

foydalanuvchilar = {

    "+11002255": "Tomson",

    "+15245999": "Bobi",

    "+15415154": "Elison"

}

user1 = users.get("+15415154")

print (user1)     # Alison

user2=users.**get**("+15245999","Noma’lum foydalanuvchi")

print (user2 )     #Bobi

user3=users.**get**("+0000000", "Noma’lum foydalanuvchi")

print(user3)     # Noma’lum foydalanuvchi

**DEL funksiyasi**

**Del** funksiyasi elementni kalit bilan olib tashlash uchun ishlatiladi :

foydalanuvchilar = {

    "+11002255": "Tomson",

    "+15245999": "Bobi",

    "+15415154": "Elison"

}

**del** foydalanuvchilar["+15415154"]

print (foydalanuvchilar)

*# { "+11002255": "Tomson", "+15245999": "Bobi"}*

Ammo shuni yodda tutish kerakki, agar bunday kalit lug‘atda bo‘lmasa, KeyError xatoligi qaytariladi. Shuning uchun, o‘chirilishi kerak bo‘lgan elementni oldin, berilgan kalitli element mavjudligini tekshirish tavsiya etiladi.

dict\_user = {

    "+11002255": "Tomson",

    "+15245999": "Bobi",

    "+15415154": "Elison"

}

key = "+99990000"

**if** key **in** dict\_user:

**del** dict\_user[key]

    print(f"{key} kaliti element olib tashlandi")

**else**:

    print (f"{key} kalitli element topilmadi")

**POP() metodi**

O‘chirishning yana bir usuli - **pop**() usuli. Pop metodidan ikki xil ko‘rinishda foydalanish mumkin:

1. ***pop(key):*** ***key*** kalit berilgan elementni olib tashlaydi va olib tashlangan elementni qaytaradi. Agar berilgan kalit bilan element topilmasa, KeyError xatoligi qaytariladi.
2. ***pop(key, default):***  ***key*** kalitli elementni o‘chiradi va olib tashlangan elementni qaytaradi. Agar berilgan kalit bilan element bo‘lmasa, ***default*** qiymati qaytariladi.

foydalanuvchilar = {

    "+11002255": "Tomson",

    "+15245999": "Bobi",

    "+15415154": "Elison"

}

key = "+11002255"

user1 = users.pop (key)

print (user1)      # Tomson

user1 = users.pop("+00000000", "Noma’lum foydalanuvchi")

print (user1)      # Noma’lum foydalanuvchi

Agar barcha elementlarni olib tashlash kerak bo‘lsa, ya’ni bir so‘z bilan aytganda tozalash kerak bo‘lsa, unda **clear()** metodidan foydalanish kerak bo‘ladi:

per.clear()

**Lug‘atlarni nusxalash va birlashtirish**

**copy()** usuli lug‘at tarkibini nusxalab, yangi lug‘atni qaytaradi:

foydalanuvchilar={"998888":"Tomson","998771": "Bobi"}

talabalar = users.copy()

print (talabalar) # {"998888": "Tomson", "998771": "Bobi"}

**update()** metodi ikkita lug‘atni birlashtiradi:

foydalanuvchilar={"998888":"Tomson","998771": "Bobi"}

users2 = {"222222": "Semmi", "666000": "Kail"}

users.update(users2)

print(users)    *# {"998888": "Tomson", "998771": "Bobi", "222222": "Semmi", "666000": "Kail"}*

print(users2)    *# {"222222": "Semmi", "666000": "Kail"}*

Bunday holda, users2 lug‘ati o‘zgarishsiz qoladi. Faqat foydalanuvchilarning lug‘ati o‘zgartiriladi, unga boshqa lug‘at elementlari qo‘shiladi. Ammo agar ikkala manba lug‘ati ham o‘zgarishsiz qolishi kerak bo‘lsa va birlashmaning natijasi uchinchi lug‘at bo‘lsa, avval bitta lug‘atni boshqasiga nusxalanishi kerak:

users3 = users.**copy**()

users3.**update**(users2)

**Lug‘atni takrorlash**

Lug‘at tarkibini sikldan foydalanib olish uchun for siklidan foydalanish mumkin:

users = {"998888": "Tomson", "998771": "Bobi"}

**for** a **in** users:

    print(f"id: {a}   Foydalanuvchi: {users[a]} ")

Takrorlash orqali elementlarga murojaat qilishda joriy elementning kalitini elementning o‘zini olish uchun ishlatiladi.

Elementlarni olishning yana bir usuli - **items()** metodidan foydalanish**:**

users = {"998888": "Tomson", "998771": "Bobi"}

**for** **key**, **val** **in** users.**items**():

    print(f"id: {**key**}   Foydalanuvchi: {**val**} ")

**items()** usuli kortejlar to‘plamini qaytaradi. Har bir kortej elementning kaliti va qiymatini o‘z ichiga oladi, ularga murojaat qilishda ***key*** va ***val*** o‘zgaruvchilarini chaqirilsa yetarli.

Shuningdek, kalitlar ustida takrorlash va qiymatlarni takrorlash uchun alohida usul ham mavjud. Lug‘atdagikalitlarni olish uchun **keys()** metodidan foydalaniladi:

**for key in dict\_users.keys():**

**print(key)**

Faqat qiymatlarni takrorlash uchun lug‘atning *values()* metodidan foydalanladi*:*

**for val in dict\_users.values():**

**print(val)**

**Murakkab lug‘atlar**

Raqamlar va satrlar kabi eng oddiy ob’ektlardan tashqari, lug‘atlar murakkabroq ob’ektlarni ham saqlashi mumkin. Masalan bir xil ro‘yxatlar, kortejlar yoki boshqa lug‘atlarni o‘z ichiga element sifatida joylashtirishi mumkin:

dict\_users = {

    Tomson: {

        "id": "745",

        "e\_main": "tomson12@gmail.com"

    },

    Bobi: {

        "id": "444",

        "e\_mail": "bobi@gmail.com",

        "skype": "bob123"

    }

}

Bunda lug‘atning har bir elementining qiymati o‘z navbatida alohida lug‘atni ifodalashi mumkin.

Ichki lug‘at elementlariga kirish uchun mos ravishda ikkita kalitdan foydalanish kerak:

old\_e\_mail = dict\_users["Tomson"]["e\_mail"]

dict\_users["Tomson"]["e\_mail"] = "supertomson@gmail.com"

**print** (dict\_users["Tomson"])

# { "id": "745", "e\_main": "supertomson@gmail.com }

Ammo agar lug‘atda bo‘lmagan kalit orqali qiymat olishga harakat qilinsa, Python ***KeyError*** xatoligini chiqaradi:

tom\_skype = dict\_users["Tom"]["skype"]    # KeyError

Xatolikka yo‘l qo‘ymaslik uchun lug‘atda kalit mavjudligini tekshirish kerak:

key = "skype"

**if** key **in** dict\_users["Tomson"]:

**print**(dict\_users["Tomson"]["skype"])

**else**:

**print**("skype lug‘atdan topilmadi!")

Boshqa barcha jihatlarda murakkab va ichki lug‘atlar bilan ishlash oddiy lug‘atlar bilan ishlash kabi bajariladi.

**Nazorat savollari:**

1. Pythonda tug‘at tushunchasiga tarif bering.
2. Pythonda lug‘atlar qanday usullar bilan yaratilishi mumkin?
3. Ro‘yxatlar va kortejlarni lug‘atga aylantirish qanday amalga oshiriladi?
4. Lug‘at Elementlariga murojaat qilish va o‘zgartirish qanday amalga oshiriladi?
5. **get()** metodining vazifasi qanday va uning qabul qiluvchi parametrlari qanday?
6. **del** metodining vazifasi va ishlash prinsipi haqida gapirib bering.
7. **pop()** metodining vazifasi qanday va uning qabul qiluvchi parametrlari qanday?
8. **clear()** metodining vazifasi va ishlash prinsipi haqida gapirib bering.
9. **copy()** metodining vazifasi qanday va uning qabul qiluvchi parametrlari qanday?
10. **upgrate()** metodining vazifasi va ishlash prinsipi haqida gapirib bering
11. **items()** metodining vazifasi va ishlash prinsipi haqida gapirib bering

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**10-mashg‘ulot.** Pythonda Funksiya tushunchasi. Foydalanuvchi funksiyasi..

O‘quv savollari:

1. Funksiyalarni aniqlash va uni chaqirish.
2. Parametrli va parametrsiz funksiya.
3. Anonim funksiya. Dekoratorlar. Global va lokal o‘zgaruvchi.
4. Lambda funktsiyasi.

**1. Funksiyalarni aniqlash va uni chaqirish.**

Funktsiyalar ma’lum bir vazifani bajaradigan va dasturning boshqa qismlarida qayta ishlatilishi mumkin bo‘lgan kod blokini ifodalaydi. Funktsiyalardan oldingi mavzularni tushuntirish davomida allaqachon foydalaib o‘tildi. Masalan, konsolga qiymatlarni chop etadigan **print()** funksiyasidan deyarli har bir mavzuda foydalanildi. Python ko‘plab standart o‘rnatilgan funktsiyalarga ega bo‘lgan dasturlash tilidir. Bunday standart funksiyalardan tashqari foydalanuvchi o‘ziga kerakli funktsiyalarni yaratish imkoni mavjud. Funksiya yaratish uchun quyidagi sintaksisdan foydalaniladi:

**def** function\_name([parametrlar]):

[funksiya tanasi]

Pythonda funksiya boshqa dasturlash tillaridan farqli ravishda def kalit so‘zi bilan boshlanadi. Def kalit so‘zidan keyin funksiya nomi va agar qabul qiluvchi parametrlari mavjud bo‘lsa, ularni qavs ichida, vergul bilan ajratgan holda nomlari keltiriladi va qavslardan keyin osti-usti ikki nuqta qo‘yiladi. Va keyingi qatordan funksiya bajaradigan ko‘rsatmalar bloki boshlanadi. Barcha funktsiya tanasidagi operatorlar satr boshidan ichkariga chekingan holda bir ustundan boshlanishi kerak.

Masalan, eng oddiy funktsiyaning ko‘rinishi:

**def** say\_hello():

print ("Salom")

Funktsiya nomi: say\_hello deb ataladi. U hech qanday parametrlarga ega emas va konsolga "Salom" satrini chop etadigan bitta operatorni o‘z ichiga oladi.

E’tibor bering, funktsiya iboralari funktsiya boshidan chekinishi kerak. Masalan:

**def** say\_hello():

print("Salom")

print("Xayr")

Bu yerda print("Xayr") buyrug‘i say\_hello funksiyasi bilan bir ustunda va shuning uchun bu funksiyaga tegishli bo‘lmagan operator hisoblanadi. Funktsiya tarkibiga kiruvchi blok va funktsiyaga tegishli bo‘lmagan operatorlar bir-biri bilan oldidagi qoldirilgan tab (bo‘sh joy) bilan farq qiladi.

**Funksiyaga murojaat**. Funktsiyani chaqirish uchun funktsiya nomi va uning qabul qiluvchi parametrlariga mos qiymatlar qavs ichida ko‘rsatiladi:

**funktsiya\_nomi** ([parametrlar])

Masalan, salom tekstini konsolga chiqaruvchi bitta oddiy funksiya yaratamiz va uni chaqiramiz:

**def** hello():

print ("Salom")

hello()

hello()

hello()

Bu yerda hello funksiyasi ketma-ket uch marta chaqirilmoqda. Natijada, konsolda quyidagi natijalar paydo bo‘ladi:

Salom

Salom

Salom

Shuni esda tutish kerakki, funktsiya yuqorida yaratiladi va keyin chaqiriladi.

Agar funktsiya tarkibida faqat bitta operator mavjud bo‘lsa, bu operatorni funksiya ta’rifining davomidan ikki nuqtadan keyin joylashtirilishi mumkin:

**def** hello(): print("Salom")

hello ()

Boshqa funktsiyalarni ham xuddi shunday aniqlash va chaqirish mumkin. Masalan, bir nechta funksiyalarni aniqlaymiz va bajaramiz:

**def** say\_hello():

print ("Salom")

**def** say\_goodbye():

print ("Xayr")

say\_hello ()

say\_goodbye ()

Natijaning konsolda ko‘rinishi :

Salom

Xayr

**Lokal funksiya.** Ba’zi funktsiyalarni boshqa funktsiyalar ichida aniqlash mumkin. Bunday funksiyalarni ichki funktsiyalar ya’ni ***lokal funksiyalar*** deb ataladi. Lokal funksiyalar faqat o‘zi tegishli bo‘lgan funksiya doirasida ishlatilishi mumkin. Masalan :

**def** print\_msg():

**def** say\_hello(): print("Salom")

**def** say\_bye(): print("Xayr")

say\_hello()

say\_bye()

print\_msg()

Bu yerda say\_hello() va say\_bye() funksiyalari print\_msg() funksiyasi ichida aniqlanadi va shuning uchun u uchun lokal fuksiya hisoblanadi. Shunga ko‘ra, ular faqat print\_msg() funksiyasi ichidagina ishlatilishi mumkin.

**main funksiyasi** **va undan qo‘llanilishi**. Dasturda ko‘plab funktsiyalar ishtirok etishi mumkin. Va ularning barchasini birlashtirish uchun asosiy bitta funksiya ishlatiladi. Odatda bu funksiya main deb ataladi va unda boshqa funktsiyalar chaqiriladi:

**def** main\_f ():

say\_hello ()

say\_bye ()

**def** say\_hello():

print("Salom")

**def** say\_bye():

print("Xayr")

# Asosiy funksiyani chaqirish

main\_f()

**2. Parametrli va parametrsiz funksiya**

Funksiyalarni parametrli va parametrsiz funksiyalarga ajratishimiz mumkin. Parametrsiz funksiyalar tashqaridan hech qanday qiymat qabul qilmaydi. Parametrli funksiyalar esa funksiya chaqirilayotganda biror bir qiymatni berilishini kutadigan funksiyalardir. Ushbu parametrlar orqali funksiyaga turli ma’lumotlarni uzatish mumkin. Buni **print()** funksiyasimisolida ko‘rib chiqamiz. **print()** parametr sifatida konsolga chop etilishi kerak bo‘lgan xabarli qabul qiladi.

Endi bitta oddiy, ism paramrli funksiya yaratamiz va uni chaqiramiz:

**def** say\_hello(ism):

    print(f"Salom, { ism }")

say\_hello("Jamshid")

say\_hello("Boboy")

say\_hello("Asilbek")

Say\_hello funksiyasi ism parametriga ega va funksiya chaqirilganda ushbu parametrga ba’zi qiymatlarni berish mumkin. Funksiya ichida parametrni oddiy o‘zgaruvchi sifatida ishlatish mumkin. Masalan, ushbu parametrning qiymatini print funksiyasi bilan konsolga chop etish. Shunday qilib, ifodada:

say\_hello("Jamshid ")

"Jamshid" qatori ism parametriga uzatiladi. Natijada, dasturni bajarishda quyidagi natijani konsol chiqaradi:

Salom, Jamshid

Salom, Boboy

Salom, Asilbek

Funktsiya chaqirilganda, qiymatlar pozitsiya bo‘yicha parametrlarga o‘tkaziladi. Masalan, bir nechta parametrli funksiyani quyidagicha yaratish va chaqirish mumkin:

**def** print\_per (ism, yosh):

print(f"Ism: {ism}")

print(f"Yosh: {yosh}")

print\_per ("Jamshid", 25)

Bu yerda print\_per funksiyasi ikkita parametrni qabul qiladi: ism va yosh. Funktsiyani chaqirganda :

  print\_per("Jamshid", 25)

Funksiya chaqirilish vaqtidagi unga parametr sifatida beriladigan qiymatlar mos parametrlarga qabul qilinadi. Ya’ni birinchi qiymat – "Jamshid" birinchi parametrga, ya’ni ism parametriga qabul qilinsa, ikkinchi qiymat – 25 bo‘lsa, yosh parametriga qabul qilinadi. Va funktsiya ichida parametr qiymatlari konsolga chop etiladi:

Ism: Tom

Yosh: 37

**Standart qiymatlar**

Funktsiyani yaratish mobaynida, funksiyaning mavjud parametrlari uchun boshlang‘ich qiymatlarni belgilash orqali funksiyaning ba’zi parametrlariga qiymat berishni ixtiyoriy qilinishi mumkin. Masalan :

**def** say\_hello(ism="Jamshid"):

    print(f"Salom, {ism}")

say\_hello()   # bu yerda ism = "Jamshid"

say\_hello("Bobo") # bu yerda ism = "Bobo"

Bu yerda ism parametriga qiymat berish majburiy emas. Ya’ni, agar funktsiyani chaqirishda uning ism parametriga qiymat berilmasa standart qiymat – "Jamshid" qo‘llaniladi. Ushbu dasturning konsolda ko‘rinishi quyidagicha bo‘ladi:

Salom, Jamshid

Salom, Bobo

Agar funktsiya bir nechta parametrlarga ega bo‘lsa, qiymat berish ixtiyoriy bo‘lgan parametrlar, qiymat berish majburiy bo‘lgan qilinganlardan keyin kelishi kerak. Masalan :

**def** print\_per(ism, yosh = 54):

    print(f"Ism: {ism}   Yosh: {yosh}")

print\_per("Bobo")

print\_per("Jamshid", 25)

Bu yerda yosh parametri ixtiyoriy va standart boʻyicha 54 ga teng. Undan oldin qiymat berish majburiy bo‘lgan ism parametri joylashgan. Shuning uchun funktsiyani chaqirishda yosh parametriga qiymat o‘tkaza olmaymiz, lekin ism parametriga qiymat uzatish kerak.

 Agar kerak bo‘lsa, biz barcha parametrlarni ixtiyoriy qilishimiz mumkin:

def print\_per(ism = "Jamshid", yosh = 18):

    print(f"Ism: {ism}   Yosh: {yosh}")

print\_per()  # Ism: Jamshid Yosh: 18

print\_per("Bobo")        # Ism: Bobo   Yosh: 18

print\_per("Samat", 54)   # Ism: Samat   Yosh: 54

**Nomlangan parametrlar.** Yuqoridagi misollarda funktsiya chaqirilganda parametrlarga qiymatlar joylashish o‘rni bo‘yicha uzatildi. Ammo qiymatlarni parametrlarga nomi ko‘ra uzatish ham mumkin. Buning uchun funktsiyani chaqirishda parametr nomi ko‘rsatiladi va unga qiymat beriladi:

**def** print\_per(ism, yosh):

print(f"Ism: {ism}   Yosh: {yosh}")

print\_per(yosh = 22, ism = "Jamshid")

Bunday holda, yosh va ism parametrlari nomiga ko‘ra qiymatlarni topib olishadi. Funktsiya yaratilayotganda ism parametri birinchi o‘rinda turibdi. Shunga qaramay, funktsiyani chaqirishda print\_per(yosh= 22, ism = "Jamshid") kabi chaqirilsa ham bo‘ladi va shu tariqa yosh parametriga 22 ni va ism parametriga "Jamshid" qiymatini berish mumkin.

**Funksiya parametriga faqat nomi bo‘yicha qiymat berish.** \* belgisi qaysi parametrlarga nom berilishini belgilash imkonini beradi. Ya’ni qiymatlarni faqat nomi bo‘yicha uzatish mumkin bo‘lgan parametrlar. \* belgisining o‘ng tomonida joylashgan barcha parametrlar qiymatlarni faqat nomi bilan qabul qiladi:

**def** print\_per(ism, \*, yosh, tashkilot):

print(f"Ism: {ism} Yosh: {yosh} Tashkilot: {tashkilot}")

print\_per("Bobo", yosh = 41, tashkilot = "Microsoft")

Bunday holda, yosh va tashkilot parametrlari nomi bilan qiymatlar qabul qilishadi.

Parametrlar ro‘yxatiga \* bilan prefiks qo‘yish orqali barcha parametrlarni nomlash mumkin:

**def** print\_person(\*,   ism, yosh, tashkilot):

print(f"Ism: {ism} Yosh: {yosh} Tashkilot: {tashkilot}")

Aksincha, siz qiymatlarni faqat pozitsiya bo‘yicha, ya’ni pozitsion parametrlar bo‘yicha qabul qilish mumkin bo‘lgan parametrlarni belgilashingiz kerak bo‘lsa, unda / belgisidan foydalanish mumkin.

Bunda **/** belgisidan olda joylashgan barcha parametrlar faqat joylashish o‘rniga ko‘ra qiymatlar va qabul qilishi mumkin.

**def** print\_per(ism, /, yosh, tashkilot="Micros"):

print(f"Ism:{ism} Yosh: {yosh} Tashkilot: {tashkilot}")

print\_per("Jamshid", tashkilot="ShamsWeb", yosh = 11)

print\_per("Bobo", 41)

Bunday holda, ism parametri pozitsiondir.

Bitta funktsiyada bir vaqtning o‘zida pozitsion va nomli parametrlar mavjud bo‘lishi mumkin.

**def** print\_per(ism, /,  yosh = 18, \*, tashkilot):

print(f"Ism: {ism} Yosh:{yosh} Tashkilot: {tashkilot}")

print\_per("Samat", tashkilot ="Google")

print\_per("Tomson", 37, tashkilot ="JetBrains")

print\_per("Bobo", tashkilot ="Micros", yosh= 42)

Bunday holda, ism parametri / belgisining chap tomonida joylashgan, shuning uchun u pozitsion, majburiydir hamda faqat pozitsiya bo‘yicha qiymat berilishi mumkin.

Tashkilot parametri \* belgisining o‘ng tomonida joylashganligi sababli nomlangan parametr hisoblanadi. Yosh parametri nomi bo‘yicha ham, joylashgan o‘rni bo‘yicha ham qiymat qabul qilishi mumkin.

**Aniqlanmagan sondagi parametrlar.** Funksiyaga noma’lum miqdordagi qiymatlarni uzatish uchun yulduzcha belgisidan foydalaniladi. Bu funktsiyaga nomalum miqdordagi, bir nechta qiymatlarni qabul qilishda qo‘llaniladi. Masalan, sonlar yig‘indisini hisoblash funksiyasini ko‘rib chiqamiz:

**def** sum(\*raqamlar):

natija = 0

**for** j **in** raqamlar:

natija += j

print(f"sum = {natija}")

sum(1, 2, 3, 4, 5) # sum = 15

sum(3, 4, 5, 6) # sum = 18

Bunday holda, sum funktsiyasi bitta parametrni oladi - \* raqamlar, lekin parametr nomi oldidagi yulduzcha, aslida ushbu parametr o‘rniga cheksiz miqdordagi qiymatlarni yoki qiymatlar to‘plamini qabul qilish mumkinligini ko‘rsatadi. Funktsiyaning o‘zida for siklidan foydalanib, ushbu to‘plamdan birma-bir elementlarni olish mumkin, ushbu to‘plamdan har bir qiymatni j o‘zgaruvchiga olish va u bilan qandaydir amallarni bajarish mumkin. Misol uchun, bu dasturda uzatilgan raqamlarning yig‘indisi hisoblandi.

**3. Dekoratorlar.**

Dekoratorlarni tushunish har qanday Python dasturchisi uchun muhim bosqichdir. Ushbu maqola dekorativlar sizga qanday qilib samaraliroq va samarali Python dasturchisi bo‘lishingizga yordam berishi haqida bosqichma-bosqich qo‘llanmadir.

Python-dagi dekorativlar chaqiriladigan ob'ektning o‘zini doimiy ravishda o‘zgartirmasdan, *chaqiriladigan* ob'ektlarning (funktsiyalar, usullar va sinflar) xatti-harakatlarini kengaytirish va o‘zgartirish imkonini beradi .

Mavjud sinf yoki funktsiyaning xatti-harakatlariga "biriktirilishi" mumkin bo‘lgan har qanday etarlicha umumiy funksionallik dekorativlardan foydalanishning ajoyib namunasidir. Bunga quyidagilar kiradi:

* ro‘yxatga olish,
* kirish nazorati va autentifikatsiyani ta'minlash,
* vaqtni boshqarish vositalari va funktsiyalari,
* Tezlik chegarasi,
* keshlash va boshqalar.

**Nima uchun Pythonda dekorativlarni o‘rganishingiz kerak?**

Bu adolatli savol. Men hozir gapirgan narsa mavhum bo‘lib tuyuladi va dekorativlar Python dasturchisiga kundalik ishida qanday yordam berishini tushunish qiyin bo‘lishi mumkin. Mana bir misol:

Tasavvur qiling, sizning hisobot dasturingiz biznes mantig‘iga ega 30 ta funktsiyaga ega. Yomg‘irli dushanba kuni ertalab menejeringiz sizga aytadi:

“TPS hisobotlarini eslaysizmi? Hisobot ishlab chiqaruvchining har bir bosqichida kirish va chiqish ma'lumotlarini jurnalga qo‘shishimiz kerak. XYZ kompaniyasiga bu audit uchun kerak. Men ularga buni chorshanbagacha hal qilishimiz mumkinligini aytdim.

Python dekorativlari bilan qanchalik tanish ekanligingizga qarab, bu so‘rov sizning qon bosimingizni oshiradi yoki ushbu yangi talablarni asta-sekin qabul qiladi.

Dekorator haqida ma'lumotga ega bo‘lmasangiz, ehtimol siz keyingi uch kun davomida ushbu 30 ta funktsiyaning har birini o‘zgartirishga va ularni qo‘lda ro‘yxatdan o‘tish qo‘ng‘iroqlari bilan aralashtirib yuborishga harakat qilasiz. Umuman olganda, vaqtingizni qiziqarli o‘tkazing.

Agar siz dekoratorlarni bilsangiz, xotirjam jilmayib, "Yaxshi, bugun soat 14:00 da tayyor bo‘ladi" kabi bir narsa deysiz.

[Shundan so‘ng, siz umumiy @audit\_log](https://habr.com/users/audit_log) dekoratorining kodini chop etasiz (faqat 10 qator uzunlikda) va uni har bir funktsiya ta'rifidan oldin qo‘shasiz. Keyin siz va'da qilasiz va bir chashka qahva ichasiz.

Bu yerda men bo‘rttirib aytyapman, albatta. Ammo ozgina. Dekoratorlar haqiqatan ham kuchli bo‘lishi mumkin.

Men dekorativlarni tushunish har qanday tajribali Python dasturchisi uchun muhim bosqich deb aytardim. Ular bir nechta ilg‘or til tushunchalarini, shu jumladan birinchi darajali funktsiyalarning xususiyatlarini yaxshi tushunishni talab qiladi.

Lekin:

**Dekoratorlarni tushunish bunga arziydi**

Python-da dekorativlarning qanday ishlashini tushunish juda foydali.

Albatta, dekorativlarni birinchi marta taqdim etilganda tushunish juda qiyin bo‘lib tuyuladi, lekin bu juda foydali xususiyat bo‘lib, uni uchinchi tomon ramkalari va Python standart kutubxonasida tez-tez ko‘rishingiz mumkin.

Dekoratorlarni tushuntirish har qanday yaxshi Python o‘quv qo‘llanmasining muhim bo‘limidir. Ushbu maqolada men sizni ular bilan bosqichma-bosqich tanishtirishga harakat qilaman.

Mavzuga kirishdan oldin, keling, Python-da birinchi darajali funktsiyalarning xususiyatlarini ko‘rib chiqaylik. Men [dbader.org](https://dbader.org/blog/python-first-class-functions) saytida ularga qo‘llanma yozdim , uni o‘qish uchun bir necha daqiqa vaqt ajratishingizni tavsiya qilaman. Bu erda dekorativlarni tushunish uchun "birinchi darajali funktsiyalar" dan eng muhim xulosalar:

* Funktsiyalar ob'ektlardir - ular o‘zgaruvchilarga tayinlanishi, boshqa funktsiyalarga o‘tkazilishi va qaytarilishi mumkin.
* Funktsiyalar boshqa funktsiyalar ichida aniqlanishi mumkin va bola funktsiyasi ota-ona funksiyasining mahalliy holatini (leksik yopilishlar) olishi mumkin.

Xo‘sh, ketishga tayyormisiz? Keling, boshlaymiz.

Python dekorator asoslari

Xo‘sh, aslida dekorativlar nima? Ular boshqa funktsiyani "bezatadi" yoki "o‘radi" va kodni o‘ralgan funksiya bajarilishidan oldin va keyin bajarishga imkon beradi.

Dekoratorlar boshqa funktsiyalarning harakatlarini o‘zgartirishi yoki kengaytirishi mumkin bo‘lgan qayta ishlatiladigan modullarni aniqlash imkonini beradi. Bundan tashqari, ular sizga o‘ralgan funktsiyani doimiy ravishda o‘zgartirmasdan buni amalga oshirishga imkon beradi. Funktsiyaning harakati faqat bezatilganida *o‘zgaradi* .

Xo‘sh, oddiy dekoratorni amalga oshirish nimaga o‘xshaydi? Umuman olganda, dekorator chaqiriladigan ob'ekt bo‘lib, u qo‘ng‘iroq qilinadigan ob'ektni kiritadi va boshqa chaqiriladigan ob'ektni qaytaradi.

Quyidagi funktsiya bu xususiyatga ega va siz yozishingiz mumkin bo‘lgan eng oddiy dekorativ deb hisoblanishi mumkin:

**def** null\_decorator(func):

**return** func

Ko‘rib turganingizdek, null\_decoratorchaqiriladigan ob'ekt bo‘lib, u boshqa chaqiriladigan ob'ektni kiritish sifatida qabul qiladi va uni o‘zgartirmasdan bir xil kirish ob'ektini qaytaradi.

Keling, uni boshqa funktsiyani bezash (yoki o‘rash) uchun ishlatamiz:

**def** greet():

**return** 'Hello!'

greet = null\_decorator(greet)

>>> greet()

'Hello!'

Ushbu misolda greetmen funktsiyani belgilab qo‘ydim va keyin uni funksiya orqali ishga tushirish orqali darhol bezadim null\_decorator. Bilaman, bu hozircha unchalik foydali ko‘rinmaydi (biz maxsus null dekoratorni foydasiz qilib ishlab chiqdik, to‘g‘rimi?), lekin bir muncha vaqt o‘tgach, bu Pythonda dekorator sintaksisi qanday ishlashini aniqroq qiladi.

O‘zgaruvchini aniq chaqirish va keyin uni qayta tayinlash null\_decoratoro‘rniga , funktsiyani bir bosqichda bezash uchun Python sintaksisidan foydalanishingiz mumkin :greetgreet@

@null\_decorator

**def** greet():

**return** 'Hello!'

>>> greet()

'Hello!'

[Funksiya taʼrifidan oldin @null\_decorator](https://habr.com/users/null_decorator) satrlarini qoʻyish avval funksiyani belgilash va keyin unga dekoratorni qoʻllash bilan bir xil. Sintaksisdan foydalanish @oddiygina sintaktik shakar va bu tez-tez ishlatiladigan naqsh uchun stenografiyadir.

E'tibor bering, sintaksisdan foydalanish @funktsiyani to‘g‘ridan-to‘g‘ri belgilash vaqtida bezatadi. Bu mo‘rt xakerlarsiz bezaksiz asl nusxaga kirishni qiyinlashtiradi. Shuning uchun, bezaksiz funksiyani chaqirish imkoniyatini saqlab qolish uchun siz ba'zi funktsiyalarni qo‘lda bezashingiz mumkin.

Hozirgacha juda yaxshi. Keling, bu qanday amalga oshirilganini ko‘rib chiqaylik.

**Dekoratorlar xatti-harakatni o‘zgartirishi mumkin**

Endi siz dekorator sintaksisi bilan bir oz tanish bo‘lganingizdan so‘ng, keling, *aslida* biror narsani bajaradigan va bezatilgan funksiyaning harakatini o‘zgartiradigan boshqa dekorator yozamiz.

Mana, bezatilgan funktsiya natijasini katta harflarga o‘zgartiradigan biroz murakkabroq dekorator:

**def** uppercase(func):

**def** wrapper():

original\_result = func()

modified\_result = original\_result.upper()

**return** modified\_result

**return** wrapper

Kirish funksiyasini null dekorator qilganidek oddiygina qaytarish o‘rniga, dekorator uppercaseyangi funksiyani (yopish) aniqlaydi va undan kirish funksiyasini chaqirish vaqtida uning harakatini o‘zgartirish uchun o‘rash uchun ishlatadi.

Yopish wrapperbezaksiz kirish funksiyasiga kirish huquqiga ega va kirish funksiyasi chaqirilishidan oldin va keyin qo‘shimcha kodni bajarishi mumkin. (Texnik jihatdan kiritish funksiyasini umuman chaqirish shart emas).

E'tibor bering, bezatilgan funktsiya ilgari hech qachon bajarilmagan. Aslida, bu nuqtada kirish funktsiyasini chaqirish hech qanday ma'noga ega emas - dekorator chaqirilganda o‘zining kiritish funktsiyasining harakatini o‘zgartirishi kerak.

uppercaseDekoratorning harakatini ko‘rish vaqti keldi . Agar asl funktsiyani u bilan bezasangiz nima bo‘ladi greet?

@uppercase

**def** greet():

**return** 'Hello!'

>>> greet()

'HELLO!'

Umid qilamanki, bu siz kutgan natija bo‘ldi. Keling, bu erda nima sodir bo‘lganini batafsil ko‘rib chiqaylik. dan farqli o‘laroq null\_decorator, dekorator funksiyani bezashda *boshqa funksiya ob'ektini*uppercase qaytaradi :

>>> greet

<function greet at 0x10e9f0950>

>>> null\_decorator(greet)

<function greet at 0x10e9f0950>

>>> uppercase(greet)

<function uppercase.<locals>.wrapper at 0x10da02f28>

Avval ko‘rganingizdek, bu bezatilgan funksiya chaqirilganda uning harakatini o‘zgartirish uchun kerak. Katta harf dekoratorining o‘zi funksiyadir. Va u bezatgan kirish funktsiyasining "kelajakdagi xatti-harakatiga" ta'sir qilishning yagona yo‘li - kirish funktsiyasini yopish bilan almashtirish (yoki o‘rash).

Shuning uchun uppercaseu keyinroq chaqirilishi mumkin bo‘lgan boshqa funktsiyani (yopish) belgilaydi va qaytaradi, dastlabki kiritish funktsiyasini ishga tushiradi va uning chiqishini o‘zgartiradi.

Dekoratorlar qo‘ng‘iroq qilinadigan ob'ektning harakatini o‘ram yordamida o‘zgartiradilar, shuning uchun siz doimo asl nusxani o‘zgartirishingiz shart emas. Chaqiriladigan ob'ekt doimiy o‘zgarishlarga duch kelmaydi - uning xatti-harakati faqat bezatilganida o‘zgaradi.

Bu sizga mavjud funktsiyalar va sinflarni jurnallar va boshqa vositalar kabi qayta foydalanish mumkin bo‘lgan modullar bilan "biriktirish" imkonini beradi. Bu dekorativlarni Python-da ko‘pincha standart kutubxona va uchinchi tomon paketlarida ishlatiladigan kuchli xususiyatga aylantiradi.

Qisqa tanaffus

Aytgancha, agar siz hozirgi vaqtda qisqa kofe tanaffusga muhtoj ekanligingizni his qilsangiz, bu mutlaqo normaldir. Menimcha, yopilishlar va dekorativlar Pythonda tushunish eng qiyin tushunchalardan biridir. Shoshilmang va birdaniga hamma narsani tushunishga urinmang. Tarjimon sessiyasida birin-ketin misollar keltirish ko‘pincha mazmunni tushunishga yordam beradi.

Bilaman, hammasi siz uchun yaxshi bo‘ladi :)

Bitta funktsiyaga bir nechta dekorativlarni qo‘llash

Funktsiyaga bir nechta dekorator qo‘llanilishi ajablanarli emas. Bu ularning effektlarini to‘playdi va dekorativlarni qayta foydalanish mumkin bo‘lgan modullar kabi foydali qiladi.

Mana bir misol. Keyingi ikkita dekorator bezatilgan funksiyaning chiqish satrini HTML teglari bilan o‘rab oladi. Teglar qanday joylashtirilganiga qarab, Python bir nechta dekorativlarni qo‘llash tartibini ko‘rishingiz mumkin:

**def** strong(func):

**def** wrapper():

**return** '<strong>' + func() + '</strong>'

**return** wrapper

**def** emphasis(func):

**def** wrapper():

**return** '<em>' + func() + '</em>'

**return** wrapper

Keling, ushbu ikkita dekoratorni olib, ularni greetbir vaqtning o‘zida bizning funktsiyamizga qo‘llaymiz. Buni amalga oshirish uchun siz odatiy sintaksisdan foydalanishingiz @va bitta funktsiya ustiga bir nechta dekorativlarni joylashtirishingiz mumkin:

@strong

@emphasis

**def** greet():

**return** 'Hello!'

Bezatilgan funktsiyani ishga tushirsangiz, qanday natija ko‘rishni kutasiz? [@emphasis](https://habr.com/users/emphasis) dekoratori avval o‘z tegini qo‘shadimi yoki <em>@strong [ustunlik](https://habr.com/users/strong)   qiladimi? Bezatilgan funktsiyani chaqirganingizda nima sodir bo‘ladi:

>>> greet()

'<strong><em>Hello!</em></strong>'

Bu erda siz dekorativlar qanday tartibda ishlatilganligini aniq ko‘rishingiz mumkin: *pastdan yuqoriga* . Dastlab kiritish funksiyasi [@emphasis](https://habr.com/users/emphasis) decorator bilan o‘ralgan , so‘ngra hosil bo‘lgan (bezatilgan) funksiya yana [@strong](https://habr.com/users/strong) decorator bilan o‘ralgan .

Ushbu pastdan yuqoriga tartibni eslash uchun men bu xatti-harakatni "dekorator to‘plami" deb atashni yaxshi ko‘raman. Siz stackni pastki qismdan qurishni boshlaysiz va keyin yuqoriga ko‘tarilish uchun yangi bloklarni qo‘shishda davom etasiz.

Agar biz yuqoridagi misolni buzsak va dekorativlarni qo‘llash uchun sintaksisdan foydalanmasak @, dekorativ funktsiyalarga qo‘ng‘iroqlar zanjiri quyidagicha ko‘rinadi:

decorated\_greet = strong(emphasis(greet))

Bu erda yana dekoratorning birinchi qo‘llanilishini ko‘rishingiz mumkin emphasis, keyin esa natijada o‘ralgan funksiya yana dekoratorga o‘ralgan strong.

Bu shuningdek, dekorativ qoplamaning chuqur darajalari oxir-oqibat ishlashga ta'sir qiladi, chunki ular ichki funksiya chaqiruvlarini qo‘shishda davom etadilar. Amalda bu odatda muammo emas, lekin agar siz unumdorlikni talab qiluvchi kod ustida ishlayotgan bo‘lsangiz, buni yodda tutish kerak.

Argumentlarni qabul qiladigan dekoratsiya funktsiyalari

greetHozirgacha berilgan barcha misollar hech qanday dalil talab qilmaydigan oddiy nullary funktsiyani bezatadi . Shunday qilib, siz bu erda ko‘rgan dekoratorlar kiritish funktsiyasiga argumentlarni uzatish bilan shug‘ullanmagan.

Agar siz ushbu dekoratorlardan birini argumentlarni qabul qiluvchi funktsiyaga qo‘llamoqchi bo‘lsangiz, u to‘g‘ri ishlamaydi. O‘zboshimchalik bilan argumentlar oladigan funktsiyani qanday bezash kerak?

Bu erda Python funksiyasi o‘zgaruvchan sonli argumentlar bilan ishlashda \*args*yordam* beradi . \*\*kwargs Dekorator proxyushbu xususiyatdan foydalanadi:

**def** proxy(func):

**def** wrapper(\*args, \*\*kwargs):

**return** func(\*args, \*\*kwargs)

**return** wrapper

Ushbu dekorativning ikkita diqqatga sazovor tomoni bor:

* U barcha pozitsion va kalit argumentlarni yig‘ish va ularni o‘zgaruvchilarda ( va ) saqlash uchun yopish ta'rifidagi \*va operatorlaridan foydalanadi .\*\*wrapperargskwargs
* \*Keyin oʻramni yopish “argumentni ochish” va yordamida toʻplangan argumentlarni asl kiritish funksiyasiga oʻtkazadi \*\*.

(Yulduzcha va qoʻsh yulduzcha operatorlarining maʼnosi haddan tashqari yuklanganligi va ular qoʻllanilgan kontekstga qarab oʻzgarib turishi biroz uyatli. Lekin bu fikrni tushunasiz degan umiddaman).

Keling, dekorativning orqasidagi texnikani proxyfoydaliroq amaliy misol bilan kengaytiraylik. Mana, tracefunksiyaning argumentlarini va ishlayotgan natijalarini chop etadigan dekorator:

**def** trace(func):

**def** wrapper(\*args, \*\*kwargs):

print(f'TRACE: calling {func.\_\_name\_\_}() '

f'with {args}, {kwargs}')

original\_result = func(\*args, \*\*kwargs)

print(f'TRACE: {func.\_\_name\_\_}() '

f'returned {original\_result!r}')

**return** original\_result

**return** wrapper

Funksiyani bezash traceva uni chaqirish orqali siz bezatilgan funksiyaga uzatilgan argumentlarni va uning qaytish qiymatini chop etishingiz mumkin. Bu hali ham o‘yinchoq misolidir, ammo bu nosozliklarni tuzatish uchun ajoyib yordam bo‘ladi:

@trace

**def** say(name, line):

**return** f'{name}: {line}'

>>> say('Jane', 'Hello, World')

'TRACE: calling say() with ("Jane", "Hello, World"), {}'

'TRACE: say() returned "Jane: Hello, World"'

'Jane: Hello, World'

Nosozliklarni tuzatish haqida gapiradigan bo‘lsak, dekorativlarni tuzatishda bir nechta narsalarni yodda tutish kerak.

"Debuggable" dekorativlarni qanday yozish kerak

Dekoratordan foydalansangiz, aslida bir funktsiyani boshqasiga almashtirasiz. Bu jarayonning kamchiliklaridan biri shundaki, u asl (bezaksiz) funksiyaga biriktirilgan metama’lumotlarning bir qismini “yashiradi”.

Masalan, asl funktsiya nomi, uning docstring va parametrlar ro‘yxati yopish orqali yashiriladi:

**def** greet():

"""Return a friendly greeting."""

**return** 'Hello!'

decorated\_greet = uppercase(greet)

Agar siz ushbu funksiyaning istalgan metamaʼlumotlariga kirishga harakat qilsangiz, uning oʻrniga oʻramning yopish metamaʼlumotlarini koʻrasiz:

>>> greet.\_\_name\_\_

'greet'

>>> greet.\_\_doc\_\_

'Return a friendly greeting.'

>>> decorated\_greet.\_\_name\_\_

'wrapper'

>>> decorated\_greet.\_\_doc\_\_

**None**

Bu nosozliklarni tuzatish va Python tarjimoni bilan ishlashni noqulay va qiyinlashtiradi. Yaxshiyamki, buning tezkor yechimi bor: Python standart kutubxonasiga kiritilgan [functools.wraps dekoratori.](https://docs.python.org/3/library/functools.html" \l "functools.wraps)

functools.wrapsSiz o‘zingizning dekoratorlaringizda dekoratsiya qilinmagan funksiyadan bezatuvchining yopilishiga yetim metama'lumotlarni nusxalash uchun foydalanishingiz mumkin . Mana bir misol:

**import** functools

**def** uppercase(func):

@functools.wraps(func)

**def** wrapper():

**return** func().upper()

**return** wrapper

Dekorator tomonidan qaytarilgan oʻramni yopishni qoʻllash functools.wrapshujjat qatori va kiritish funksiyasining boshqa metamaʼlumotlarini oʻrab oladi:

@uppercase

**def** greet():

"""Return a friendly greeting."""

**return** 'Hello!'

>>> greet.\_\_name\_\_

'greet'

>>> greet.\_\_doc\_\_

'Return a friendly greeting.'

functools.wrapsO‘zingiz yozgan barcha dekorativlardan foydalanishni tavsiya qilaman . Bu ko‘p vaqtni talab qilmaydi va kelajakda disk raskadrovka paytida sizni (va boshqalarni) ko‘p bosh og‘rig‘idan xalos qiladi.

**4. Lambda funktsiyasi**

Python dasturlash tilidagi lambda ifodalari - bu lambda operatori yordamida aniqlangan kichik anonim funksiyalar. Lambda ifodasining sintaksisi quyidagicha:

**lambda** [parametrlar]: operatorlar

Eng oddiy lambda ifodasini aniqlaymiz:

msg = **lambda**: print ("salom")

msg()    # salom

Bu yerda lambda ifodasi msg o‘zgaruvchisiga yuklatilgan. Ushbu misolda lambda ifodasi hech qanday parametrga ega emas, hech narsa qaytarmaydi va shunchaki konsolga "salom" xabarini chop etadi. Va msg o‘zgaruvchisi orqali bu lambda ifodasini oddiy funktsiya kabi chaqirishimiz mumkin. Aslida, u quyidagi funktsiyaga o‘xshaydi:

**def** msg():

    print("salom")

Agar lambda ifodasi parametrlarga ega bo‘lsa, ular lambda kalit so‘zidan keyin aniqlanadi. Agar lambda ifodasi natijani qaytarsa, u holda u ikki nuqtadan keyin ko‘rsatiladi. Masalan, sonning kvadratini qaytaruvchi lambda ifodasini yozaylik:

kvadrat = **lambda** a : a \* a

print(kvadrat(4))     # 16

print(kvadrat(5))     # 25

Bunday holda, lambda ifodasi bitta a nomli parametrni qabul qiladi va undan keyingi Ikki nuqtaning o‘ng tomonida qaytariladigan qiymat - a \* a. Ushbu lambda ifodasi quyidagi funktsiyaning alternativi hisoblanadi:

**def** kvadrat2(a): **return** a \* n

Xuddi shu usulda, bir nechta parametrlarni qabul qiluvchi lambda ifodalarini yaratish mumkin:

sum = **lambda** a, b: a + b

print(sum(4, 5))     # 9

print(sum(5, 6))     # 11

Lambda ifodalari funksiya ta’riflarini biroz qisqartirishga imkon bersada, ular faqat bitta operatsiyani bajarishi mumkinligi bilan cheklangan. Biroq, parametr sifatida o‘tish yoki boshqa funktsiyaga qaytish uchun funktsiyadan foydalanish kerak bo‘lgan hollarda ular juda qulay bo‘lishi mumkin. Masalan, lambda ifodasini parametr sifatida o‘tkazish:

**def** do\_operatsiya(a, b, operatsiya):

    result = operatsiya(a, b)

    print(f"result = {result}")

do\_operatsiya(5, 4, **lambda** a, b: a + b)  #result = 9

do\_operatsiya(5, 4, **lambda** a, b: a \* b)  #result = 20

Bunday holda, oxirgi mavzuda bo‘lgani kabi, ularni parametr sifatida jo‘natish uchun funktsiyalarni belgilashimiz shart emas.

Xuddi shu narsa funktsiyalardan lambda ifodalarini qaytarish uchun ham amal qiladi:

**def** tanlash\_funk(tanlangan):

**if** tanlangan == 1:

**return** **lambda** a, b: a + b

**elif** tanlangan == 2:

**return** **lambda** a, b: a - b

**else**:

**return** **lambda** a, b: a \* b

operation = tanlash\_funk (1) # operation = a+b

print(operation(10, 6))  # 16

operation = tanlash\_funk (2) # operation = a-b

print(operation(10, 6))  # 4

operation = tanlash\_funk (3) # operation = a\*b

print(operation(10, 6))  # 60

**Nazorat savollari:**

1. Lambda funksiyasining vazifasi nima?
2. Lambda funksiyasining sintaksisi va ishlash prinsipi qanday?
3. Oddiy funksiyadan qanday afzallik va kamchiliklarga ega?
4. Lokal funksiya deb qanday funksiyalarga aytiladi?
5. Funksiyalar dasturlashda qanday ro‘l o‘ynaydi?
6. Foydalanuvchi funksiyalari qanday yaratiladi?
7. Funksiya tanasi va funksiyaga tegishli bo‘lmagan operatorlar qanday farqlanadi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**11-mashg‘ulot**. Fayllar va kataloglar bilan ishlash.

O‘quv savollari:

1. Faylni ochish. Fayllar bilan ishlovchi metodlar.
2. os modulining imkoniyatlari.
3. Fayl va katalog yo‘lini o‘zgartirish.
4. Katalog va fayl bilan ishlovchi funksiya va metodlar.

**1. Faylni ochish. Fayllar bilan ishlovchi metodlar.**

Python dasturlash tilidagi fayllar bilan o‘zaro aloqada bo‘lish foydalanuvchiga ilova tomonidan qayta ishlangan ma’lumotlarni saqlash imkoniyatini beradi va foydalanuvchi keyinchalik istalgan qulay vaqtda faylga murojaat qilishi mumkin bo‘ladi. Ushbu python dasturlash tilining asosiy funktsiyalari fayllarni yaratish, faylga ma’lumotlarni yozish va fayldan ma’lumotlarni o‘qishni ancha osonlashtiradi.

**Faylni yaratish va ochish**

Fayl bilan ishlash uchun albatta avval uni yaratish kerak. Buni standart operatsion tizim vositalari yordamida kerakli katalogga o‘tish va **\*.txt** formati bilan yangi hujjat yaratish orqali amalga oshirish ham mumkin. Biroq, shunga o‘xshash harakat Python dasturlash tilida **open()** metodi yordamida amalga oshiriladi. Bu metodga **fayl nomi** va uni **qayta ishlash rejimi** parametrlar sifatida berilishi kerak.

Quyidagi kod fayl o‘zgaruvchisini yangi hujjatga ishora qilishini ko‘rsatadi. Agar ushbu dasturni ishga tushirilsa, u manba kodi saqlanadigan papkada **test.txt** matn faylini yaratadi.

file\_txt = **open**("test.txt", "**w**")

file\_txt.**close**()

Agar **test.txt** nomli fayl kod katalogida allaqachon mavjud bo‘lsa, dastur yangi hujjat yaratmasdan u bilan ishlashni davom ettiradi. Ko‘rib turganingizdek, fayl nomi **open()** metodining birinchi parametridir. Undan so‘ng ma’lumotlarni **qayta ishlash usulini** ko‘rsatadigan maxsus belgi parametr sifatida keladi. Bu yerda ikkinchi parametr sifatida kelgan “w” - fayl faqat yozish uchun ochilayotganini bildiradi.

Endi ma’lumot yo‘qolmasligini ta’minlash uchun fayl ustida turli amallarni bajarilgandan so‘ng, uni **close()** funktsiyasidan foydalanib yopish kerak bo‘ladi.

Oldingi misolda faylga kirish uchun nisbiy yo‘l ishlatilgan bo‘lib, u qattiq diskdagi ob’yektning joylashuvi haqida to‘liq ma’lumotni o‘z ichiga olmaydi. Ularni o‘rnatish uchun **open()** funksiyaga birinchi argument sifatida mutlaq yo‘lni ko‘rsatilishi kerak. Bunday holda, test.txt hujjati dastur papkasida emas, balki D diskidagi qaysidir bir katalokga joylashgan bo‘ladi.

file\_txt = **open**(**r**"D:\test\test.txt", "**w**")

file\_txt.**close**()

Bu yerda faylga yo‘lni ko‘rsatishda satrdan oldin **r** belgisi qo‘yiladi. Aks holda, kompilyator "\t" ketma-ketligini maxsus operator sifatida ko‘rib chiqadi va xatolik kelib chiqadi.

**Faylni ochilish rejimlari**

Barcha dasturlash tillaridagi kabi Python dasturlash tilida ham faylni ochishda ishlatiladigan maxsus belgilardan foydalanladi. Ular faylni ochish rejimi deb ataladi va faylni qanday maqsadda ochishni aniq beradi. Ularning barchasi quyidagi jadvalda keltirilgan, unda ularning belgisi va qisqacha mazmuni keltirilgan:

5-jadval

| **Belgi** | **Mazmuni** |
| --- | --- |
| «r» | O‘qish uchun ochiq (standart) |
| «w» | Yozish uchun ochiladi. Agar fayl ko‘rsatilgan kotalogda mavjud bo‘lmasa, yangisi yaratiladi. Yozilgan ma’lumot hujjatdagi barcha ma’lumotlarni o‘rniga yoziladi |
| «a» | Yozish uchun ochiladi. Yozilgan ma’lumot hujjatning oxiriga qo‘shiladi |
| «b» | Faylni ikkilik rejimda ochish |
| «t» | Faylni matn rejimida ochish (standart) |
| «+» | O‘qish va yozish uchun bir vaqtning o‘zida ochish. Ushbu amal alohida o‘zi ochish rejimi sifatida ishlatilmaydi. a+, r+, w+ kabi ishlatiladi. |

**Open()** metodining ikkinchi argumentidan foydalanib, turli xil fayl rejimlarini birlashtirish mumkin. Masalan, yozma ma’lumotlarni ikkilik rejimda o‘qish uchun "rb" ni belgilash kerak.

Yana bir misol: "r+" va "w+" o‘rtasidagi farq shundaki, “w+” - agar fayl mavjud bo‘lmasa, yangi fayl yaratiladi. Birinchi holda, xatolik kelib chiqadi. "r+" va "w+" dan foydalanish faylni o‘qish va yozish uchun ochadi.

**Usullari**

**open()** funksiyasi tomonidan qaytarilgan ob’yekt mavjud faylga tavsifni o‘z ichiga olgan to‘rtta parametrni qaytaradi. Quyidagi jadvalda ularning barchasi nomlari va ma’nolari keltirilgan:

6-jadval

| **Nomi** | **Ma’nosi** |
| --- | --- |
| name | fayl nomini qaytaradi |
| mode | ochilgan rejimni qaytaradi |
| closed | fayl yopiq bo‘lsa true ni, ochiq bo‘lsa false qiymatini qaytaradi |
| softspace | agar faylning chiqishi alohida bo‘sh joy belgisini o‘z ichiga olmasa, true qiymatini qaytaradi |

Faylning xususiyatlarini ko‘rsatish uchun kirish operatoridan, ya’ni nuqtadan foydalanish va keyin buni **print()** funktsiyasiga parametr sifatida berish kifoya bo‘ladi.

**Masalan** :

f = **open**(r"D:\test\test.txt", "**w**")

print(f.**name**)

f.**close**()

**Natijasi:**

D:\test\test.txt

**Faylga ma’lumot yozish**

Faylga ma’lumot yozish **write()** usuli yordamida amalga oshiriladi. Usul mavjud faylga ishora qiluvchi ob’yektda chaqiriladi. Shuni esda tutish kerakki, buni amalga oshirish uchun avval hujjatni ochish funktsiyasidan foydalanib ochish va "w" belgisi bilan yozish rejimini belgilash kerak bo‘ladi. **Write()** usuli argument sifatida matn fayliga yoziladigan ma’lumotlarni qabul qiladi. Quyidagi kod misolida "Salom dunyo!" satrini kiritilishi ko‘rsatilgan:

file\_txt = **open**("test.txt", "**w**")

file\_txt.**write**("Salom dunyo!")

file\_txt.**close**()

Agar ilgari yozilgan ma’lumotlarga yangi ma’lumotlarni qo‘shish kerak bo‘lsa, u holda ochish rejimi sifatida "a" belgisini ko‘rsatib, **open** funksiyasini qayta chaqirish kerak. Aks holda, **test.txt** faylidagi barcha ma’lumotlar butunlay o‘chirib tashlanadi. Quyidagi kod misolida faylga ma’lumot qo‘shish uchun matnli hujjatni ochadi. Shundan so‘ng unda " Salom dunyo!" satri hujjat oxiriga joylashtiriladi. Shunday qilib, **test.txt** faylida "Salom dunyo! Salom dunyo!" satri paydo bo‘ladi. Bularning barchasidan so‘ng, faylni majburiy yopish haqida unutmang.

file\_txt = **open**("test.txt", "a")

file\_txt.**write**(" Salom dunyo!")

file\_txt.**close**()

Matn fayliga ma’lumotlarni yozishning eng oddiy protsedurasi shunday amalga oshiriladi. Shuni ta’kidlash kerakki, Python dasturlash tilida hujjatlar bilan yanada ilg‘or ishlash uchun ko‘plab qo‘shimcha vositalar mavjud bo‘lib, ular yaxshilangan yozishni ham o‘z ichiga oladi.

**Ikkilik ma’lumotlarni yozing**

Ikkilik ma’lumotlarni yozishda "wb" rejimidan foydalanish kerak. Utf8 kodlashda satr yozishga misol:

file\_txt = **open**(‘test.dat’, ‘**wb**’)

file\_txt.**write**(bytes(‘наш строк’, **‘utf8’**))

file\_txt.**close**()

**Fayldan o‘qish**

Fayldan ma’lumotni o‘qish uchun mavjud faylga ishora qiluvchi ob’yektda **read()** metodini chaqirish kerak bo‘ladi. Matn faylini ochishda **open()** funksiyaning ikkinchi parametri sifatida "r" belgidini ko‘rsatishni ham unutmaslik kerak.

Quyidagi misolda o‘qish **test.txt** dan ma’lumotni **print()** funksiyasiga qaytaradi, so‘ngra ma’lumotni ekranga chop etadi. Avvalgidek, dastur hujjatni **close()** usuli bilan yopish bilan yakunlanadi. **Read()** metodi butun son parametrini ham qabul qilishi mumkin, bu o‘qish uchun belgilar sonini uzatish uchun ishlatiladi. Misol uchun, agar 5 ni kiritilsa, dastur faqat *“Salom”* satrini o‘qiydi.

**try**:

file\_txt = **open**("test.txt", "**r**")

print(file\_txt.**read**())

file\_txt.**close**()

**except** **FileNotFoundError**:

print(‘Fayl topilmadi!’)

**except** **IOError**:

print(‘Qandaydir xatolik!’)

Iltimos, ochishda agar ko‘rsatilgan fayl topilmasa xatolik yuz berishi mumkinligini unutmang. Shunday qilib, try bloki bilan orab olish kerak edi. Pythonda buni oldini olish uchun **with** konstruktoridan foydalanish mumkin. Bu holda xatoliklarni ko‘rib chiqish va hatto faylni yopish ham shart emas.

**Ikkilik ma’lumotlarni o‘qish**

Agar ma’lumotlar ikkilik turda bo‘lsa, **open()** funksiyasining ikkinchi parametri sifatida "rb" dan foydalanish kerak:

**try**:

f\_txt = **open**("test.dat", "**rb**")

b = f\_txt.**read**(1)

str = ""

**while** True:

b = f\_txt.**read**(1)

**if** b == b’’:

**break**

str += b.**hex**()

print(str)

f\_txt.**close**()

**except** **IOError**:

print(‘error’)

**Natijasi:**

81d182d180d0bed0bad0b081d182d180d0bed0bad0b0

Bu yerda fayl bayt bo‘yicha o‘qiladi. Har bir bayt o‘n oltilik sanoq sistemasi ko‘rinishdagi satr shaklida natija beradi. **Print**() funksiyasi olingan qatorni chop etadi.

**with as konstrultori**

Matnli fayllarni qayta ishlashni biroz avtomatlashtirish uchun bir qator **with as** iboralaridan foydalanish tavsiya etiladi. Ushbu operatrolardan foydalanilsa, yopilishi kerak bo‘lgan hujjatda **close()** usulini chaqirishning hojati yo‘q, chunki bu avtomatik ravishda sodir bo‘ladi. Buni quyidagi qism dasturda ko‘rib chiqamiz. Odatdagidek, ekranga satr ma’lumotlarini chop etish uchun **print()** usuli qo‘llaniladi:

**with** open(‘test.txt’, ‘r’) **as** file\_txt:

print(file\_txt.**read**())

Bundan tashqari, bu holda istisno bilan shug‘ullanish kerak emas. Agar ko‘rsatilgan nomga ega fayl bo‘lmasa, **with** operatoridagi ichki kodli satrlar bajarilmaydi.

Python dasturlash tilining ko‘rib chiqilgan xususiyatlari yordamida foydalanuvchi ma’lumotlarni fayllarga o‘qish va yozishning asosiy operatsiyalarini osongina bajarishi mumkin.

**2. os modulining imkoniyatlari**

Python dasturlash tilining standart kutubxonasidagi OS moduli odatda o‘rnatilgan OT va shaxsiy kompyuter fayl tizimi bilan ishlash uchun qo‘llaniladi. Unda qattiq diskda saqlanadigan fayllar va kataloglar bilan ishlash uchun juda ko‘p foydali o‘rnatilgan metodlar mavjud. OS moduli bilan ishlaydigan dasturlar OT turiga bog‘liq bo‘lmasdan, boshqa platformalarga osongina ko‘chiriladi.

**OS moduli nima?**

Pythondagi OS moduli operatsion tizim bilan ishlash uchun kerak bo‘ladigan turli funktsiyalar kutubxonasidir. Unga kiritilgan usullar **operatsion tizim turini aniqlash**, **muhit o‘zgaruvchilariga kirish**, **katalog** va **fayllarni boshqarish** imkonini beradi:

* berilgan yo‘lda ob’yektning mavjudligini tekshirish;
* baytlarda o‘lchamlarni aniqlash;
* o‘chirish;
* nomini o‘zgartirish va boshqalar.

OS funksiyalarini chaqirayotganda, ularning ba’zilari joriy opratsion tizim tomonidan qo‘llab-quvvatlanmasligi mumkin.

OS moduli metodlaridan foydalanish uchun kutubxonani “import os” orqali chaqirish kerak. Ushbu ko‘rsatmani dasturning boshida joylashtirish tavsiya etiladi.

OS modulidagi mavjud metodlardan foydalanuvchi turli maqsadlarda foydalanishi mumkin. Quyida operatsion tizim haqida ma’lumot olish imkonini beruvchi eng mashhur metodlar keltirilgan. Shuningdek, kompyuterning qattiq diskidagi xotirada saqlangan fayl va papkalar haqida ma’lumot olish mumkin.

**OT haqida ma’lumot olish**

Joriy OT nomini bilish uchun faqat **name()** metodidan foydalaning. O‘rnatilgan platformaga qarab, u o‘zining qisqa nomini satr ko‘rinishida qaytaradi. Quyidagi dastur Windows 10 kompyuterida ishga tushirildi, shuning uchun **name()** funktsiyasining natijasi sifatida **nt** qaytarildi. Buni odatiy print() funksiyasi bilan ko‘rish mumkin.

import **os** *#os modulini chaqirish*

print(**os**.**name**) *#OT nomini bilish*

**Natijasi:**

nt

**environ()** metodidan foydalanib, kompyuterning konfiguratsiyasi bilan bog‘liq ma’lumotlarni olish mumkin. Undan **os** kutubxonasini chaqirish orqali foydalanuvchi   konsolga yoki satr o‘zgaruvchisiga chiqadigan muhit o‘zgaruvchilari bilan katta lug‘at turidagi ma’lumot oladi. Shunday qilib, tizim diskining nomini (C:\....), katalogining manzilini, tizim nomini va boshqa ko‘plab ma’lumotlarni bilib olish mumkin. Quyidagi misol **environ**() metodidan foydalanishni ko‘rsatadi.

import **os**

print(**os**.**environ**)

**environ**({‘ALLUSERSPROFILETXT’: ‘C:\\ProgramData’, …})

**getenv** funktsiyasidan foydalanib, turli xil muhit o‘zgaruvchilariga kirish mumkin. Buning uchun quyidagi misoldagi kabi kerakli o‘zgaruvchi nomini argument sifatida beriladi. Bunda **print** displeydagi TMP haqidagi ma’lumotlarni konsolga chop etadi.

import **os**

print(os.**getenv**("TMP"))

**Natijasi:**

C:\Users\IT\AppData\Local\Temp

**Ishchi katalogni o‘zgartirish**

Odatda dasturning ishchi katalogi uning manba kodi bilan hujjatni o‘z ichiga olgan katalog bo‘ladi. Buning yordamida, agar u ushbu papkada joylashgan bo‘lsa, faylga mutlaq yo‘lni aniqlashning iloji bo‘lmaydi. Qattiq diskdagi ishchi katalogning to‘liq manzilini qaytaruvchi **getcwd** funksiyasidan foydalanib joriy katalog haqida ma’lumot olinadi. Quyidagi qism dastur, agar ushbu usulning natijasini chop etish uchun o‘tkazsa nima bo‘lishini ko‘rsatadi. Ko‘rib turganingizdek, ishchi katalog C tizim diskidagi dastur katalogidir.

import **os**

print(**os**.**getcwd**())

**Natijasi:**

C:\Users\IT\source\repos\program

Agar so‘ralsa, ishchi katalogni **os** kutubxonasidan **chdir** usuli yordamida xohishga ko‘ra sozlash mumkin. Buni amalga oshirish uchun uni yangi katalogning mutlaq manzilini parametr sifatida o‘tkazish kerak. Agar ko‘rsatilgan yo‘l haqiqatda mavjud bo‘lmasa, dastur xatolik qaytaradi va dastur ishlashi to‘xtatiladi. Quyidagi kod misoli D diskidagi papka deb nomlangan yangi ishchi katalogga o‘tishni ko‘rsatadi.

import **os**

os.**chdir**(r"D:\folder")

**Natijasi:**

D:\folder

**3. Fayl va katalog yo‘lini o‘zgartirish**

Ko‘pincha, faylning to‘liq nomi mavjud bo‘lganda, uning kengaytmasini bilish kerak bo‘ladi. Yoki foydalanuvchi kerakli formatni kiritgan yoki kiritmaganligi noma’lum bo‘lsa, kerakli formatni qo‘shish kabi masalalar mavjud. Ushbu mavzu fayl nomi bilan ishlashning asosiy usullari haqida bo‘ladi.

**Faylga mutlaq yo‘l**

Pythonda faylga mutlaq yo‘lni bilish uchun OS modulidan foydalanish kerak bo‘ladi. **Path** sinfida faylga absolyut yo‘lni qaytaruvchi - **abspath** metodi mavjud:

import **os**

r = os.**path**.**abspath**(‘file.txt ‘)

print(r)

**Natijasi:**

C:\python\file.txt

Buning uchun standart **pathlib** kutubxonasidan ham foydalanish mumkin. U Python 3.4 dan boshlab asosiy kutubxonalarga kiritilgan. Undan oldin uni pip install pathlib buyrug‘i yordamida o‘rnatish kerak edi. U turli xil operatsion tizimlarda fayl tizimi yo‘llari bilan ishlash uchun mo‘ljallangan va bu muammoni hal qilish uchun juda mos keladi.

import **pathlib**

r = pathlib.**Path**(‘file.txt ‘)

print(r)

**Natijasi:**

C:\python\file.txt

**Fayl nomi**

To‘liq yo‘l qatoridan fayl nomini olish uchun os modulining **basename()** metodidan foydalaniladi.

import **os**

file\_name = os.**path**.**basename**(r’C:\python\file.txt ‘)

print(file\_name)

**Natijasi:**

file.txt

Bu yerda maxsus belgilarni oddiy belgilar sifatida ishlatish uchun qatordan oldin **r** belgisi qo‘yilgan.

**Kengaytmasiz fayl**

Endi Pythonda kengaytmasiz fayl nomini qanday topish mumkinligini aniqlaylik. Bunday hollarda os modulining **splittext** metodidan foydalaniladi:

from **os** import **path**

full\_file\_name = path.**basename**(r’C:\python\file.tar.gz ‘)

name = **path**.**splitext**(full\_file\_name)[0]

print(name)

**Natijasi:**

file.tar

Ko‘rinib turibdiki, **gz** arxivatorining oxirgi kengaytmasi o‘chirilgan, siqilmagan **tar** arxivining kengaytmasi esa fayl nomida qolgan.

Agar faqat nom kerak bo‘lsa, birinchi nuqtadan keyin kelgan qabul qilingan satrning barcha belgilarini olib tashlash mumkin. Nuqta belgisini ham olib tashlaylik.

Oldingi misolni quyidagi kod bilan to‘ldiraylik:

index = name.**index**(‘.’)

print(**name**[:**index**])

**Natijasi:**

file

**Fayl kengaytmasi**

Pythonda fayl kengaytmasini xuddi shu **splitext** funksiyasidan foydalanib olish mumkin. Ushbu metod  o‘zidan kortej tipli ma’lumot qaytaradi. Kortejning birinchi elementi fayl nomi bo‘lsa, ikkinchi element kengaytmadir. Bunday holda, bizga ikkinchi element kerak. Ikkinchi elementning indeksi birga teng, chunki kortej elementlari ham massivlar elementlari kabi noldan indexlanadi.

from **os** import **path**

full\_file\_name = path.**basename**(r’C:\python\file.tar.gz ‘)

name = path.**splitext**(full\_file\_name)[1]

print(name)

**Natijasi:**

.gz

Xuddi shunday, **pathlib** kutubxonasining **suffix** metodidan foydalanish ham mumkin:

from **pathlib** import **Path**

print(**Path**(r’C:\python\file.tar.gz‘).**suffix**)

**Natijasi:**

.gz

Ammo ushbu holatlarda ikkita kengaytma mavjud. Ularni **suffixes()** funktsiyasi yordamida aniqlab olish mumkin. U elementlari kengaytmalar bo‘lgan ro‘yxatni qaytaradi. Quyida kengaytmalar ro‘yxatini olish misoli keltirilgan:

from **pathlib** import **Path**

print(**Path**(r’C:\python\file.tar.gz‘).**suffixes**)

**Natijasi:**

[‘.tar’, ‘.gz‘]

To‘liq yo‘ldan fayl nomi yoki kengaytmasini olish yoki faylga mutlaq yo‘lni olish uchun **os** va **pathlib** kutubxonalaridan foydalanish tafsiya etiladi.

**4. Fayllar bilan ishlovchi metodlar**

Python dasturlash tilining asosiy xususiyatlari nafaqat matnli fayllardagi ma’lumotlarni qayta ishlash, balki ularni har tomonlama boshqarish imkonini beradi. Buning uchun bir nechta maxsus kutubxonalar mavjud bo‘lib, ularning o‘rnatilgan funktsiyalari kompyuterdagi fayllar bilan **nusxa ko‘chirish**, **o‘chirish**, **nomini o‘zgartirish** va boshqa turdagi operatsiyalarni ta’minlaydi.

**Fayl mavjudligini tekshirish**

Kompyuterning qattiq diskida uning yo‘qligi bilan bog‘liq bo‘lishi mumkin bo‘lgan matnli hujjat bilan ishlashda oddiy xatoliklarga yo‘l qo‘ymaslik OS kutubxonasining **exists()** metodidan foydalaniladi. Ushbu funksiya orqali ko‘rsatilgan yo‘lda fayl mavjudligini tekshirish mumkin va natijas sifatida mantiqiy True yoki False qiymatlarni qaytaradi. Quyidagi Python misoli D qattiq diskining **test** nomli katalogida **test.txt** va **test10.txt** fayllari mavjudligini tekshiradi. PRINT() funksiyasi esa D da faqat birinchi hujjat mavjudligini ko‘rsatadi.

import os

print(os.path.exists("D:\\test\\test.txt"))

print(os.path.exists("D:\\test\\test10.txt"))

**Natijasi:**

True

False

Ba’zan, hujjatlar bilan ishlashda, berilgan yo‘l bo‘ylab nafaqat ob’yektning mavjudligini tekshirish, balki fayl yoki papka ekanligini tekshirish ham kerak bo‘ladi. Bunday holda OS kutubxonasining **isfile** funksiyasi dasturchiga kompyuterning qattiq diskidagi ma’lum bir manzilda olingan ob’yekt papka emas, balki fayl ekanligiga ishonch hosil qilish imkoniyatini beradi. Bu usul **path** sinfida ham mavjud. Quyidagi misolda **isfile** funksiyasi **test.txt** faylini va D diskning test papkasini argument sifatida qabul qilganda qanday natija ko‘rsatadi. PRINT() funksiyasining chiqishidan ko‘rinib turibdiki, birinchi holatda True, keyin esa False ko‘rsatiladi:

import **os**

print(os.**path**.**isfile**("D:\\test\\test.txt"))

print(os.**path**.**isfile**("D:\\test\\folder"))

**Natijasi:**

True

False

**Faylni nusxalash**

**shutil** kutubxonasi qattiq diskdagi ob’yektlarning nusxalarini yaratish uchun bir nechta foydali funktsiyalarni o‘z ichiga oladi. Faylni manba katalogiga tezda nusxalash uchun **shutil** kutubxonasini qo‘shgandan so‘ng **copyfile()** usulidan foydalanish kerak. Bu yerda birinchi argument asl hujjat, ikkinchi argument esa taklif qilingan yangi fayldir. Esda tutingki, **meta ma’lumotlar** emas, faqat faylning tarkib nusxalanadi. Quyidagi misolda D diskidagi test.txt dan test2.txt ga ma’lumotlarni ko‘chiriladi. **copyfile()** funktsiyasi yangi yaratilgan hujjatning manzilini ham qaytaradi.

import **shutil**

shutil.**copyfile**("D:\\test\\test.txt", "D:\\test\\test2.txt")

**shutil** modulidan o‘rnatilgan **copy()** usuli Pythonga faylni asl nomini saqlab qolgan holda belgilangan papkaga nusxalash imkonini beradi. Quyidagi kod misoli **test.txt** dan ma’lumotni D diskida joylashgan **folder** deb nomlangan katalogdagi ob’yektga qanday ko‘chirishni ko‘rsatadi. **Copy**() funktsiyasi bilan oldingi holatda bo‘lgani kabi, faqat ichki ma’lumotlar uzatiladi, lekin hujjatni yaratish va tahrirlash sanasi haqida ma’lumotlar qaytarilmaydi.

import **shutil**

shutil.**copy**("D:\\test\\test.txt", "D:\\folder")

Matn faylidan ma’lumotni, shuningdek, u haqidagi barcha ma’lumotlarni to‘liq nusxalash uchun tayyor nusxa **copy2** usulidan foydalanish kerak. Uni ishlatish usuli **copy** funksiyasi bilan bir xil. Birinchi parametr o‘rniga bu asl faylning manzili, ikkinchi parametr esa yangi hujjatning joylashuvi va nomini belgilaydi. Quyida kontent va metamaʼlumotlar folser papkasidan **test2.txt** ga koʻchiriladigan misol keltirilgan:

import **shutil**

shutil.**copy2**("D:\\test\\test.txt", "D:\\folder\\test2.txt")

**Fayl o‘chirish**

Ob’yektning nomi va diskdagi aniq joylashuvini bilib olinsa, undan qutulish juda oson. Yuqorida aytib o‘tilgan operatsion tizim kutubxonasidan **remove** usuli bu vazifani hal qilishga yordam beradi. Buning uchun keraksiz faylning to‘liq manzilini unga parametr sifatida beriladi.

Quyida D diskining **test** katalogidagi **test.txt** faylini oʻchirish uchun remove metodidan qanday foydalanishga misol keltirilgan:

import **os**

os.**remove**("D:\\test\\test.txt")

**Fayl hajmini bilib olish**

Fayl hajmini baytlarda qaytaradigan OS modulidagi **getsize** standart funksiyasidan foydalanib, qattiq diskdagi istalgan ob’yektning aniq hajmini aniqlash mumkin. Bu yerda **getsize** funksiyasi argumenti hujjatning kompyuter xotirasidagi joylashuvidir. **getsize** funksiyasi ma’lumotlariga ko‘ra, **test.txt** faylining o‘lchami 7219. **Print()** funksiyasi buni ekranga chop etadi:

import **os**

print(os.**path**.**getsize**("D:\\test\\test.txt"))

**Natijasi:**

7219

Pythonda fayl hajmini hisoblashning yana bir usuli - uni **open** funksiyasi orqali ochish va **seek** funksiyasini chaqirish. U faylning boshidan oxirigacha ma’lumotlarni o‘qish uchun maydonni parametr sifatida uzatilishi kerak. Natijada, matnli faylga yo‘l orqali **tell** usulini chaqirish kerak va keyin uning ishining natijasini konsolga chiqarish uchun print() funksiyasi uchun yuborish kerak.

import **os**

**with** open("D:\\test\\test.txt") **as** **file**:

**file**.**seek**(0, **os.SEEK\_END**)

print(file.tell())

**Natijasi:**

7219

**Fayl nomini o‘zgartirish**

Hujjat nomini nafaqat tizim vositalari, balki OS modulining tayyor funksiyalari yordamida ham o‘zgartirish mumkin. Buning uchun **rename()** metodidan foydalanish maqsadga muvofiq bo‘ladi. Bu metod asl va yangi fayl nomlarini parametr sifatida qabul qiladi. Quyidagi misolda D diskining **test** katalogidagi **test.txt** fayli nomi test1.txt deb o‘zgartirish ko‘rsatilgan:

import **os**

os.**rename**("D:\\test\\test.txt", "D:\\test\\test1.txt")

Xuddi shunday, **shutil** modulidan **copy** metodi yordamida Pythonda fayl nomini o‘zgartirish mumkin. Buning uchun ushbu kutubxonani dasturga import qilish, funktsiyaga hujjatning joylashuvi va yangi nomini berish kifoya qiladi:

import **shutil**

**shutil**.**move**("D:\\test\\test.txt", "D:\\test\\test1.txt")

Shunday qilib, Python tilidagi fayllar bilan ishlash bo‘yicha asosiy operatsiyalar bir nechta o‘rnatilgan kutubxonalar, jumladan **os** va **shutil** yordamida amalga oshiriladi. Ushbu modullarning funktsiyalari diskda **fayl mavjudligini tekshirish**, uni bir nechta **turli rejimlarda nusxalash**, shuningdek **o‘chirish**, **nomini o‘zgartirish** va **hajmini ko‘rsatish** imkonini beradi

**Nazorat savollari:**

1. Mutlaq fayl yo‘li qanday olinadi?
2. Fayl nomini qanday olish mumkin?
3. Fayl kengaytmasini qanday olish mumkin?
4. Fayl kengaytmasini o‘zgartirish qanday amalga oshiriladi?
5. Qanday turdagi fayllar mavjud?
6. Fayl yaratishning qanday usullari mavjud?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**12-mashg‘ulot.** Pythonda OOP asoslari.

**O‘quv savollari:**

1. OOP asoslari. Klasslarni e’lon qilish va nusxasini yaratish.
2. Sinf va obyekt. Sinf konstruktori.
3. \_\_init\_\_() va \_\_del\_\_() metodlari.

**1. OOP asoslari. Klasslarni e’lon qilish va nusxasini yaratish.**

Pythonda dasturda foydalanish mumkin bo‘lgan int, str va shunga o‘xshash ko‘plab o‘rnatilgan turlar mavjud. Ammo Python, sinflar yordamida o‘z tiplarimizni aniqlashga imkon beradi.

Quyidagi o‘xshashlikni ham o‘tkazish mumkin. Har birimiz ismi-sharifi, yoshi, va shu kabi boshqa xususiyatlarga ega bo‘lgan inson haqida qandaydir tasavvurga egamiz. Inson muayyan harakatlarni bajarishi mumkin: yurish, yugurish, o‘ylash va hokazo. O‘ziga xos xususiyatlar va harakatlar majmuasini o‘z ichiga olgan bu ko‘rinishni ***class*** deb atash mumkin. Aniq bir xususiyatlarni o‘zida jamlagan bu class obyektlar uchun shablon xisoblanadi va turli o‘yektlar bir-biri bilan farq qiladi. masalan: bir odamning ism bunaqa bo‘lsa, boshqalari boshqa ismga ega.

**Sinf** class kalit so‘zi yordamida aniqlanadi:

**class** sinf\_nomi:

    sinf\_atributlari

    class\_methods

Sinf ichida uning atributlari aniqlanadi, ular sinfning turli xususiyatlarini saqlaydi va metodlar sinfning funktsiyalari vazifasini bajarishadi.

**Keling, oddiy sinf yarataylik** :

class Person:

pass

Bunda shartli ravishda shaxsni ifodalovchi Person sinfi aniqlanadi. Bunday holda, sinfda hech qanday metodlar yoki atributlar aniqlanmaydi. Biroq, unda biror narsa aniqlanishi kerakligi sababli, pass iborasi sinf funksionalligi o‘rnini bosuvchi sifatida ishlatiladi. Ushbu bayonot sintaktik jihatdan ba’zi kodlarni aniqlash uchun zarur bo‘lganda ishlatiladi, lekin biz buni xohlamaymiz va ma’lum bir kod o‘rniga biz pass bayonotini kiritamiz.

 Sinf yaratilgandan so‘ng, ushbu sinf ob’yektlarini aniqlash mumkin. Masalan :

**class** Person:

**pass**

tomson = Person() # tomson ob’yektini aninqlash

bobbi = Person() # bobbi ob’yektini aninqlash

Person sinfi aniqlangandan so‘ng, Person sinfining ikkita - tomson va bobbi ob’yektlari yaratildi. Klass ob’yektini yaratish uchun maxsus funktsiyadan - konstruktordan foydalaniladi. U class nomi bilan chaqiriladi va class ob’yektini qaytaradi. Ya’ni, bu holda, Person() chaqiruvi konstruktor chaqiruvini ifodalaydi. Har bir yaratilgan classda parametrlarsiz standart konstruktor mavjud bo‘ladi:

tomson = Person() *# Person() - bu Person sinfining ob’yektini qaytaruvchi konstruktor chaqiruvi*

**2. Sinf va obyekt. Sinf konstruktori**

Class metodlari aslida class ichida belgilangan va uning xususiyatlarini belgilaydigan funktsiyalarni ifodalaydi. Masalan, bitta metod bilan Person sinfini aniqlaylik:

**class** Person: # Person klasining aniqlanishi

**def** say\_hello(self):

**print**("Salom")

tom = Person()

tom.say\_hello() # Salom

Bu yerda **say\_hello()** metodi aniqlandi, u shartli ravishda Salom xabarini konsolga chop etadi. Classga tegishli har qanday metodlarini aniqlashda shuni yodda tutish kerakki, ularning barchasi birinchi parametr sifatida ***self*** kalit so‘zini qabul qilishadi. Self bu o‘zi degan ma’noni anglatadi va ushbu klasga tegishli ekanligini bildiradi. Ammo metod chaqirilganda, bu marametr hisobga olinmaydi hamda qiymat berilmaydi.

ob’yekt nomidan foydalanib, classning metodlariga murojaat qilish mumkin. Mavjud metodlarni chaqirish uchun nuqta belgisi qo‘llaniladi - ob’yekt nomidan keyin nuqta qo‘yiladi va undan keyin metod chaqiruvi keladi:

Ob’yekt.metod([metod qabul qiladigan parametrlar])

Masalan, konsolga salomlashishni chop etish uchun say\_hello() metodini chaqirish:

tomson.say\_hello()     # Salom

Natijada, ushbu dastur konsolga "Salom" qatorini chop etadi.

**Agar metod self dan tashqari boshqa parametrlarni qabul qiladigan bo‘lsa, ular self** parametridan keyin belgilanadi va bunday metodni chaqirishda, ushbu parametrlariga qiymatlar berish talab qilinadi:

class Person: *# Person klasining aniqlanishi*

def say(self, msg): *# method*

print(msg)

tom = Person()

tom.say("Hello , HOW ARE YOU?") *# Hello , HOW ARE YOU?*

**Say()** metodi bu yerda aniqlanadi. Bu metod ikkita parametr oladi: **self** va **msg**. Va ikkinchi parametr uchun - ***msg***, metodni chaqirganda ushbu parametrga qiymat berilishi kerak.

Self kalit so‘zi orqali classning atributi va funksiyalariga murojaat qilish mumkin:

**self**.attribute   *# methodning atributiga murojaat qilish*

**self**.method      *# method chaqiruvi*

**Masalan, Person** sinfida ikkita metodni aniqlaymiz :

**class** Person:

**def** say(**self**, msg):

print(msg)

**def** say\_hello(**self**):

**self**.say("Hello JOB")

tom = Person()

tom.say\_hello() *# Hello work*

Bu yerda **say\_hello()** metodi ichida turib **say()** metodiga murojaat amalga oshirilmoqda

self.say("Hello JOB")

Say() metodi o‘ziga qo‘shimcha ravishda parametrlarni (msg parametrini) qabul qilganligi sababli, metod chaqirilganda ushbu parametr uchun qiymat berish kerak.

Bundan tashqari, metod ichida turib, ob’yekt metodini chaqirganda, self kalit so‘zini ishlatishimiz kerak. Aks holda xatolik kelib chiqadi:

**def** say\_hello(**self**):

say("Hello job") *# Xatolik*

**3.\_\_init\_\_() va \_\_del\_\_() metodlari**

**Konstruktorlar.** Konstruktor sinf ob’yektini yaratish uchun ishlatiladi. Shunday qilib, yuqorida, Person sinfining ob’yektlarini yaratganda, hech qanday parametrlarni qabul qilmaydigan va barcha sinflar bilvosita ega bo‘lgan standart konstruktordan foydalanildi:

tom = Person()

Biroq**, \_\_init\_\_()** deb nomlangan maxsus metod yordamida sinflarda konstruktorni aniq belgilash mumkin (har bir tomonda ikkita past chiziqcha). Masalan, keling, Person sinfini unga konstruktor qo‘shish orqali o‘zgartiramiz:

**class** Person:

*# konstruktor*

**def** \_\_init\_\_(**self**):

print("Person obyektining yaratildi")

**def** say\_hello(**self**):

print("Hello")

tom = Person() *# Person obyekti yaratildi*

tom.say\_hello() *# Hello*

Demak, bu yerda Person sinfining kodida konstruktor va say\_hello () metodi aniqlangan. Birinchi parametr sifatida konstruktor ham metodlar kabi – ***self*** ni qabul qiladi. Odatda konstruktorlar obyekt yaratilganda bajariladigan amallarni aniqlash uchun ishlatiladi:

tom = Person()

Person sinfidagi **\_\_init\_\_()** konstruktori chaqiriladi, u konsolga " Person obyekti yaratildi " matnini chop etadi.

**Ob’yekt atributlari.** Atributlar ob’yekt holatini saqlaydi. Class ichidagi atributlarni aniqlash va o‘rnatish uchun ***self*** so‘zidan foydalanish mumkin. Masalan , quyidagi Person sinfini aniqlaymiz :

**class** Person:

**def** \_\_init\_\_(**self**, nom):

**self**.nom = nom *# shaxning nomi*

**self**.yosh = 1 *# shaxsning yoshi*

tomson = Person("Tomson")

*# Atributga murojaat*

*# qiymatni qabul qilish*

print(tomson.nom) *# Tom*

print(tomson.yosh) *# 1*

*# Qiymatni o‘zgartirish*

tomson.yosh = 37

print(tomson.yosh) *# 37*

Endi Person klassi konstruktori yana bitta parametr - **nom** ni qabul qiladi. Ushbu parametr orqali yaratilgan person ob’yektining nomi konstruktorga uzatiladi.

Konstruktor ichida ikkita atribut o‘rnatilgan – **nom** va **yosh** (shartli ravishda, shaxsning ismi va yoshi):

**def** \_\_init\_\_(**self**, nom):

**self**.nom = nom

**self**.yosh = 1

**self.nom** atributi **nom** o‘zgaruvchisiga tenglashtirib qo‘yildi. ***yosh*** atributining boshlang‘ich qiymati 1 ga teng deb qabul qilindi.

Agar sinfda **\_\_init\_\_** konstruktoridan foydalanilgan bo‘lsa, endi standart konstruktorni chaqira olmaymiz. Endi biz aniq belgilangan **\_\_init\_\_** konstruktorimizni chaqirishimiz kerak, unga **nom** parametri uchun qiymat berilishi kerak:

tomson = Person("Tomson")

Keyinchalik ushbu nom orqali ob’yektning atributlariga murojaat qilib, uni olish va o‘zgartirish mumkin:

print(tomson.nom)*# nom atributining qiymatini olish*

tomson.yosh = 37 *# yosh atributining qiymatini o‘zgartiring*

Asosan, class ichida atributlarni aniqlamasdan uni ob’yektning o‘ziga biriktirib qo‘yish mumkin. Python bu atributni dinamik ravishda ob’yektga qo‘shishga imkon beradi:

**class** Person:

**def** \_\_init\_\_(**self**, nom):

**self**.nom = nom *# shaxsning nomi*

**self**.yosh = 1 *# shaxsning yoshi*

tomson = Person("Tomson")

tomson.company = "Microsoft"

print(tomson.company) *# Microsoft*

Bu yerda kompaniya atributi dinamik ravishda o‘rnatildi. Bu atribut person ob’yektining ish joyini o‘zida saqlaydi. Va o‘rnatgandan so‘ng, uning qiymatiga murojaat qilib, qiymatini olish mumkin. Shuni esda turish kerakki, agar atributni aniqlashdan oldin unga murojjat qilishga harakat qilinsa, dastur xatolik qaytaradi:

tomson = Person("Tomson")

print(tomson.company) *# ! Xatolik - AttributeError:*

*# Person object has no attribute company*

Sinf ichidagi ob’yektning atributlariga murojaat qilish uchun uning metodlarida self so‘zi ham qo‘llaniladi:

**class** Person:

**def** \_\_init\_\_(**self**, nom):

**self**.nom = nom *# shaxsning nomi*

**self**.yosh = 1 *# shaxsning yoshi*

**def** display\_info(**self**):

print(f"Nomi: {**self**.nom} Yoshi: {**self**.yosh}")

tomson = Person("Tomson")

tomson.display\_info() *# Nomi: Tomson Yoshi: 1*

Bu konsolga ma’lumotlarni chop etadigan ***display\_info()*** metodini belgilaydi**.** Metoddagi ob’yekt atributlariga murojaat qilish uchun esa ***self*** so‘zi ishlatiladi: ***self****.nom* va ***self****.yosh*

**Ob’yektlarni yaratish.** Yuqorida bitta ob’yekt yaratilgan. Ammo shunga o‘xshash tarzda class ning boshqa ob’yektlarni yaratish mumkin:

**class** Person:

**def** \_\_init\_\_(**self**, nom):

**self**.nom = nom *# shaxsning nomi*

**self**.yosh = 1 *# shaxsning yoshi*

def display\_info(**self**):

print(f"Nomi: {**self**.nom} Yoshi: {**self**.yosh}")

tomson = Person("Tomson")

tomson.yosh = 37

tomson.display\_info() *# Nomi: Tomson yoshi: 37*

bobbi = Person("Bobbi")

bobbi.yosh = 41

bobbi.display\_info() *# Nomi: Bobbi Yosh: 41*

Ushbu dasturda Person sinfining ikkita obyekti: tomson va bobbi yaratilgan. Ushbu ob’yektlar Person sinfga tegishli bo‘lib, ular bir xil atributlar va metodlar to‘plamiga egalar. Biroq ular bir-biridan xususiyatlari bilan farq qilishadi.

Dastur bajarilganda, Python self parametri dinamik ravishda aniqlanadi - bu chaqiriladigan ob’yektni ifodalaydi. Masalan, satrda:

tomson.display\_info() *# Nomi: Tomson yoshi: 37*

bobbi.display\_info() *# Nomi: Bobbi Yosh: 41*

**Nazorat savollari:**

1. Pythonda ob’yekt tushunchasiga ta’rif bering.
2. Pythonda class tushunchasiga tarif bering.
3. Classning methodlari qanday yaratiladi?
4. Classga murojaat qanday amalga oshiriladi?
5. Class ob’yekti qanday hosil qilinadi?
6. Metodga qanday murojaat qilinadi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**13-mashg‘ulot**. Pythonda Vorislik tushunchasi.

**O‘quv savollari:**

1. Vorislik.
2. Maxsus metodlar.
3. Klass xususiyatlari.
4. Dekoratorlar.

**1. Python tilidagi “Merosxo‘rlik” tushunchasi, atributlari va xossalari**

OOP da merosxo‘rlik konsepsiyasi mavjud class asosida yangi class yaratish imkonini beradi. Inkapsulyatsiya bilan bir qatorda merosxorlik ob’yektga yo‘naltirilgan dasturlashning asoslaridan hisoblanadi.

Merosxorlik konsepsiyasining asosiy tushunchalari: ***subklass*** va ***superclass*** lardir. ***Subklass*** barcha umumiy atributlar va metodlarni yuqori classdan meros qilib oladi. Shuningdek superclass, asosiy (**base class**) yoki ota ( **parent class**) deb atalsa, subclass - voris class ( **derived class**) yoki bola class (**child class**) deb ataladi.

Merosxorlik classini yaratish sintaksisi quyidagicha ko‘rinishda bo‘ladi:

**class** subclass (superclass):

    subclass\_metodlari

Masalan, shaxsni ifodalovchi Person sinfi mavjud bo‘lsin:

**class** Person:

**def** \_\_init\_\_(**self**, nom):

**self**.\_\_nom = nom

@property

**def** nom(**self**):

**return** **self**.\_\_nom

**def** display\_info(**self**):

print(f"Nomi: {**self**.\_\_nom} ")

Aytaylik, bizga qandaydir korxonada ishlaydigan ishchilar classi kerak. Biz noldan yangi sinf yaratishimiz mumkin, masalan, *Employee* sinfi:

**class** Employee:

**def** \_\_init\_\_(**self**, nom):

**self**.\_\_nom = nom

@property

**def** nom(**self**):

**return** **self**.\_\_nom

**def** display\_info(**self**):

print(f"Nomi: {**self**.\_\_nom} ")

**def** work(**self**):

print(f"{**self**.nom} works")

*Biroq, Employee* classi Person classi bilan bir xil atribut va metodlarga ega bo‘lishi mumkin, chunki xodimlar ham inson classiga mansub ob’yeklar hisoblanadi. Shunday qilib, yuqoridagi  ***Employee***  sinfida faqat ***work*** metodi qo‘shiladi va qolgan kod Person sinfining funksionalligidan nusxalab olinadi. Agar mersoxorlikda foydalanilmasa, bir xil funksiyanallik ikkala sinfda ham takrorlanaveradi. Bunday holda merosxorlik konsepsiyasidan foydalanish ancha kodni tejalishiga olib keladi.

Shunday qilib, ***Person*** sinfidan ***Employee*** sinfini meros qilib olaylik :

**class** Person:

**def** \_\_init\_\_(**self**, nom):

**self**.\_\_nom = nom

@property

**def** nom(**self**):

return **self**.\_\_nom

**def** display\_info(**self**):

print(f"Nomi: {**self**.\_\_nom} ")

**class** Employee(Person):

**def** work(**self**):

print(f"{**self**.nom} works")

tomson = Employee("Tomson")

print(tomson.nom) *# Tomson*

tomson.display\_info() *# Nomi: Tomson*

tomson.work() *# Tomson works*

*Employee*  sinfi ***Person*** sinfining funksiyalarini to‘liq o‘ziga nusxalab oladi, faqat **work()** metodi *Employee* sinfiga qo‘shimcha tarzda qo‘shiladi. *Shunga ko‘ra, Employee* sinfi ob’yektini yaratishda ***Person*** dan meros qilib olingan konstruktordan foydalanish mumkin :

tomson = Employee ("Tomson")

Shuningdek, meros qilib olingan atributlar/ xususiyatlar va metodlarga murojaat qilish mumkin:

print(tomson.nom) *# Tomson*

tomson.display\_info() *# Nomi: Tomson*

Shuni yodda tutish kerakki, Employee sinfi uchun Person sinfining yopiq atributi bo‘lgan **\_\_nom** atributiga kirishga ruxsat yo‘q. Masalan, ***work()*** metodi tarkibida **self.\_\_nom** xususiy atributiga murojaat qila olmaymiz :

def work(self):

print(f"{self.\_\_nom} works") *# Xatolik!*

**Ko‘p merosxorlik**

Python tilining ajralib turadigan xususiyatlaridan biri - bu bir ko‘p merosxorlik konsepsiyasini qo‘llab-quvvatlashligidir. Ya’ni bitta sinf bir nechta sinflardan meros olishi mumkin:

**class** Employee:

**def** work(**self**):

print("Employee jobs")

**class** Students:

**def** study(**self**):

print("Students studying")

**class** WorkingStudents(Employee, Students):

pass

tomson = WorkingStudent()

tomson.job() *# Employee jobs*

tomson.study() *# Students studying*

U firma xodimini ifodalovchi  Employee sinfini va talabalarini ifodalovchi  Studentssinfini belgilaydi*.*Ishlayotgan talabani ifodalovchiWorkingStudents sinfi  tarkibida hech qanday metod va atributlar aniqlanmaganligi pass operatori bilan to‘ldirib qo‘yilgan holos.

WorkingStudents sinfi ikkita sinfdan Employee va  Studentsfunksionallikni meros qilib oladi. Shunday ekan ushbu sinf ob’yektida ikkala sinfning metodlarini chaqirish imkoniyati mavjud.

Shu bilan birga, meros qilib olingan sinflar funktsional jihatdan murakkabroq bo‘lishi mumkin, masalan:

**class** Employee:

**def** \_\_init\_\_(**self**, nom):

**self**.\_\_nom = nom

@property

**def** nom(**self**):

**return** **self**.\_\_nom

**def** work(**self**):

print(f"{**self**.nom} works")

**class** Students:

**def** \_\_init\_\_(**self**, nom):

**self**.\_\_nom = nom

@property

**def** nom(**self**):

**return** **self**.\_\_nom

**def** study(**self**):

print(f"{**self**.nom} studying")

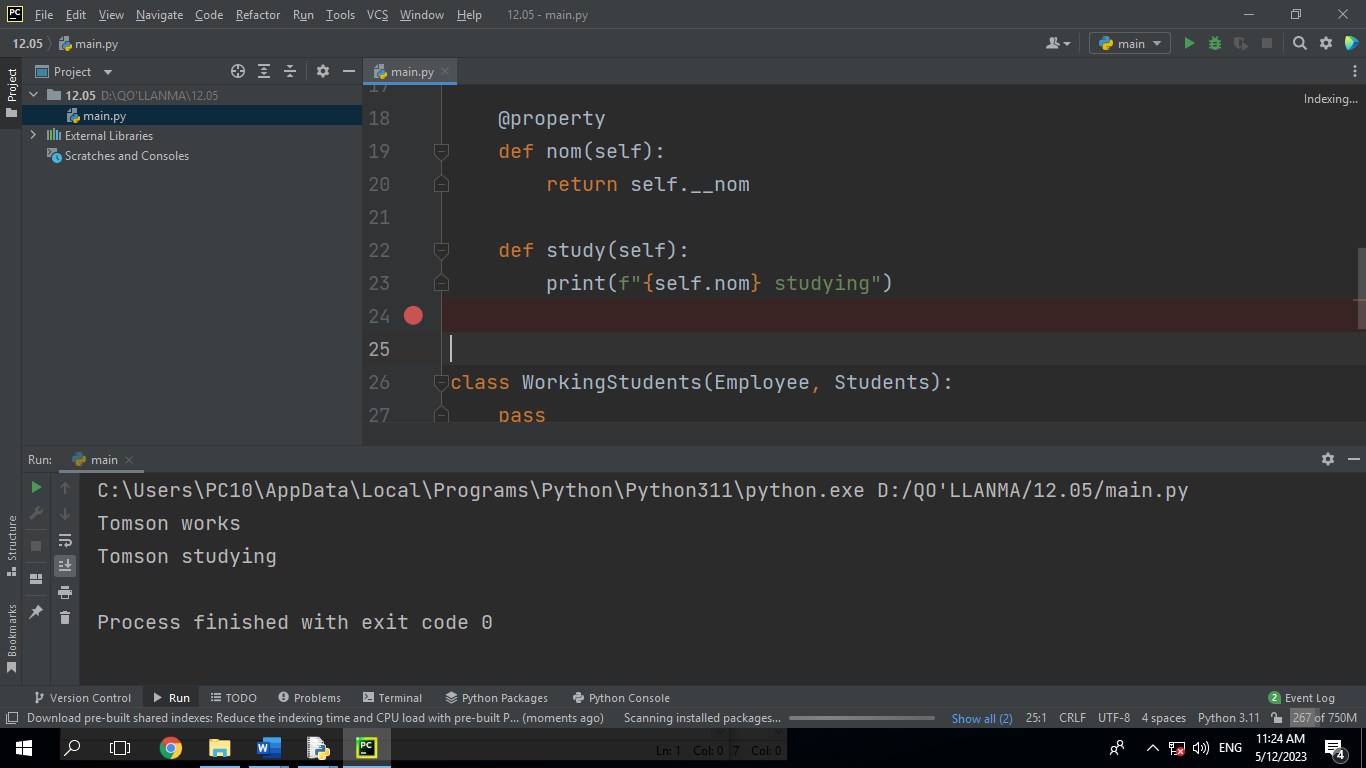
**class** WorkingStudents(Employee, Students):

**pass**

tomson = WorkingStudents("Tomson")

tomson.work()

tomson.study()



**2. Maxsus 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 yoki qisqa qilib dunder metodlar deb ataladi. Dunder metolar 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\_\_',

'\_\_gt\_\_',

'\_\_hash\_\_',

'\_\_init\_\_',

'\_\_init\_subclass\_\_',

'\_\_le\_\_',

'\_\_lt\_\_',

'\_\_module\_\_',

'\_\_ne\_\_',

'\_\_new\_\_',

'\_\_reduce\_\_',

'\_\_reduce\_ex\_\_',

'\_\_repr\_\_',

'\_\_setattr\_\_',

'\_\_sizeof\_\_',

'\_\_str\_\_',

'\_\_subclasshook\_\_',

'\_\_weakref\_\_',

'make',

'model',

'narh',

'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'zilari bilan tanishamiz.

**OBYEKT HAQIDA MA'LUMOT**

Obyektga print() yoki str() orqali murojat qilinganda 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,narh):

"""Avtomobilning xususiyatlari"""

self.make = make

self.model = model

self.rang = rang

self.yil = yil

self.narh = narh

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

Natija: <\_\_main\_\_.Avto object at 0x00000238A6DAE0C8>

Qandaydur tushunarsiz ma'lumot. Ekrandagi natijadan biz faqat avto1 obyektimiz Avto klassiga tegishli ekanini ko‘ramiz. Qanday qilib shuning o‘rniga obyekt haqida tushunarliroq ma'lumot olishimiz mumkin?

Gap shundaki biz har gal obyketga print() (yoki str() yoki repr()) orqali murojat qilganimizda, Python obyket ichida \_\_str\_\_ yoki \_\_repr\_\_ metodlariga murojat qiladi. Agar biz bu metodlarni yozmagan bo‘lsak, yuqoridagi kabi umumiy ma'lumot qayataradi.

Biz ushbu metodlarni yangidan yozib, biz istagan ma'lumotni qayataradian qilishimiz mumkin. Odatda, yuqoridagi ikki metoddan birini yozish kifoya. Odatda, \_\_repr\_\_ umumiyorq, \_\_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 yuoqirdagi klassimizga\_\_repr\_\_metodini qo‘shamiz:

class Avto:

\_\_num\_avto = 0

"""Avtomobil klassi"""

def \_\_init\_\_(self,make,model,rang,yil,narh):

"""Avtomobilning xususiyatlari"""

self.make = make

self.model = model

self.rang = rang

self.yil = yil

self.narh = narh

Avto.\_\_num\_avto += 1

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)

Natija: Avto: Qora GM Malibu

Mana endi natijamiz ancha tushunarli ko‘rinishda chiqdi.

**SUPER KLASS VA VORIS KLASS**

Obyektga yo‘naltirilgan dasturlashning qulayliklaridan biri bu klasslardan boshqa klass yaratish imkoniyati. Bizga kerak bo‘lgan yangi klass, avval yaratilgan boshqa klass bilan o‘xshashlik joylari bo‘lsa, biz 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 boshlanishiga Shaxs klassini yaratamiz, bu klassimiz keyinchalik boshqa klasslar uchun super klass vazifasini bajaradi:

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.")

Natija: Hasan Alimov. Passport:FB001122, 1995-yilda tug`ilgan. 26 yoshda.

**VORIS KLASS YARATISH**

Endi avvalgi darsimizda yaratgan Talaba klassimizni qaytadan yaratamiz. Bu safar, avvalgidan farqli ravishda, Talaba ni yaratishda, Shaxs dan super klass sifatida 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

5-qatorda (def \_\_init\_\_ ichida) klassimiz super klassning xususiyatlarini meros olishini ko‘rsatdik

Yangi yaratgan Talaba klassimiz Shaxsning barcha xususiyatlari va metodlariga ega bo‘ladi.

talaba = Talaba("Valijon","Aliyev","FA112299",2000)

print(talaba.get\_info())

Natija: Valijon Aliyev. Passport:FA112299, 2000-yilda tug`ilgan

Talaba klassi uchun alohida get\_info() metodini yozmagan bo‘lsakda, bu metod Talabaga Shaxsdan meros o‘tdi.

Huddi shu kabi get\_age() metodiga ham murojat qilishimiz mumkin:

>>>print(talaba.get\_age(2021))

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 yarataylik. Avvalosiga, talabaning bosqichi va ID raqamini xususiyat sifatida qo‘shamiz. Bunda ID raqami obyekt yaratilishida parameter sifatida uzatiladi, bosqich esa standart qiymatga ega.

class Talaba(Shaxs):

"""Talaba klassi"""

def \_\_init\_\_(self, ism, familiya, passport, tyil,idraqam):

"""Talabaning xususiyatlari"""

super().\_\_init\_\_(ism, familiya, passport, tyil)

self.idraqam = idraqam

self.bosqich = 1

Endi yangi, Talaba obyektini yaratishda qo‘shimcha idraqam parametrini ham kiritish talab qilinadi:

Copy

talaba = Talaba("Valijon","Aliyev","FA112299",2000,"0000012")

So‘ngra, bu qiymatlarni qaytaruvchi alohida metodlar yozamiz:

class Talaba(Shaxs):

"""Talaba klassi"""

def \_\_init\_\_(self, ism, familiya, passport, tyil,idraqam):

"""Talabaning xususiyatlari"""

super().\_\_init\_\_(ism, familiya, passport, tyil)

self.idraqam = idraqam

self.bosqich = 1

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 raqami:{talaba.get\_id()}")

Valijon Aliyev. Passport:FA112299, 2000-yilda tug`ilgan. ID raqami:0000012

>>>print(f"{talaba.get\_bosqich()}-bosqich talabasi")

1-bosqich talabasi

Shu zayilda yangi klassimizga istalgancha yangi xususiyatlar va metodlar qo‘shishimiz mumkin. Bunda, agar yangi xususiyat yoki metod super klassga ham aloqador bo‘lsa uni birdan super klassga qo‘shish tavsiya qilinadi.

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 qilish mumkin. Avvalgi misolimizdagi get\_info() super metodini ko‘raylik, bu metod talabaning ismi, familiyasi, passport raqami va tug‘ilgan yilini qaytaradi:

>>> print(talaba.get\_info())

Valijon Aliyev. Passport:FA112299, 2000-yilda tug`ilgan

Endiget\_info() metodi talabaga mos ma'lumotlar qaytarishi uchun, Talaba klassi ichida huddi shu nomli metodni qayta yozamiz:

class Talaba(Shaxs):

"""Talaba klassi"""

def \_\_init\_\_(self,ism,familiya,passport,tyil,idraqam):

"""Talabaning xususiyatlari"""

super().\_\_init\_\_(ism, familiya, passport, tyil)

self.idraqam = idraqam

self.bosqich = 1

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

Metodni tekshirib ko‘ramiz:

>>> print(talaba.get\_info())

Valijon Aliyev. 1-bosqich. ID raqami: 0000012

**OBYEKT ICHIDA OBYEKT**

Ba'zida klassimiz xususiyatlar va ular bilan ishlaydigan metodlarga to‘lib ketishi, bu esa o‘z navbatida obyektga murojat qilishni qiyinlashitirishi mumkin. Shunday holatlarda ba'zi xususiyatlarni alohida klass ko‘rinishida yozish va keyinchalik bu klassdan yaratilgan obyektni 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 bo‘ladi.

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,idraqam,manzil):

"""Talabaning xususiyatlari"""

super().\_\_init\_\_(ism, familiya, passport, tyil)

self.idraqam = idraqam

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:

talaba\_manzil = Manzil(12,'Olmazor',"Bog‘bon","Samarqand")

talaba = Talaba("Valijon","Aliyev","FA112299",2000,"0000012",talaba\_manzil)

Obyekt ichidagi obyektning xususiyatlari va metodlariga ham avvalgidek nuqta orqali murojat qilishimiz mumkin:

>>> print(talaba.manzil.get\_manzil())

Samarqand viloyati, Bog‘bon tumani, Olmazor ko‘chasi, 12-uy

>>> print(talaba.manzil.tuman)

**Amaliyot uchun masalalar**

1. 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=[])
2. Fan degan yangi klass yarating va bu klassdan turli fanlar uchun alohida obyektlar yarating.
3. 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.
4. 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.
5. Yuqoridagi Shaxs klassidan boshqa turli voris klasslar yaratib ko‘ring (masalan Professor, Foydalanuvchi, Sotuvchi, Mijoz va hokazo)
6. Har bir klassga o‘ziga hoz xususiyatlar va metodlar yuklang.
7. Barcha yangi klasslar uchun get\_info() metodini qayta yozib chiqing.
8. Voris klasslardan yana boshqa voris klass yaratib ko‘ring. Misol uchun Foydalanuvchi klassidan Admin klassini yarating.
9. Admin klassiga foydalanuvchida yo‘q yangi metodlar yozing, masalan, ban\_user() metodi konsolga "Foydalanuvchi bloklandi" degan matn chiqarsin.

**Nazorat savollari:**

1. Merosxorlik tushunchasiga tarif bering;
2. Pythonda merosxorlik qanday amalga oshiriladi?
3. Inkopsulyatsiya nima maqsadda ishlatiladi?
4. Merosxor classning qanday afzalliklari mavjud?
5. Subclass va Superclass tushunchasiga tarif bering.

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.

**1-mashg‘ulot.** PyQt5 paketi va uning imkoniyatlari bilan tanishish.

O‘quv savollari:

1. PyQt5 paketining imkoniyatlari. PyQt5 paketini o‘rnatish.
2. PyQt5 paketini pip yordamida o‘rnatish.
3. QtDesigner dasturini o‘rnatish hamda uning imkoniyatlari bilan tanishish.

**1. PyQt5 paketining imkoniyatlari. PyQt5 paketini o‘rnatish**

Python, boshqa dasturlash tillari kabi, grafikalar bilan ishlash qobiliyatiga ega. Quyida Pythonning eng yaxshi 5 ta grafik kutubxonasi haqida foydali ma’lumotlar keltirilgan:

1. PyQt5

2. Python Tkinter

3. PySide 2

4. Kivi

5. wxPython

Dasturlash tillaridan foydalanib turli dasturlar tuzish davomida GUI dasturlar tuzish degan tushunchaga duch kelinadi. Xo‘sh GUI o‘zi nima?

**GUI** (Grafics User Interface)- bu shakllar, hujjatlar, testlar va boshqalar kabi turli vaziyatlarda foydalanuvchilardan javob olish uchun dasturchi tomonidan ishlab chiqilgan interaktiv muhit. Bu foydalanuvchiga an’anaviy Buyruqlar qatori interfeysi (CLI) ga qaraganda chiroyli interaktiv ekranni taqdim etadi.

PyQt5 - bu Python uchun grafik foydalanuvchi interfeysi (GUI). Bu dasturchilar orasida juda mashhur va GUI kodlash yoki QT dizayner dasturi tomonidan ishlab chiqilishi mumkin. QT freymworki foydalanuvchi interfeyslarini yaratish uchun vidjetlarni bevosita sichqoncha yordamida tanlab kerakli joyga joylashtirish imkonini beruvchi vizual tizimdir.

Bu bepul va ochiq kodli bog‘lovchi dasturiy ta’minot bo‘lib, crossplatformali ilovalarni ishlab chiqish uchun xizmat qiladi. U Windows, Mac, Android, Linux, Arduino va Raspberry PI da qo‘llanilishi mumkin.

PyQt5 ni o‘rnatish uchun quyidagi buyruqdan foydalaniladi:

**pip install pyqt5**

Bu yerda oddiy koddan foydalanib, PyQt5 kutubxonasidan qanday foydalanishni ko‘rsatib o‘tilgan:

from **PyQt5**.**QtWidgets** import **QApplication**, **QMainWindow**

import **sys**

​

**class** Window(**QMainWindow**):

**def** \_\_init\_\_(**self**):

       super().\_\_init\_\_()

​

       self.**setGeometry**(300, 300, 600, 400)

       self.**setWindowTitle**("PyQt5 window")

       self.**show**()

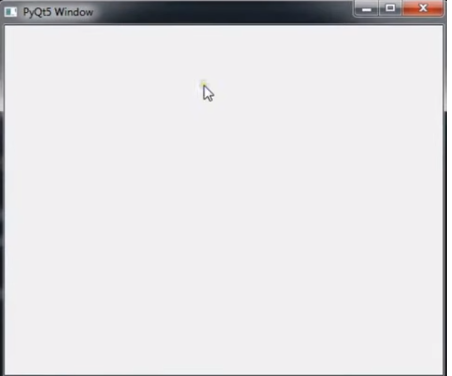
​

app = **QApplication**(sys.**argv**)

window = **Window**()

sys.**exit**(**app**.**exec\_**())

**Dastur natijasi:**



*PyQt 5 kutubxonasi interfeysi ko‘rinishi*

PyQt – bu python dasturlash tili va Qt platformasini bir-biri bilan hamjihatlikda ishlashiga imkon beruvchi mukammal bir kutubxona. U murakkab grafik interfeyslar uchun maxsus vidjetlar yaratish uchun o‘rnatilgan vidjetlar va asboblarning boy tanlovini, shuningdek, ma’lumotlar bazalariga ulanish va ular bilan o‘zaro ishlash uchun mustahkam SQL ma’lumotlar bazasini qo‘llab-quvvatlaydi.

**2. PyQt5 paketini pip yordamida o‘rnatish**

**PyQt -** Python dasturlash tili uchun Qt grafik freymvorki uchun Python kengaytmasi sifatida amalga oshirilgan kengaytmalar (bog‘lashlar) to‘plami.

PyQt Britaniyaning Riverbank Computing kompaniyasi tomonidan ishlab chiqilgan python kutubxonasi. PyQt Qt tomonidan qo‘llab-quvvatlanadigan barcha platformalarda ishlaydi: Linux, UNIX, macOS va Windows va shu kabi operatsion tizimlar.  Hozirgi kungacha PyQt ning 3 ta versiyasi ishlab chiqilgan: PyQt4, PyQt5 va PyQt6. PyQt kutubxonasi GPL (pythonning 2 va 3 versiyalari) va tijorat litsenziyalari ostida tarqatiladi.

PyQt deyarli to‘liq Qt imkoniyatlarini qamrab olgan. Bu esa 600 dan ortiq sinflar, 6000 dan ortiq funktsiyalar va metodlar degani.

**PyQt** shuningdek, QtDesigner nomli GUI dizaynerini ham o‘z ichiga oladi.  QtDesigner dasturida yaratilgan \*.ui formatli dizayn faylini Python faylga o‘girib, qayta ishlash yoki to‘g‘ridan-to‘g‘ri \*.ui fayli bilan \*.py fayllari o‘rtasida bog‘lanish hosil qilish uchun ***pyuic*** kutubxonasidan foydalaniladi. QtDesigner dasturi PyQtni tez prototiplash uchun juda foydali vositaga aylantiradi. Bundan tashqari, Qt Designer dasturiga Pythonda yozilgan yangi grafik boshqaruv elementlarini qo‘shish mumkin.

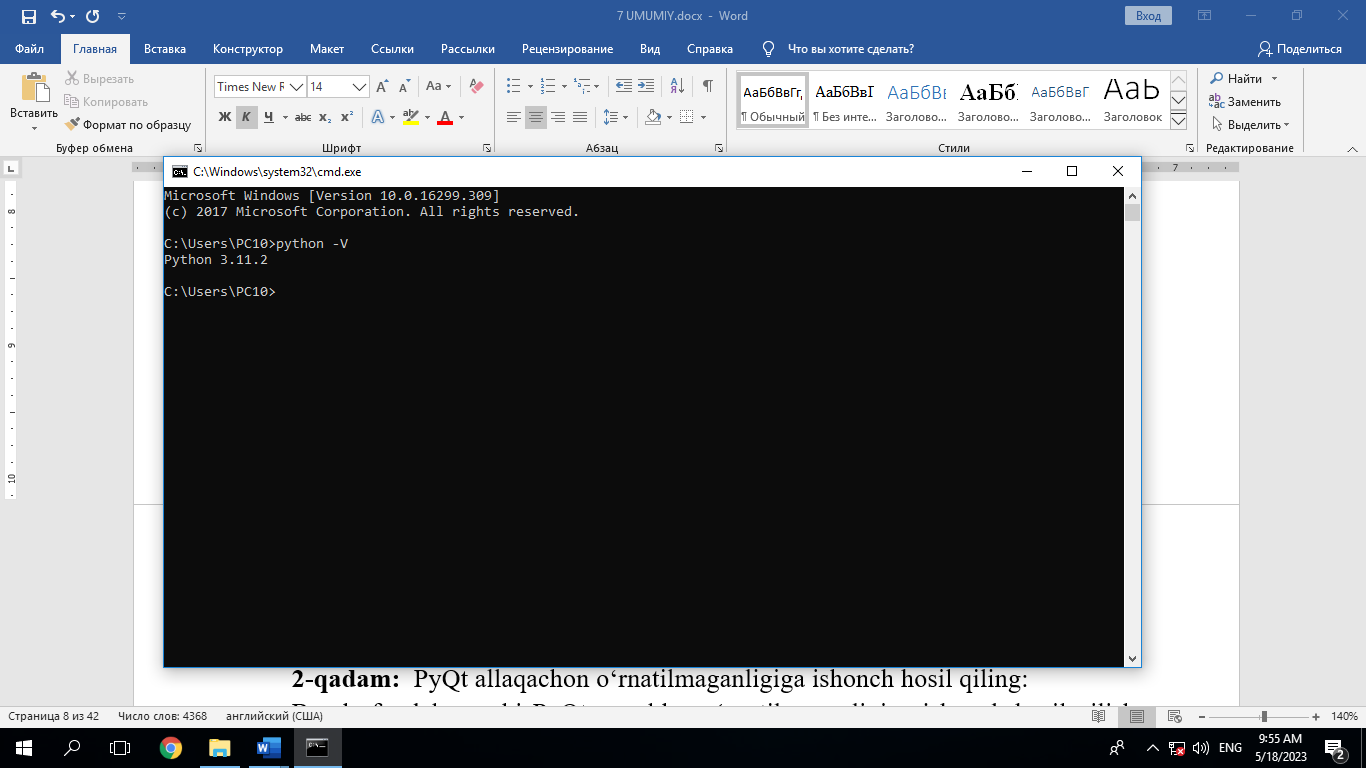
**PyQt5 paketini o‘rnatish.** PythonningPyQt5 paketini Windows operatsion tizimiga o‘rnatish uchun quyidagi amallarni bajarish kerak:

**1-qadam:**  Python operatsion tizimga o‘rnatilganligiga ishonch hosil qilish kerak. Buning uchun klaviaturadan **WIN+R** klaviaturalarini bir vaqtda bosiladi va ekranda **Выполнить** oynasi paydo bo‘ladi. Bu oynaga **CMD** buyrug‘i yoziladi va ok tugmachasini bosiladi:

**Выполнить** oynasi ko‘rinishi

Shundan so‘ng consol oynasi ochiladi va u yerga operatsion tizimda python o‘rnatilganligini bilish maqsadida oddiy pythonning versiyasini bilib beruvchi buyruqni kiritamiz: **python -V**

Shundab so‘ng, agar tizimda python allaqachon o‘rnatilgan bo‘lsa, buyruqning natijasi pythonning versiyasi bo‘ladi :



*Python versiyasini aniqlash*

Aks holda, xatolik yuzaga keladi va python o‘rnatilishi kerak bo‘ladi.

**2-qadam.**  Kompyuterga PyQt5 o‘rnatilmaganligiga ishonch hosil qilish.

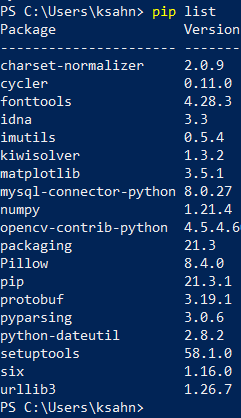
Bunda foydalanuvchi PyQt avvaldan oʻrnatilmaganligiga ishonch hosil qilish uchun quyidagi buyruqni kiritishi kerak.

Consolni ochiladi va u yerga pythonning o‘rnatilgan barcha paketlari ro‘yxatini chiqaruvchi buyruqni yoziladi: **pip list.**

Shunday qilib PyQt5 tizimga o‘rnatilmagan yoki o‘rnatilganligini bilib olinadi.

**3-qadam.** **PyQt-ni o‘rnatish.** Demakoperatsion tizimga python dasturi o‘rnatilganligini va PyQt5 paketi o‘rnatilmaganligiga ishonch hosil qilindi.

Endi PyQt5 paketini tizimga o‘rnatish uchun foydalanuvchi quyida keltirilgan buyruqni konsolga kiritishi kerak: **pip install PyQt5**



*O‘rnatilgan paketlar ro‘yxatini aniqlash.*

Shundan so‘ng hech qanday xatoliklar sodir bo‘lmasa, PyQt5 tizimga muvaffaqiyatli o‘rnatiladi va 2-qadamni takrorlash orqali foydalanuvchi PyQt o‘rnatilganligini tekshirishi mumkin.



*PyQt 5 paketini o‘rnatish*

Shunday qilib Windows muhitida Python uchun PyQtni muvaffaqiyatli o‘rnatildi.

**3. QtDesigner dasturini o‘rnatish hamda uning imkoniyatlari bilan tanishish.**

QtDesigner dasturi **pyqt5-tools** to‘plami bilan birga o‘rnatiladi. Va uni o‘rnatish juda oddiy. Kompyuterning konsol oynasiga quyidagi buyruqni yoziladi va ishga tushiriladi:

**pip install pyqt5-tools**

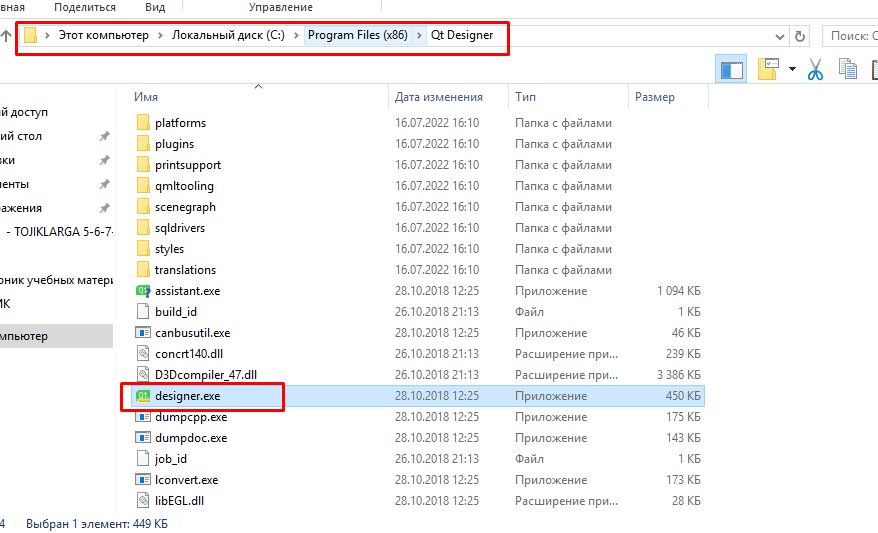
Agar yuqoridagi buyruq ishlamasa, unda quyidagi- pip buyrug‘i pip3 buyrug‘I bilan almashtirilgan buyruqni sinab ko‘rish tavsiya etiladi:

**pip3 install pyqt5-tools**

**QtDesignerni ishga tushiring.**

Agar hammasi normada o‘rnatilgan bo‘lsa, unda bemalol dasturni ishga tushirib undan foydalanish mumkin bo‘ladi. Buning uchun oldin QtDesigner dasturi o‘rnatilgan joyni topish kerak. U odatda python o‘rnatilgan kotalogda bo‘ladi:

*C:\Program Files (x86)\Qt Designer*



*Qt Designer dasturini ishga tushirish .*

Odatda yorliq yaratish va uni ish stoliga joylashtirish tavsiya etiladi (38-rasm):

Endi dizayn dasturini ishga tushirish va python tilidagi dasturning grafik interaktiv qismini yaratishni boshlash mumkin.

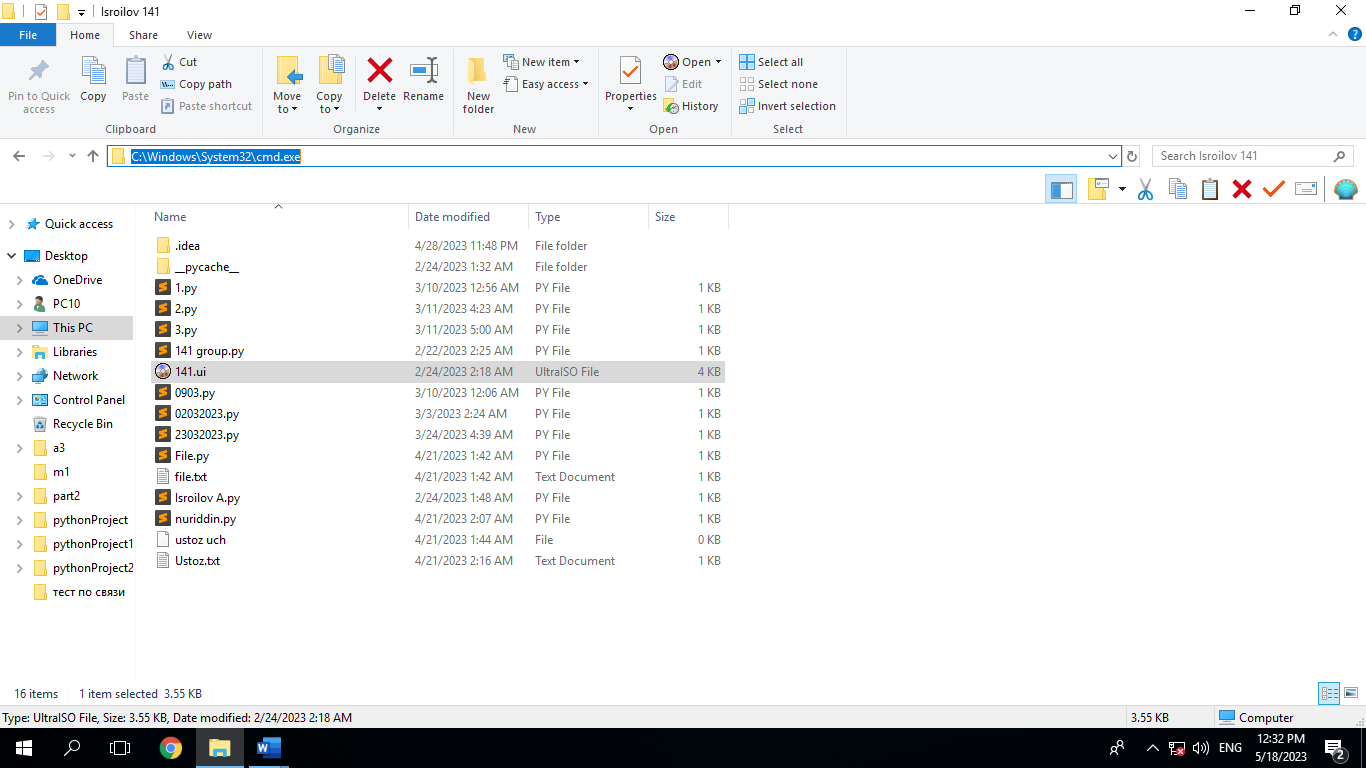


*Qt Designer yorliq dasturini yaratish*

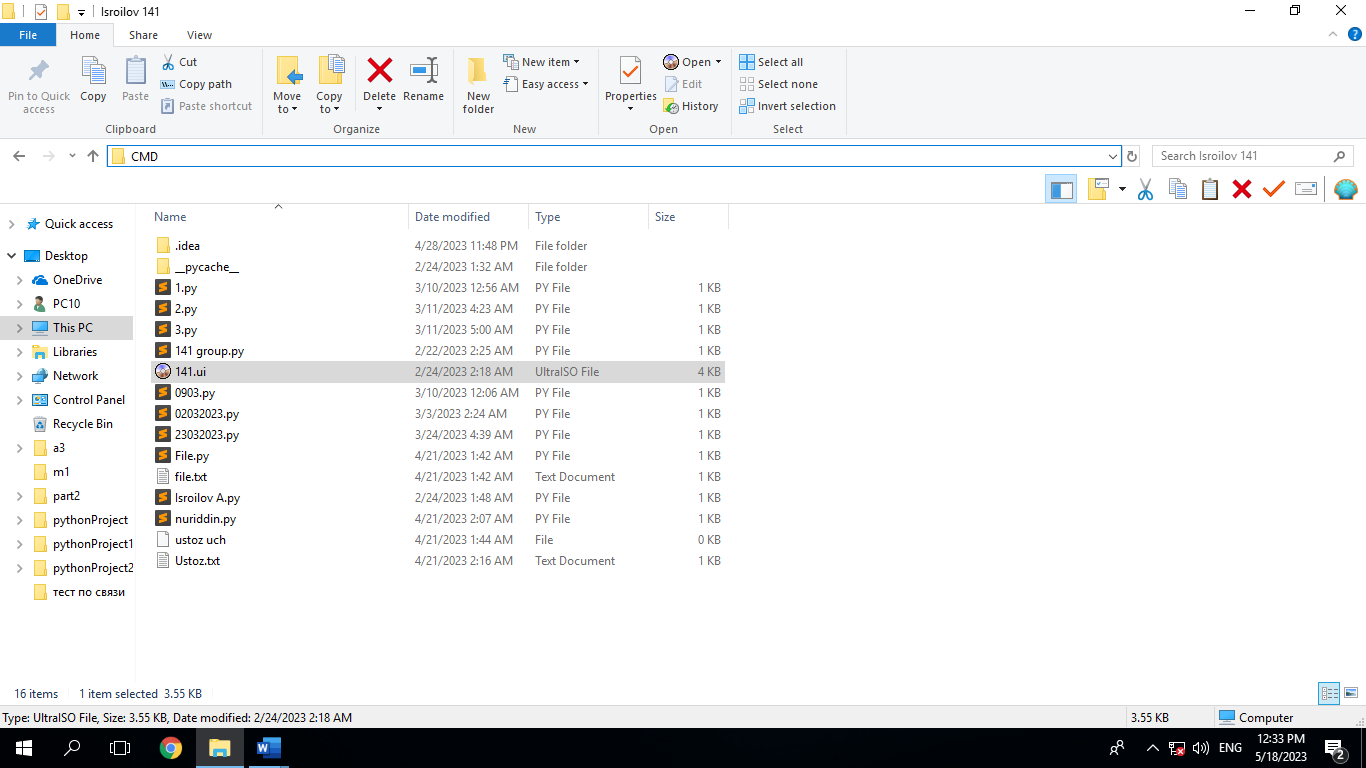
QtDesigner dasturidan yaratilgan, ilova ko‘rinishini o‘zida saqlovchi faylni ichida python kodlarini joylashtirish uchun ushbu faylni python faylga o‘girib olish talab qilinadi. Buning uchun quyidagi amallarni bajarish kerak bo‘ladi:

1. QtDesigner dasturida yaratilgan **.ui** formatli faylni topiladi;

2. Ushbu fayl saqlangan kotalogdan turib konsolni ochiladi. Buning uchun kotlaogning **URL** manzili yoziladigan joyga **CMD** deb yoziladi va **enter** tugmasini bosish orqali konsolni kerakli kotologda ochish amali bajariladi (39-, 40-rasmlar):



Papkaning URL manzili yoziladigan joyi



CMD buyrug‘i yozilishi kerak bo‘lgan joy.

3. Ochilgan konsol oynasiga quyidagi buyruqni yoziladi. Faqat ui\_file\_name yozuvi o‘rniga QtDesigner dasturining fayli nomi yozilsa, python\_file\_name o‘rniga hosil bo‘ladigan python fayl nomi yozilishi kerak bo‘ladi:

**pyuic5** –x "**ui\_file\_name**".**ui** –o "**python\_file\_name**".**py**

Shundan so‘ng ushbu kotalogda ko‘rsatilgan nomdagi .py formatdagi fayl ham paydo bo‘lganini ko‘rishingiz mumkin. Va ushbu python faylning tarkibiga turli python kodlarini qo‘shish, vidgetlarining qiymatlarini olish va ularni ishga tushirish uchun turli funksiyalar class, metodlar qo‘shish imkoniyati mavjud.

**Nazorat savollari :**

1.  Python PyQt 5 paketining xususiyatlarini tavsiflab bering .

2.  Windows muhiti uchun PyQt5 paketi qanday o‘rnatiladi?

3.  QtDesigner dasturini o‘rnatish qanday amalga oshiriladi?

4. QtDesigner dasturining .ui faylini .py formatli faylga o‘girish uchun qanday amallarni bajarish kerak bo‘ladi?

5. Nima uchun .ui formatli faylni python faylga og‘irish kerak?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.

**2-mashg‘ulot.** PyQt5 kutubxonasi. QLabel vidjeti.

**O‘quv savollari:**

1. QLabel vidjeti;
2. QLabel shrift, o‘lcham va text xususiyatlari;

**1. QLabel vidjeti**

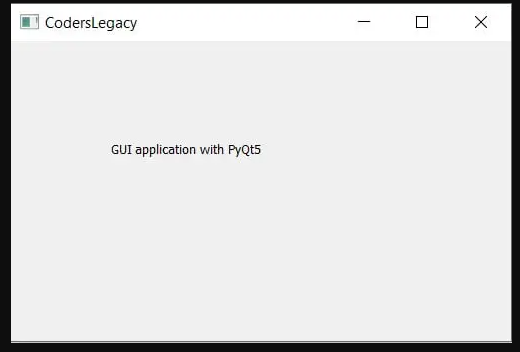
QLabel matn **satrlarini** ko‘rsatish qobiliyatiga ega PyQt-dagi eng oddiy vidjetlardan biridir. Ushbu vidjet xohlagan vaqtda ushbu matnni olish va yangilash imkonini beruvchi koʻplab yordamchi funksiyalar va usullar bilan birga keladi.

**QLabel** bilan bog‘liq barcha usullarning to‘liq ro‘yxatini ushbu sahifaning pastki qismida topish mumkin.

**QLabel yaratish.** Quyidagi misolda PyQt5 paketi yordamida oddiy QLabel yaratilishini ko‘rsatib o‘tilgan:

|  |
| --- |
| from PyQt5 import QtWidgets  from PyQt5.QtWidgets import QApplication, QMainWindow  import sys    app = QApplication(sys.argv)  win = QMainWindow()  win.setGeometry(400,400,500,300)  win.setWindowTitle("CodersLegacy")  label = QtWidgets.QLabel(win)  label.setText("GUI application with PyQt5")  label.adjustSize()  label.move(100,100)    win.show()  sys.exit(app.exec\_()) |

Ushbu kodning chiqishi quyida ko‘rsatilgan:



*42-rasm. PyQt5 QLabel*

QLabel bilan bog‘liq kodni satr bo‘yicha muhokama qilib chiqaylik.

|  |  |
| --- | --- |
|  | label = QtWidgets.QLabel(win) |

QtWidgets.QLabel() ga murojaat qilish GUI uchun foydalanish mumkin bo‘lgan yorliq ob'ektini yaratadi. Biz yaratgan (win) oyna ob'ektini uning parametrlari sifatida berilishi yorliq ushbu oynaning bir qismi ekanligini anglatadi.

|  |  |
| --- | --- |
|  | label.setText("GUI application with PyQt5") |

Yuqoridagi oddiy kod labelga matn o‘rnatadi.

|  |  |
| --- | --- |
|  | label.adjustSize() |

**adjustSize()** QLabel o‘z o‘lchamini avtomatik ravishda sozlashiga imkon beradi.

|  |  |
| --- | --- |
|  | label.move(100,100) |

Ushbu **move** **(100, 100)** labening dastur ekranining yuqori chap burchagidagi (100, 100) koordinatalarida paydo bo‘lishini ta’minlaydi. Yodda tutingki, **ekranning** yuqori chap burchagida (0, 0) pozitsiya mavjud.

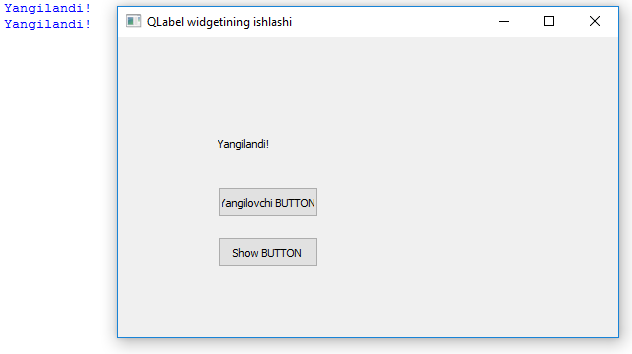
**2. QLabel shrift, o‘lcham va text xususiyatlari**

Ushbu bo‘limda hozirda QLabel da ko‘rsatilgan matnni qanday yangilashni, shuningdek QLabelda ko‘rsatilayotgan joriy matnni qanday qilib olish mumkinligini ko‘rib chiqamiz. Bunday ishlarni qilish uchun QPushButton vidjetidan foydalanish kerak, u kerakli buyruqlarni o‘z ichiga olgan funksiyalarni chaqiradi.

QLabelni yangilash uchun avvalgi **setText()** usulidan foydalaniladi va matn qiymatini olish uchun ushbu **text()** usulidan foydalaniladi:

|  |  |
| --- | --- |
|  | from PyQt5 import QtWidgets  from PyQt5.QtWidgets import QApplication, QMainWindow  import sys    def update():  matn.setText("Yangilandi!")    def retrieve():  print(matn.text())    app = QApplication(sys.argv)  win = QMainWindow()  win.setGeometry(400,400,500,300)  win.setWindowTitle("QLabel widgetining ishlashi")    matn = QtWidgets.QLabel(win)  matn.setText("PyQt5 paketida GUI dasturlari")  matn.adjustSize()  matn.move(100,100)    upd\_btn = QtWidgets.QPushButton(win)  upd\_btn.clicked.connect(update)  upd\_btn.setText("Yangilovchi BUTTON")  upd\_btn.move(100,150)    show\_btn = QtWidgets.QPushButton(win)  show\_btn.clicked.connect(retrieve)  show\_btn.setText("Show BUTTON")  show\_btn.move(100,200)    win.show()  sys.exit(app.exec\_()) |

Yuqoridagi kodni ishga tushirgandan so‘ng ikkita QPushButton va QLabel vidgetlari orqali ishlovchi ilova paydo bo‘ladi. Unda QLabelga “***PyQt5 paketida GUI dasturlari***” yozuvi, buttonlarning biriga “***Yangilovchi BUTTON***”, ikkinchisiga esa “***Show BUTTON***” yozuvlari yozilgan bo‘ladi. Ilovaning ishlashi quyidagicha: “***Yangilovchi BUTTON***” tugmachasi bosilganda labelning matnini “Yangilandi!” yozuviga o‘zgartiruvchi update funksiyasini ishga tushiradi. ***Show BUTTON*** tugmachasi bosish orqali esa “retrieve” funksiyasini ishga tushirish orqali QLabel ning text xususiyatidagi yozuvni konsolga chiqariladi.



*43-rasm. QLabel uchun yozilgan dasturning natijasi.*

**Label vidjeti uchun mavjud bo‘lgan eng muhim metodlar:**

jadval

| **Metod nomi** | **Tavsif** |
| --- | --- |
| .setAlignment() | Label ichidagi matnni tekislash. U 4 xil qiymatlardan birini olishi mumkin: *Qt.AlignLeft*, *Qt.AlignRight*, *Qt.AlignCenter* va Qt.AlignJustify*.* |
| .setIndent() | Label matni uchun “tab”larni belgilaydi. |
| .setPixmap() | Labelga qo‘yilgan rasmni ko‘rsatish uchun ishlatiladi. |
| .text() | Label uchun (ko‘rsatilgan) matn qiymatini qaytaradi. |
| .setText() | Labelda ko‘rsatiladigan matnni o‘rnatadi. |
| .selectedText() | Foydalanuvchi tomonidan tanlangan matnni qaytaradi. (Buning ishlashi uchun textInteractionFlag TextSelectableByMouse xuxusiyatlariga o‘rnatilishi kerak). |
| .setWordWrap() | Label matnini avtomatik ravishda keying qatorga tushadigan xususiyatini yoqish. |
| .adjustSize() | Label o‘lchamini ko‘rsatilayotgan matnga avtomatik moslashtiradi. |

**Nazorat savollari:**

1. QLabel vidjetidan qanday maqsadda foydalaniladi?
2. QLabel vidjetining adjustSize() metodining vazifasi qanday?
3. QLabel vidjetiningmove() metodining vazifasi qanday?
4. Label ichidagi matnni tekislash uchun qaysi metod ishlatiladi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish**.**

**3-mashg‘ulot.** PyQt5 kutubxonasi. QLineEdit vidjeti.

**O‘quv savollari:**

1. QLineEdit vidjeti;
2. setStyleSheet() metodi.

**1. QLineEdit vidjeti**

**QLineEdit** - bu matnning bir qatorini kiritish va tahrirlash imkonini beruvchi vidjet. Ushbu vidjetda Cencel va Repeat, Cut va Past hamda “Sudrab olib tashlash” funksiyalari mavjud.

**import** **sys**

**from** **PyQt5.QtWidgets** **import** (QWidget, QLabel,

    QLineEdit, QApplication)

**class** **Examples**(QWidget):

**def** \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.initUI()

**def** initUI(self):

        self.lbl = QLabel(self)

        qlineE = QLineEdit(self)

        qlineE.move(100, 100)

        self.lbl.move(100, 40)

        qlineE.textChanged[str].connect(self.changed)

        self.setGeometry(300, 300, 280, 170)

        self.setWindowTitle(‘QlineEdit Example’)

        self.show()

**def** changed(self, text):

         self.lbl.setText(Label text)

        self.lbl.adjustSize()

**if** \_\_name\_\_ == ‘\_\_main\_\_’:

    app = QApplication(sys.argv)

    ex = Examples()

    sys.exit(app.exec\_())

Ushbu misolda QLineEdit vidjeti va QLabel ko‘rsatilgan. Tahrirlash paneliga kiritgan matnimiz darhol teg vidjetida ko‘rsatiladi.

qlineE = **QLineEdit**(**self**)

* QLineEdit vidjetining qline obyekti yaratildi.

qlineE.**textChanged**[str].**connect**(**self**.**changed**)

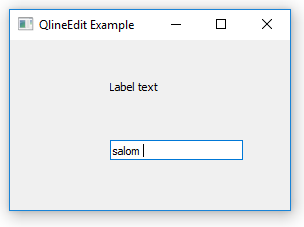
* Agar QLineEdit vidjetidagi matn o‘zgarsa, changed() funksiyasi ishga tushadi.

def **changed**(**self**, text):

**self**.lbl.**setText**(text Label)

**self**.lbl.**adjustSize**()

* changed funksiyasining ichida kiritilgan matnni teg vidjetiga o‘rnatish buyrug‘i joylashtirilgan. Yorliq o‘lchamini matn uzunligiga mos ravishda o‘zgartirish uchun adjustSize() metodidan foydalanilgan .



Yuqoridagi dastur ko‘rinishi

**PyQt QLineEdit**

PyQt QLineEdit bir qatorli matnli vidjet yaratish imkonini beradi. Odatda, siz QLineEditma'lumotlarni kiritish shaklida foydalanasiz .

Amalda siz ko‘pincha QLineEditvidjetni vidjet bilan ishlatasiz QLabel.

Vidjet yaratish uchun QLineEditquyidagi amallarni bajaring.

Birinchidan, moduldan import QLineEditqiling PyQt6.QtWidgets:

from PyQt6.QtWidgets import QLineEditKod tili: Python ( python )

Ikkinchidan, QLineEditfoydalanadigan yangi ob'ekt yarating:

* Hech qanday dalil yo‘q.
* Faqat ota-ona vidjeti bilan.
* Yoki birinchi argument sifatida standart satr qiymati bilan.

Masalan:

line\_edit = QLineEdit('Default Value', self)Kod tili: Python ( python )

Bundan tashqari, siz quyidagi qo‘shimcha xususiyatlardan foydalanishingiz mumkin:

| **Metod** | **Turi** | **Ta’rifi** |
| --- | --- | --- |
| matn | ip | Satrni tahrirlash mazmuni |
| readOnly | Mantiqiy | To‘g‘ri yoki noto‘g‘ri. Agar rost boʻlsa, qatorni tahrir qilib boʻlmaydi |
| clearButtonEnabled | Mantiqiy | Aniq tugmani qo‘shish to‘g‘ri |
| placeholderText | ip | Satrni tahrirlash bo‘sh bo‘lganda paydo bo‘ladigan matn |
| maxLength | butun son | Maksimal kiritilishi mumkin bo‘lgan belgilar sonini belgilang |
| echoMode | QLineEdit.EchoMode | Matnni ko‘rsatish usulini o‘zgartiring, masalan, parol |

PyQt QLineEdit vidjetiga misollar

Keling, vidjetdan foydalanishga misollar keltiraylik QLineEdit.

1) Oddiy PyQt QLineEdit misoli

Quyidagi dastur vidjetni qanday yaratishni ko‘rsatadi QLineEdit:

import sys

from PyQt6.QtWidgets import (

QApplication,

QWidget,

QLineEdit,

QVBoxLayout

)

class MainWindow(QWidget):

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.setWindowTitle('PyQt QLineEdit Widget')

self.setGeometry(100, 100, 320, 210)

search\_box = QLineEdit(

self,

placeholderText='Enter a keyword to search...',

clearButtonEnabled=True

)

*# place the widget on the window*

layout = QVBoxLayout()

layout.addWidget(search\_box)

self.setLayout(layout)

*# show the window*

self.show()

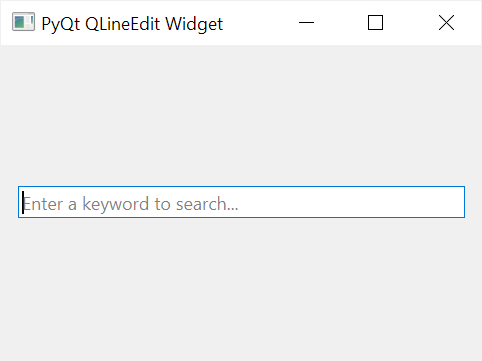
if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

window = MainWindow()

sys.exit(app.exec())Kod tili: Python ( python )

Chiqish:



2) Parol yozuvini yaratish uchun PyQt QLineEdit dan foydalanish

Quyidagi dastur QLineEditparol kiritish sifatida yangi vidjet yaratadi:

import sys

from PyQt6.QtWidgets import (

QApplication,

QWidget,

QLineEdit,

QVBoxLayout

)

class MainWindow(QWidget):

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.setWindowTitle('PyQt QLineEdit Widget')

self.setGeometry(100, 100, 320, 210)

password = QLineEdit(self, echoMode=QLineEdit.EchoMode.Password)

*# place the widget on the window*

layout = QVBoxLayout()

layout.addWidget(password)

self.setLayout(layout)

*# show the window*

self.show()

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

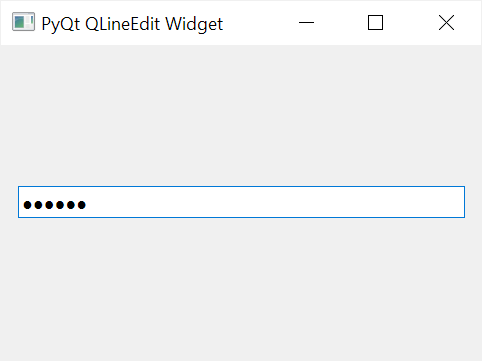
window = MainWindow()

sys.exit(app.exec())Kod tili: Python ( python )

QLineEditVidjetni parol kiritishiga aylantirish uchun siz echoModeni o‘rnatasiz QLineEdit. EchoMode.Password:

password = QLineEdit(self, echoMode=QLineEdit.EchoMode.Password)Kod tili: Python ( python )

Chiqish:



3) PyQt QLineEdit-dan avtomatik to‘ldirish xususiyati bilan foydalanish

Avtomatik to‘ldirish xususiyati bilan yozuv yaratish uchun siz quyidagi amallarni bajaring:

Birinchidan, modulni import QCompleterqiling PyQt6.QtWidgets.

Ikkinchidan, QCompleteravtomatik toʻldirish funksiyasi uchun ishlatiladigan satrlar roʻyxati bilan vidjet yarating:

completer = QCompleter(word\_list)Kod tili: Python ( python )

Uchinchidan, a yarating va to‘ldiruvchi ob'ekt bilan QLineEdituning usulini chaqiring :setCompleter()

line\_edit = QLineEdit(self)

line\_edit.setCompleter(completer)Kod tili: Python ( python )

Masalan, quyidagi dastur QLineEditavtomatik to‘ldirish xususiyatiga ega vidjetni ko‘rsatadi:

import sys

from PyQt6.QtWidgets import (

QApplication,

QWidget,

QLineEdit,

QVBoxLayout,

QCompleter

)

class MainWindow(QWidget):

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.setWindowTitle('PyQt QLineEdit Widget')

self.setGeometry(100, 100, 320, 210)

common\_fruits = QCompleter([

'Apple',

'Apricot',

'Banana',

'Carambola',

'Olive',

'Oranges',

'Papaya',

'Peach',

'Pineapple',

'Pomegranate',

'Rambutan',

'Ramphal',

'Raspberries',

'Rose apple',

'Starfruit',

'Strawberries',

'Water apple',

])

fruit = QLineEdit(self)

fruit.setCompleter(common\_fruits)

*# place the widget on the window*

layout = QVBoxLayout()

layout.addWidget(fruit)

self.setLayout(layout)

*# show the window*

self.show()

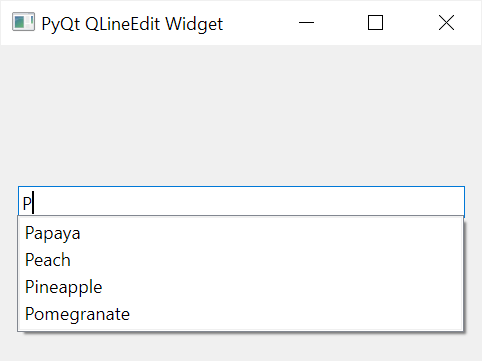
if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

window = MainWindow()

sys.exit(app.exec())Kod tili: Python ( python )

Chiqish:



Xulosa

* QLineEditBir qatorli kirish vidjetini yaratish uchun foydalaning .
* echoModeMatnni ko‘rsatish usulini o‘zgartirish uchun xususiyatdan foydalaning .
* QLineEditAvtomatik to‘ldirish funksiyasini qo‘llab-quvvatlash uchun vidjetdan QCompleter vidjeti bilan foydalaning .

**Nazorat savollari:**

1. PyQt da yorliq yaratish uchun qaysi vidgetdan foydalaniladi?
2. QLineEdit vidjetining vazifasi qanday?
3. QLineEdit vidjetining xususiyatlari qanday?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.

**4-mashg‘ulot.** PyQt5 modal dialog. QMessageBox vidgeti bilan ishlash.

**O‘quv savollari:**

1. QMessageBox vidgetining vazifasi.
2. QMessageBox vidgetining asosiy xususiyatlari.
3. Statik funksiyalari.
4. Piktogramma va Pixmap xususiyatlari.

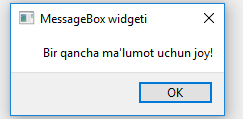
**1. QMessageBox vidgetining vazifasi**

QMessageBox - bu dialoglar yaratish uchun ishlatiladigan foydali PyQt5 vidjeti. Ushbu dialog oynalari turli maqsadlarda ishlatilishi va turli xil xabarlar va tugmalarni ko‘rsatish uchun ko‘p jihatdan moslashtirilgan bo‘lishi mumkin.

**Quyida oddiy QMessageBox** vidjetini yaratmoqda. Ammo bu hech qanday qo‘shimcha funksiyalar yoki sozlashlarsiz faqat MessageBox vidjetining o‘zining tinchlik xolatidagi ko‘rinishi bo‘ladi.

|  |  |
| --- | --- |
|  | def display():  msg = QMessageBox(wind)  msg.setWindowTitle("MessageBox widgeti")  msg.setText("Bir qancha ma’lumot uchun joy!")    x = msg.exec\_() |

Va quyida chiqish. Agar siz ekranda oddiy xabarni ko‘rsatmoqchi bo‘lsangiz, buni qilishning yo‘li.



*61-rasm. QmessageBox vidjetining o‘zining ko‘rinishi*

**2. QMessageBox vidgetining asosiy xususiyatlari**

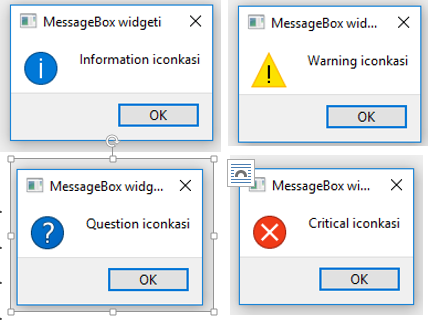
**QMessageBox –** da .**setIcon()** metodini chaqirish orqali foydalanish mumkin bo‘lgan 4 xil turdagi piktogramma mavjud bo‘lib, ko‘rsatmoqchi bo‘lgan xabar turiga qarab qaysi birini ishlatishni hal qilinadi.

* QMessageBox.Question
* QMessageBox.Information
* QMessageBox.Warning
* QMessageBox.Critical

Yuqoridagilardan birini quyida ko‘rsatilganidek **setIcon()** funksiyasiga o‘tkazish kifoya .

|  |  |
| --- | --- |
|  | msg.setIcon(QMessageBox.Question) |

Quyida ko‘rsatilishi mumkin bo‘lgan to‘rt xil xabarlar mavjud.



*62-rasm. QMessageBox pictogrammalarining turlari*

**QMessageBox tugmalari**

**Hozirgacha biz QMessageBox taqdim etgan standart “OK**” tugmasidan foydalanib kelmoqdamiz. Biroq, QMessageBox foydalanish uchun o‘ndan ortiq turli xil tugmalar taklif qiladi. Quyida ba’zi tugmalardan foydalanish misoli keltirilgan.

**setStandardButtons**() funktsiyadan foydalanib xohlagan tugma turini o‘ratish mumkin.

|  |  |
| --- | --- |
|  | def display():  msg = QMessageBox(wind)  msg.setWindowTitle("Message Box Sarlavhasi")  msg.setText("Bir qancha ma’lumot uchun joy!")  msg.setIcon(QMessageBox.Question)  msg.setStandardButtons(QMessageBox.Cancel  |QMessageBox.Ok  |QMessageBox.Retry)  msg.setInformativeText("Ush bu havolani yopasizmi?")  x = msg.exec\_() |

Ushbu dasturning natijasi quyida keltirilgan:



*63-rasm. QMessageBox ga doir oddiy dastur natijasi*

Buni allaqachon payqagan bo‘lishingiz mumkin, lekin tugmalar tartibi ularning xabarlar oynasida qanday ko‘rinishiga ta’sir qilmaydi.

**Default Button**

Yuqoridagi rasmga qarab “OK” tugmasi atrofida ko‘k kontur borligini payqash mumkin . Bu ushbu tugma **Default Button** ekanligini bildiradi.

**setDefaultButton**() funktsiyasidan foydalanib, standart tugmani o‘zgartirish mumkin. Kodga quyidagi qatorni qo‘shish ko‘k rangni "**Cancel**" tugmasi atrofida bo‘lishiga olib keladi.

|  |  |
| --- | --- |
|  | msg.setDefaultButton(QMessageBox.Cancel) |

Bu yerda **QmessageBox** vidjetida foydalanish mumkin bo‘lgan turli xil tugma turlarining to‘liq ro‘yxati keltirilgan:

* QMessageBox.Ok
* QMessageBox.No
* QMessageBox.Yes
* QMessageBox.Cancel
* QMessageBox.Close
* QMessageBox.Abort
* QMessageBox.open
* QMessageBox.Retry
* QMessageBox.Ignore
* QMessageBox.Save
* QMessageBox.Retry
* QMessageBox.Apply
* QMessageBox.Help
* QMessageBox.Reset
* QMessageBox.SaveAll
* QMessageBox.YesToAll
* QMessageBox.NoToAll

**3. Statik funksiyalari**

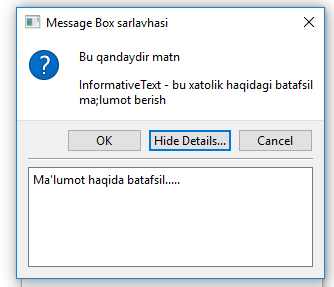
Odatda QMessageBox-da Matnni ko‘rsatadigan faqat bitta maydon mavjud. Aslida esa, qo‘shimcha matn qo‘shish uchun yana ikkita qo‘shimcha maydon mavjud.

Birinchisi, QMessageBox oynasining o‘zida qo‘shimcha matn bo‘limi. Bu qo‘shish mumkin bo‘lgan qo‘shimcha matn qatoriga o‘xshaydi. Ushbu yangi matn maydonini qo‘shish uchun faqat setInformativeText() funksiyasiga murojaat qilish kerak.

Ikkinchisi QMessageBox-dan kengaytirilgan maydonda ko‘rsatiladi. Bu **Detailed Text** – ya’ni “Batafsil matn” deb ataladi. Ushbu bo‘limni o‘rnatish avtomatik ravishda ushbu maydonni ko‘rsatish uchun ishlatiladigan tugmani qo‘shadi. Ushbu batafsil matn yoziladigan maydonni qo‘shish uchun faqat **setDetailedText()** funksiya kerak bo‘ladi.

|  |  |
| --- | --- |
|  | from PyQt5 import QtWidgets  from PyQt5.QtWidgets import \*  import sys    def show\_popup():  msg = QMessageBox(win)  msg.setWindowTitle("Message Box sarlavhasi")  msg.setText("Bu qandaydir matn")  msg.setIcon(QMessageBox.Question)    msg.setStandardButtons(QMessageBox.Cancel|QMessageBox.Ok)  msg.setDefaultButton(QMessageBox.Ok)    msg.setDetailedText("Ma’lumot haqida batafsil.....")  msg.setInformativeText("InformativeText - bu xatolik haqidagi batafsil ma;lumot berish")  x = msg.exec\_()    app = QApplication(sys.argv)  win = QMainWindow()  win.setGeometry(400,400,300,300)  win.setWindowTitle("QMessageBox widgeti haqida")    button = QtWidgets.QPushButton(win)  button.setText("A Button")  button.clicked.connect(show\_popup)  button.move(100,100)    win.show()  sys.exit(app.exec\_()) |

Yuqoridagi dasturning ishga tushirilganidan keyin quyidagi messagebox oynasi paydo bo‘ladi:



*QMessageBox widgetining to‘liq ko‘rinishi*

Hide Details/Show Details tugmasini bosish mos ravishda xabarlar qutisi ostidagi qo‘shimcha maydonni yashirish/ko‘rsatish mumkin.

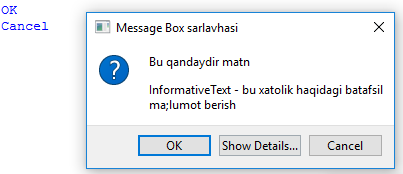
**QMessageBox qiymatlarini olish**

Yuqorida juda ko‘p turli xil sozlashlar va xususiyatlarni muhokama qilindi, ammo aslida bu turli xususiyat va tugmalarning hech birini kodga bog‘lanmadi.

Misol uchun, agar MessageBoxda 3 xil tugma bo‘lsa, qaysi biri bosilganligini qanday bilish mumkin?  Buning uchun quyida ko‘rsatilgan kodyaqqol javob bo‘la oladi:

|  |  |
| --- | --- |
|  | from PyQt5 import QtWidgets  from PyQt5.QtWidgets import \*  import sys    def show\_popup():  msg = QMessageBox(win)  msg.setWindowTitle("Message Box sarlavhasi")  msg.setText("Bu qandaydir matn")  msg.setIcon(QMessageBox.Question)  msg.setStandardButtons(QMessageBox.Cancel|QMessageBox.Ok)  msg.setDefaultButton(QMessageBox.Ok)    msg.setDetailedText("Ma’lumot haqida batafsil.....")  msg.setInformativeText("InformativeText - bu xatolik haqidagi batafsil ma;lumot berish")  msg.buttonClicked.connect(popup)  x = msg.exec\_()    def popup(i):  print(i.text())    app = QApplication(sys.argv)  win = QMainWindow()  win.setGeometry(400,400,300,300)  win.setWindowTitle("QMessageBox widgeti haqida")    button = QtWidgets.QPushButton(win)  button.setText("A Button")  button.clicked.connect(show\_popup)  button.move(100,100)    win.show()  sys.exit(app.exec\_()) |

**buttonClicked.connect()** usulidan foydalanib  MessageBoxdagi birorta tugma bosilganda funktsiyani ishga tushirish mumkin. Bunday holda, messageboxni **popup** deb nomlangan funktsiyaga bog‘landi. popup() funsiyas bitta parametr qabul qiladi.



*QMessageBox vidjetining to‘liq imkoniyatlari bilan ko‘rinishi*

MessageBoxning tugmasi bosilganda, u avtomatik ravishda bosilgan tugmani **buttonClicked.connect()** da e’lon qilingan funksiyaga uzatadi. Ushbu tugmadagi text() funktsiyasini chaqirish esa tugmaning matnini qaytaradi.

**Nazorat savollari:**

1. QMessageBox vidjeti qanday vazifani bajaradi?
2. QMessageBox qanday ishga tushiriladi?
3. QMessageBox ning qanday xususiyatlari mavjud?
4. QMessageBox vidgetida qanday pictogrammalar mavjud?
5. QmessageBoxga o‘rnatish mumkin bo‘lhan standart buttonlarni keltiring?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.

**5-mashg‘ulot.** PyQt da rasmlar va menyular.

O‘quv savollari:

1. PyQt paketi yordamida rasm joylashtirish usullari.
2. Menu yaratish. Menu vidgetining xususiyatlari.

**1. PyQt paketi yordamida rasm joylashtirish usullari**

**QPixMap vidjeti**

**QPixMap –** butasvirlarni ekranga chiqarish uchun ishlatiladigan vidjetlardan biri. Quyidagi misolda tasvirni ekranda ko‘rsatish uchun **QPixMap** metodidan foydalanilgan.

import sys

from PyQt5.QtWidgets import \*

from PyQt5.QtGui import QPixmap

class Example(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.initUI()

def initUI(self):

h\_box = QHBoxLayout(self)

pixmap = QPixmap("blue\_black.png")

lbl = QLabel(self)

lbl.setPixmap(pixmap)

h\_box.addWidget(lbl)

self.setLayout(h\_box)

self.move(300, 200)

self.setWindowTitle('Blue and Black')

self.show()

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

ex = Example()

sys.exit(app.exec\_())

Ushbu dasturda blue\_black.png nomli rasmni oynada ko‘rsatish uchun quyidagi amallarni bajarildi:

1. pixmap = **QPixmap**("blue\_black.png") - **QPixMap** sinfi ob’yekti yaratildi.
2. lbl = **QLabel**(self)

lbl.**setPixmap**(pixmap) - rasmni QLabel vidjetidan foydalanib ekranga joylashtirildi.



*71-rasm. QPixMap vidjetiga rasm joylashtirilgandagi ko‘rinishi*

**2. Menu yaratish. Menu vidgetining xususiyatlari.**

PyQt5 ga bag‘ishlangan bobning ushbu qismida menyu va asboblar paneli (asboblar paneli) vidgetlaridan foydalanishni ko‘rib chiqiladi.

Menyu - bu menyu satrida joylashgan buyruqlar guruhi. Asboblar panelida ilovadagi ba’zi umumiy buyruqlar uchun tugmalar mavjud.

**Menyu paneli**

Menyu paneli GUI ilovasining umumiy qismidir. Bu turli xil menyularda joylashgan buyruqlar guruhi.

import sys

from PyQt5.QtWidgets import \*

from PyQt5.QtGui import QIcon

class Example(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.initUI()

def initUI(self):

exitAction = QAction(QIcon(‘exit.png‘), ‘&Exit’, self)

exitAction.setShortcut(‘Ctrl+Q’)

exitAction.setStatusTip(‘Exit application’)

exitAction.triggered.connect(qApp.quit)

self.statusBar()

menubar = self.menuBar()

fileMenu = menubar.addMenu(‘&File’)

fileMenu.addAction(exitAction)

self.setGeometry(300, 300, 300, 200)

self.setWindowTitle(‘Menubar’)

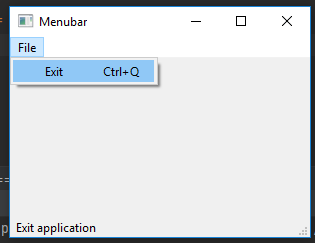
self.show()

if \_\_name\_\_ == ‘\_\_main\_\_’:

app = QApplication(sys.argv)

ex = Example()

sys.exit(app.exec\_())



Menubar widgetining ilovada ko‘rinishi.

Yuqoridagi misolda bitta  menubar vidgeti yordamida menyu panelini yaratildi. Ushbu menyu tarkibida dasturni to‘xtatadigan bitta buyruq kiritilgan.  Ushbu buyruqni menular panelidan sichqoncha bilan tanlash yoki klaviaturadan Ctrl + q tugmalari yordamida ishga tushirish ham mumkin.

**Datur tahlili:**

exitAction = QAction(QIcon(‘exit.png‘), ‘&Exit’, self)

exitAction.setShortcut(‘Ctrl+Q’)

exitAction.setStatusTip(‘Exit application’)

* Ushbu uchta qatorda tegishli belgi bilan hodisa yaratilgan. Bundan tashqari, ushbu hodisa uchun tugmalar birikmasi aniqlanadi. Uchinchi qator kursorni menyu bandi ustiga olib borilganda satr yonida maslahatchiga o‘xshagan yozuv yaratadi.

exitAction.triggered.connect(qApp.quit)

* Ushbu maxsus harakatni tanlaganimizda, signal ishga tushadi. Signal **QApplication** vidjetining **quit()** metodini ishga tushiradi va ilovani yakunlaydi.

menubar = self.menuBar()

fileMenu = menubar.addMenu(‘&File’)

fileMenu.addAction(exitAction)

* Bu yerga **menyuBar()** vidgeti yordamida bitta menular satri yaratilyapdi.  fileMenu o‘zgaruvchisiga menular satrifa File nomli menu qo‘shish buyrug‘i yuklanmoqda. Va uchinchi satrda Fayl menyusining tarkibida dasturdan chiqish amalini qo‘shilmoqda.

**Asboblar paneli (toolbar)**

Menyular ilovada foydalanishimiz mumkin bo‘lgan barcha buyruqlarni guruhlaydi.

Asboblar paneli eng ko‘p ishlatiladigan buyruqlarga tezkor kirishni ta’minlaydi.

import sys

from PyQt5.QtWidgets import \*

from PyQt5.QtGui import QIcon

class Example(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.initUI()

def initUI(self):

exitAction = QAction(QIcon(‘exit24.png‘), ‘Exit’, self)

exitAction.setShortcut(‘Ctrl+Q’)

exitAction.triggered.connect(qApp.quit)

self.toolbar = self.addToolBar(‘Exit’)

self.toolbar.addAction(exitAction)

self.setGeometry(300, 300, 300, 200)

self.setWindowTitle(‘Toolbar’)

self.show()

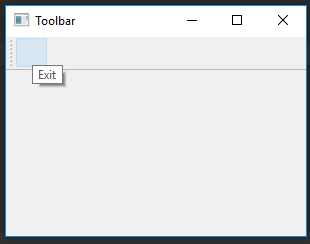
if \_\_name\_\_ == ‘\_\_main\_\_’:

app = QApplication(sys.argv)

ex = Example()

sys.exit(app.exec\_())

Bu yerda deyarli hamma narsa status paneliga o‘xshaydi.



Asboblar panelining ilovada ko‘rinishi.

**Hammasini birlashtirish**

Ushbu bo‘limning oxirgi misolida menyu satri, asboblar paneli va holat panelini yaratildi. Shuningdek, markaziy vidjetni ham ular bilan birlashtirilgan.

import sys

from PyQt5.QtWidgets import \*

from PyQt5.QtGui import QIcon

class Example(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.initUI()

def initUI(self):

textEdit = QTextEdit()

self.setCentralWidget(textEdit)

exitAction = QAction(QIcon(‘exit24.png‘), ‘Exit’, self)

exitAction.setShortcut(‘Ctrl+Q’)

exitAction.setStatusTip(‘Exit application’)

exitAction.triggered.connect(self.close)

self.statusBar()

menubar = self.menuBar()

fileMenu = menubar.addMenu(‘&File’)

fileMenu.addAction(exitAction)

toolbar = self.addToolBar(‘Exit’)

toolbar.addAction(exitAction)

self.setGeometry(300, 300, 350, 250)

self.setWindowTitle(‘Main window’)

self.show()

if \_\_name\_\_ == ‘\_\_main\_\_’:

app = QApplication(sys.argv)

ex = Example()

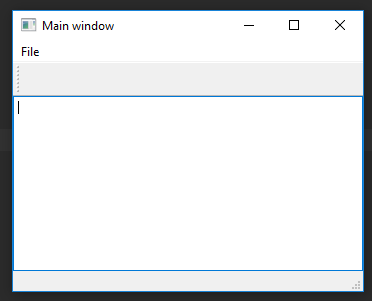
sys.exit(app.exec\_())

Ushbu kod misoli menyu, asboblar paneli va holat paneli bilan klassik GUI ilovasining skeletini yaratadi.

textEdit = QTextEdit()

self.setCentralWidget(textEdit)

Bu yerda matnni tahrirlash vidjetini yaratilmoqda. Uni **QMainWindow** ning markaziy vidjetiga aylantiriladi. Markaziy vidjet qolgan barcha bo‘sh joyni egallaydi.



Menu va uning asboblar paneli ko‘rinishi.

**Nazorat savollari:**

1. Menuni hosil qilishi uchun qanday widgetdan foydalaniladi?
2. Statusbar widgeti nima vazifani bajaradi?
3. Menular panelini qaysi widgetdan foydalanib o‘rnatiladi?
4. QPixMap vidjetining imkoniyatlaridan qanday foydalaniladi?
5. QPixMap vidjetining qanday xususiyatlari mavjud?
6. QPixMap vidjeti yordamida rasm joylashtirish qanday amalga oshiriladi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.

**6-mashg‘ulot.** Matn muharriri dizaynini yaratish.

O‘quv savollari:

1. Yaratiladigan matn muharriri uchun kerakli uskunalarni tanlash.
2. Tanlangan uskunalarni matn muharriri ekraniga joylashtirish.
3. Matn muharririr ekrani dizaynini shakllantirish.

**1. Yaratiladigan matn muharriri uchun kerakli uskunalarni tanlash**

Ushbu maqola Python -da grafik interfeys (GUI) bilan ilovalar yaratish bilan tanishishni endi boshlayotganlar uchun mo‘ljallangan . Unda biz PyQt-dan Qt Designer bilan birgalikda foydalanish asoslarini ko‘rib chiqamiz . Bosqichma-bosqich biz tanlangan katalog tarkibini ko‘rsatadigan oddiy Python GUI ilovasini yaratamiz .

**Bizga nimalar kerak bo‘ladi?**

Bizga PyQt va Qt Designer va Python kerak bo‘ladi , albatta.

Ushbu dasturni tuzishda PyQt5 va Python 3 dan foydalanadi, ammo PyQt va PySide yoki ularning Python 2 versiyalari o‘rtasida katta farqlar yo‘q .

**Birinchi navbatda Windowsda** PyQt paketini va Qt Designer dasturini yuklab olish hamda o‘rnatish talab qilinadi. QtDesigner dasturi PyQt paketi bilan birgalikda o‘rnatiladi.

Qt Designerni o‘z ichiga olgan Qt komponentlari va vositalarining aksariyati bilan paketni kerakli sayutlardan yuklab olishingiz  mumkin .

**Linux :** Sizga kerak bo‘lgan hamma narsa, ehtimol, tarqatish omborlarida. Qt Designer ilova markazidan o‘rnatilishi mumkin, lekin PyQt terminal orqali o‘rnatilishi kerak. Bizga kerak bo‘lgan hamma narsani bitta buyruq bilan o‘rnatishingiz mumkin, masalan:

# Fedora:

$ sudo dnf install python3-qt5 qt-creator

# Debian/Ubuntu:

$ sudo apt install python3-qt5 pyqt5-dev-tools qtcreator

Tayyorgarlik ishlarini tugatganingizdan so‘ng, buyruq satrini/terminalini oching va pyuic5 buyrug‘idan foydalanishingiz mumkinligiga ishonch hosil qiling. Siz quyidagilarni ko‘rishingiz kerak :

$ pyuic5

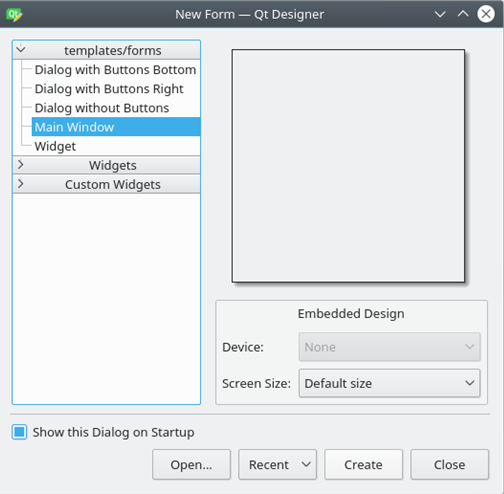
Error: one input ui-file must be specified

Agar siz bunday buyruq yoki shunga o‘xshash narsa yo‘qligi haqida xabarni ko‘rsangiz, operatsion tizimingiz va PyQt versiyasi uchun yechimni qidirib ko‘ring .

**Dastur Dizaynini yaratish**

Endi bizda hamma narsa tayyor, oddiy dizayndan boshlaylik.

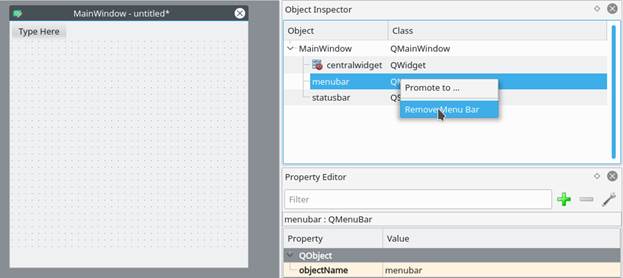
**Qt Designer** dasturini ishga tushuriladi, yangi ochilgan oynada kichik diolog oynasi paydo bo‘ladi. U yerdan **“Main Window”**  bo‘limi tanlab,   **Create**tugmasini bosing.



Shundan so‘ng sizda **forma**  - oyna dizayn uchun shablon paydo bo‘lishi kerak, uning o‘lchamini o‘zgartirish mumkin va vidjet oynasidan ob’ektlarni qayerga hohlasangiz qo‘shishingiz mumkin bo‘ladi. QtDesigner dasturi interfeysi bilan tanishib chiqing, bu juda ham oddiy tuzulishga ega.

Endi asosiy oynamiz hajmini biroz o‘zgartiramiz. Bizga bunchalik katta bo‘lishi shart emas. Va keling, avtomatik qo‘shilgan menyu va holat panelini ham olib tashlaymiz, chunki ular bizning ilovamizda foydali bo‘lmaydi.

Barcha shakl elementlari va ularning ierarxiyasi sukut bo‘yicha Qt Designer oynasining o‘ng tomonidagi “*Object Inspector”* deb nomlangan oynada ko‘rsatiladi. Ushbu oynada ob’ektlarni sichqonchaning o‘ng tugmasi bilan osongina o‘chirishingiz mumkin. Yoki ularni asosiy shaklda tanlashingiz va   klaviaturangizdagi *DEL* tugmasini bosishingiz mumkin.



Natijada, bizda deyarli bo‘sh shakl mavjud. Qolgan yagona ob’ekt -  ***centralwidget***, lekin bizga kerak bo‘ladi, shuning uchun biz u bilan hech narsa qilmaymiz.

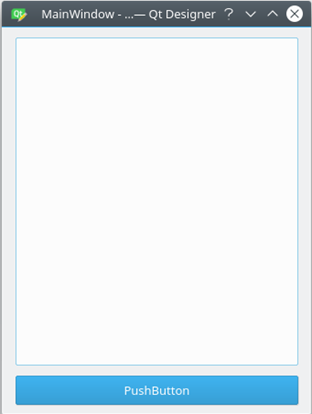
**2. Tanlangan uskunalarni matn muharriri ekraniga joylashtirish**

Endi*WidgetBox* dan *ListWidget*  ni  (*ListView* emas) va *PushButton* larni asosiy shaklning biror joyiga joylashtiring.

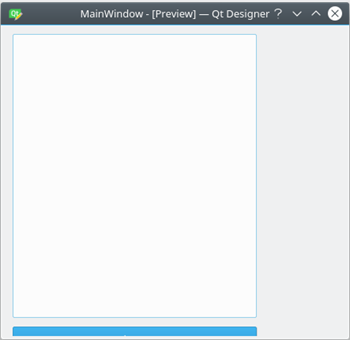
**Maketlar.** Ilovangizdagi elementlar uchun qatʼiy pozitsiyalar va oʻlchamlardan foydalanish oʻrniga, maketlardan foydalangan maʼqul. Ruxsat etilgan pozitsiyalar va o‘lchamlar sizga yaxshi ko‘rinadi (oyna o‘lchamini o‘zgartirmaguningizcha), lekin boshqa mashinalar va/yoki operatsion tizimlarda aynan bir xil bo‘lishiga hech qachon ishonch hosil qila olmaysiz.

**Maketlar** - bu vidjetlar uchun konteynerlar bo‘lib, ularni boshqa elementlarga nisbatan o‘z o‘rnida ushlab turadi. Shuning uchun, oyna hajmi o‘zgartirilganda, vidjetlarning o‘lchami ham o‘zgaradi.

Keling, maketlardan foydalanmasdan birinchi shaklimizni yarataylik. Shakldagi roʻyxat va tugmani torting va ularning oʻlchamlarini quyidagicha koʻrinishda oʻzgartiring:

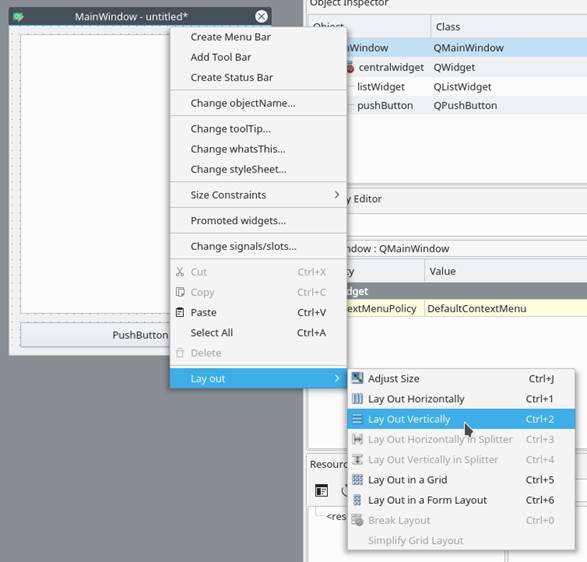


Endi Qt Designer menyusida *Formni* bosing, so‘ng  *Preview-ni* tanlang  va yuqoridagi skrinshotga o‘xshash narsani ko‘rasiz. Yaxshi ko‘rinadi, shunday emasmi? Ammo oyna o‘lchamini o‘zgartirganda nima sodir bo‘ladi:



Asosiy oynaning o‘lchami o‘zgargan va tugma deyarli ko‘rinmas bo‘lsada, bizning ob’ektlarimiz bir xil joylarda qoldi va o‘z o‘lchamlarini saqlab qoldi. Shuning uchun ko‘p hollarda maketlardan foydalanishga arziydi. Albatta, masalan, ob’ekt uchun qattiq yoki minimal/maksimal kenglik kerak bo‘lgan paytlar mavjud. Ammo umuman olganda, dasturni ishlab chiqishda maketlardan foydalanish yaxshiroqdir.

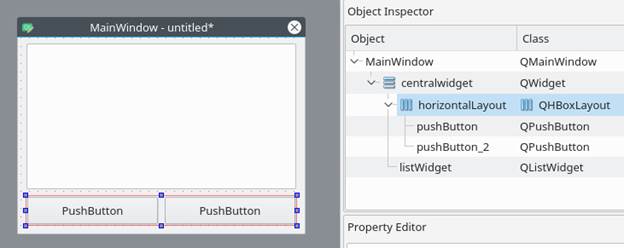
Asosiy oyna allaqachon maketlarni qo‘llab-quvvatlaydi, shuning uchun formamizga hech narsa qo‘shishimiz shart emas. *Ob’ekt*  inspektoridagi ***Main Window***ning o‘ng tugmasini bosing   va  ***Lay Out*  →  *Lay out vertically***ni tanlang. Shuningdek, siz formadagi bo‘sh joyni sichqonchaning o‘ng tugmasi bilan bosishingiz va bir xil variantlarni tanlashingiz mumkin:



**3. Matn muharririr ekrani dizaynini shakllantirish**

Sizning elementlaringiz o‘zgarishlar kiritilishidan oldingi tartibda bo‘lishi kerak, ammo agar ular bo‘lmasa, ularni kerakli joyga sudrab olib boring.

Biz vertikal joylashtirishdan foydalanganimiz sababli, biz qo‘shgan barcha elementlar vertikal ravishda joylashtiriladi. Istalgan natijani olish uchun joylashtirishlarni birlashtira olasiz. Masalan, ikkita tugmani gorizontal ravishda vertikalga qo‘yish quyidagicha ko‘rinadi:



Agar siz asosiy oynada elementni ko‘chira olmasangiz, buni  *Object Inspector* oynasida qilishingiz mumkin .

**Yakuniy urinishlar**

Endi vertikal joylashtirish tufayli bizning elementlarimiz to‘g‘ri hizalanadi. Qolgan yagona narsa (lekin shart emas) elementlarning nomini va ularning matnini o‘zgartirishdir.

Roʻyxat va tugma boʻlgan bu kabi oddiy dasturda nomlarni oʻzgartirish shart emas, chunki baribir undan foydalanish oson. Biroq, elementlarning to‘g‘ri nomlanishi siz boshidanoq o‘rganishingiz kerak bo‘lgan narsadir.

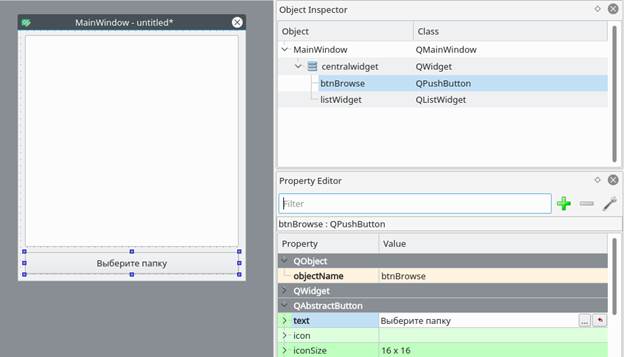
Element xususiyatlari***Property Editor*** bo‘limida o‘zgartirilishi mumkin.

*Ko‘rsatma*: Ish jarayonini tezlashtirish uchun siz Qt Designer interfeysiga tez-tez ishlatiladigan elementlarning o‘lchamini o‘zgartirishingiz, ko‘chirishingiz yoki qo‘shishingiz mumkin. *View* menyusi elementi orqali interfeysning yashirin/yopiq (скрытые/закрытые) qismlarini qo‘shishingiz mumkin.

Shaklga qo‘shgan tugmani bosing. Endi  *Property Editor* da  siz ushbu elementning barcha xususiyatlarini ko‘rishingiz kerak. Biz hozirda QAbstractButton bo‘limidagi objectName  va  matnga qiziqamiz.  Bo‘lim nomini bosish orqali *Property Editor* da bo‘limlarni yopishingiz mumkin.

*ObjectName* qiymatini  *btnBrowse*  ga  o‘zgartiring  va  matnni  *Jildni*tanlang.

**Bu shunday bo‘lishi kerak:**



**Nazorat savollari:**

1. Pythonda gui bilan ishlash uchun qanday kutubxonalari mavjud?
2. PyQT5 paketini qanday o‘rnatiladi va chaqirib ishlatish qanday amalga oshiriladi?
3. pyuic5 buyrug‘i nima uchun foydalaniladi?
4. Papkalarni ochish uchun qanday kutubxonadan foydalaniladi?
5. Kataloglar ustida qanday amallar bajarish mumkin?
6. Faylni saqlash uchun qanday amal bajariladi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.

**7-mashg‘ulot.** Matn muharriri dasturini yozish.

**O‘quv savollari:**

1. PyQt5 da matn muharriri dizaynidagi elementlarning funksiyalari uchun dastur yozish.
2. Dasturni testlash.

**1. PyQt5 da matn muharriri dizaynidagi elementlarning funksiyalari uchun dastur yozish**

Ro‘yxat ob’ektining nomi  listWidget bo‘lib , bu holda yaxshi. Dizaynni loyiha papkasida design.ui sifatida saqlang.

**Dizaynni kodga aylantirish**

Albatta foydalanishingiz mumkin. ui fayllarini to‘g‘ridan-to‘g‘ri Python kodidan olish mumkin, ammo osonroq ko‘rinishi mumkin bo‘lgan yana bir usul mavjud. Siz kodni o‘zgartirishingiz mumkin. ui faylini Python fayliga kiritamiz, keyin uni import qilishimiz va ishlatishimiz mumkin. Buning uchun terminal/buyruqlar satridan **pyuic5** buyrug‘idan foydalanamiz.

Konvertatsiya qilish uchun. ui faylini Python faylida design.py deb nomlagan holda quyidagi buyruqdan foydalaning:

$ pyuic5 path/to/ design.ui -o output/path/to/design.py

**Kodni yozishni boshlaymi**

Endi bizda QtDesigner dasturida yaratilgan dizayn faylining python faylga o‘girilgan versiyasi design.py nomli fayli bor va biz uning mantiqiy qismini ya’ni asosiy dastur qismini yozish ustida ishlashni boshlaymiz.

Birinchi navbatda **design.py** joylashgan papkada main.py faylini yarating.

**QtDesigner dasturida yaratilgan dizayndan foydalanish**

Python GUI ilovasi uchun sizga quyidagi modullar kerak bo‘ladi:

import sys

from PyQt5 import QtWidgets

Bizga avval yaratilgan dizayn kodi ham kerak, shuning uchun biz uni ham import qilamiz:

import design # Bu bizning o‘zgartirilgan dizayn faylimiz

Dizayn fayli har safar dizayn o‘zgartirilganda butunlay qayta yozilishi sababli, biz uni o‘zgartirmaymiz. Buning o‘rniga biz yangi klass yaratamiz,  ExampleApp , uni barcha funksiyalaridan foydalanish uchun dizayn kodi bilan birlashtiramiz:

Class ExampleApp(QtWidgets.QMainWindow, design.Ui\_MainWindow):

def \_\_init\_\_(self):

# Bu o‘zgaruvchilarga, usullarga kirish uchun bu erda kerak

# va hokazo. design.py faylida

        super().\_\_init\_\_()

self.setupUi(self)

# Bu dizaynimizni ishga tushirish uchun kerak

Ushbu sinfda biz interfeys elementlari bilan o‘zaro aloqada bo‘lamiz, ulanishlar va bizga kerak bo‘lgan barcha narsalarni qo‘shamiz. Lekin birinchi navbatda, biz kodni ishga tushirganimizda sinfni ishga tushirishimiz kerak. Buni asosiy ( ) funksiyada hal qilamiz  :

def main():

app = QtWidgets.QApplication(sys.argv)

window = ExampleApp()

window.show()

app.exec\_()

Va bu funktsiyani bajarish uchun biz tanish qurilishdan foydalanamiz:

if \_\_name\_\_ == '\_\_main\_\_':

main()

Natijada main.py quyidagicha ko‘rinadi:

import sys

from PyQt5 import QtWidgets

import design

class ExampleApp(QtWidgets.QMainWindow, design.Ui\_MainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

def main():

app = QtWidgets.QApplication(sys.argv)

window = ExampleApp()

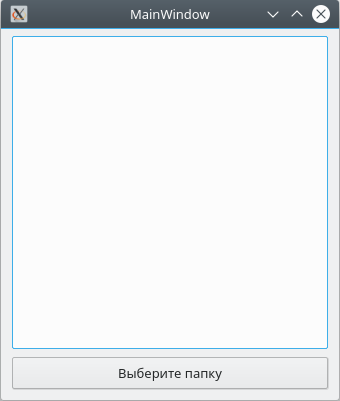
window.show()

app.exec\_()

if \_\_name\_\_ == '\_\_main\_\_':

main()

Agar biz ushbu kodni ishga tushirsak: $ python3 main.py ilovamiz ishga tushadi!



Lekin tugmani bosish hech narsa qilmaydi, shuning uchun biz u bilan shug‘ullanishimiz kerak.

**Python GUI ilovamizga funksionallikni qo‘shish**

Eslatma  Qo‘shimcha barcha kodlar ExampleApp sinfida yozilgan  .

*Keling, “****papkani tanlash****”* tugmasidan boshlaylik. Siz hodisani tugmani bosish kabi funksiyaga bog‘lashingiz mumkin:

            self.btnBrowse.clicked.connect(self.browse\_folder)

Ushbu qatorni ilova ExampleApp sinfining \_\_ init \_\_  metodiga qo‘shib qo‘ying. Endi buni batafsil ko‘rib chiqaylik:

* self.btnBrowse : bu yerda  btnBrowse - biz Qt Designer’da  aniqlagan ob’ektning nomi . o‘z  o‘zi uchun gapiradi va hozirgi sinfga tegishli bo‘lganligini anglatadi;
* cliked - biz bog‘lamoqchi bo‘lgan voqea . Turli elementlarda turli hodisalar mavjud, masalan, ro‘yxat vidjetlarida  itemSelectionChanged  va boshqalar mavjud;
* connect ( ) - o‘tkazilgan funksiyaga qo‘ng‘iroq bilan hodisani bog‘laydigan usul;
* self.browse\_folder        \_folder bu biz ExampleApp sinfida  tasvirlangan funksiya (metod) xolos.

Papka tanlash dialogini ochish uchun biz o‘rnatilgan QtWidgets.QFileDialog.getExistingDirectory metodidan foydalanishimiz mumkin :

            katalog = QtWidgets.QFileDialog.getExistingDirectory (self, "Papkani tanlash")

Agar foydalanuvchi katalogni tanlasa,  katalog o‘zgaruvchisi  tanlangan katalogning mutlaq yo‘liga o‘rnatiladi, aks holda u  Yo‘q ga o‘rnatiladi . Agar foydalanuvchi dialog oynasini yopsa, kodni boshqa bajarmaslik uchun biz  if katalogidan foydalanamiz : buyrug‘i.

Katalog tarkibini ko‘rsatish uchun biz  os ni import qilishimiz kerak :

            import os

Va shunday tarkib ro‘yxatini oling:

            os.listdir (path)

listWidget- ga elementlar qo‘shish uchun  addItem ( ) usulidan foydalanamiz va barcha elementlarni o‘chirish uchun bizda  self.listWidget.clear () mavjud.

Natijada,  browse\_folder funksiyasi  quyidagicha ko‘rinishi kerak:

def browse\_folder(self):

self.listWidget.clear()

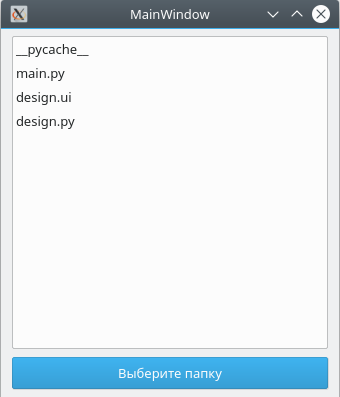
directory = QtWidgets.QFileDialog.getExistingDirectory (self, "papkani tanlash")

if directory:

for file\_name in os.listdir(directory):

self.listWidget.addItem(file\_name)

Endi, agar biz dasturni ishga tushirsak, tugmani bosing va katalogni tanlang, biz ko‘ramiz:



Python GUI ilovamizning butun kodi shunday ko‘rinadi:

import sys

import os

from PyQt5 import QtWidgets

import design

class ExampleApp(QtWidgets.QMainWindow, design.Ui\_MainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.btnBrowse.clicked.connect(self.browse\_folder)

def browse\_folder(self):

self.listWidget.clear()

directory = QtWidgets.QFileDialog.getExistingDirectory(self, "tanlash")

if directory:

for file\_name in os.listdir(directory):

self.listWidget.addItem(file\_name)

def main():

app = QtWidgets.QApplication(sys.argv)

window = ExampleApp()

window.show()

app.exec\_()

if \_\_name\_\_ == '\_\_main\_\_':

main()

Bular Python GUI ilovasini ishlab chiqish uchun Qt Designer va PyQt dan foydalanish asoslari edi. Endi siz dastur dizaynini o‘zingining ideyalaringiz bilan o‘zgartirishingiz va kodini o‘zgartirish uchun **pyuic5** buyrug‘idan qanday foydalanishni bilasiz.

**Nazorat savollari:**

1. Pythonda gui bilan ishlash uchun qanday kutubxonalari mavjud?
2. PyQT5 paketini qanday o‘rnatiladi va chaqirib ishlatish qanday amalga oshiriladi?
3. Pyuic5 buyrug‘i nima uchun foydalaniladi?
4. Papkalarni ochish uchun qanday kutubxonadan foydalaniladi?
5. Kataloglar ustida qanday amallar bajarish mumkin?
6. Faylni saqlash uchun qanday amal bajariladi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.

**8-mashg‘ulot.** PyQt5 da Minesweeper o‘yini dizaynini yaratish.

**O‘quv savollari:**

1. PyQt5 da Minesweeper o‘yinini yaratish.
2. O‘yin uchun kerakli uskunalarni tanlash.
3. O‘lchamlarni o‘rnatish. O‘yin dizaynini yaratish.

**1. PyQt5 da Minesweeper o‘yinini yaratish**

Moonsweeper - bu bitta o‘yinchili boshqotirma video o‘yini. O‘yinning maqsadi halokatli B’ug musofirlariga yaqinlashmasdan, qo‘ngan kosmik raketangiz atrofidagi hududni o‘rganishdir. Sizning ishonchli tricounteringiz sizga yaqin atrofdagi B’ugs sonini aytib beradi.

Bu Minesweeperda modellashtirilgan oddiy bitta o‘yinchili kashfiyot o‘yini, unda siz yashirin minalarga tegmasdan barcha plitkalarni ochishingiz kerak. Ushbu amalga oshirishda QWidget o‘z holatini minalar, holati va qo‘shni minalar soni sifatida saqlaydigan plitkalar uchun maxsus ob’ektlardan foydalaniladi. Ushbu versiyada minalar begona xatolar (B’ug) bilan almashtiriladi, ammo ular boshqa har qanday narsa bo‘lishi mumkin.

Windows, Linux va Mac uchun o‘rnatuvchilarni to‘liq manba kodi bilan birga yuqorida yuklab olish mumkin.

**Dasturni tuzish**

Dastur PyQt5 paketi yordamida tuziladi shuning uchun PyQt5 kutubxonasi kompyuterimizga o‘rnatilishi kerak bo‘ladi. O‘rnatishni esa oldingi darslarimizda ko‘rib chiqqanmiz.

PyQt5 kutubxonasini dasturimizga import qilib olamiz:

from PyQt5.QtGui import \*

from PyQt5.QtWidgets import \*

from PyQt5.QtCore import \*

**2. O‘yin uchun kerakli uskunalarni tanlash**

Minsweeper o‘yin maydoni NxN tarmog‘i bo‘lib, unda ma’lum miqdordagi minalar mavjud bo‘ladi. Biz foydalanadigan o‘lchamlar va minalar soni Minesweeper dasturining Windows versiyasi uchun standart qiymatlardan olingan. Foydalanilgan qiymatlar quyidagi jadvalda keltirilgan:

| **Daraja** | **O‘lchamlari** | **Minalar soni** |
| --- | --- | --- |
| Easy | 8 x 8 | 10 |
| Medium | 16 x 16 | 40 |
| Hard | 24 x 24 | 99 |

Biz bu qiymatlarni dastur kodining yuqori qismida LEVELS nomli doimiyga saqlaymiz . Barcha o‘yin maydonlari kvadrat bo‘lgani uchun biz qiymatni faqat bir marta saqlashimiz kerak (8, 16 yoki 24).

LEVELS = [

("Easy", 8, 10),

("Medium", 16, 40),

("Hard", 24, 99)

]

O‘yin to‘ri bir qancha usullarda ko‘rsatilishi mumkin, masalan, o‘yin pozitsiyalarining turli holatlarini ifodalovchi ikki o‘lchamli *"ro‘yxatlar ro‘yxati"* (mening, ochilgan, bayroqlangan).

Biroq, bu amalga oshirish ob’ektga yo‘naltirilgan yondashuvdan foydalanadi. Xaritadagi alohida kvadratlar ularning hozirgi holati haqidagi barcha tegishli ma’lumotlarni o‘z ichiga oladi va shuningdek, o‘zlarini chizish uchun javobgardir.

**3. O‘yin uchun kerakli uskunalarni tanlash**

Qt-da biz buni QWidgetoddiy bo‘yoq funktsiyasidan pastki sinflarga ajratish va keyin amalga oshirish orqali amalga oshirishimiz mumkin.

Bizning plitka ob’yektlarimiz quyi sinflarga bo‘linganligi sababli, QWidgetbiz ularni boshqa har qanday vidjet kabi joylashtirishimiz mumkin. Biz buni o‘rnatish orqali qilamiz QGridLayout.

self.grid = QGridLayout()

self.grid.setSpacing(5)

self.grid.setSizeConstraint(QLayout.SetFixedSize)

Biz o‘z pozitsiyamizdagi plitkalar vidjetlarini yaratib, ularni tarmoqqa qo‘shish orqali o‘yinni sozlashimiz mumkin. Darajani dastlabki sozlash oynadan o‘qiydi LEVELS va bir qator o‘zgaruvchilarni tayinlaydi. Oyna sarlavhasi va mina hisoblagichi yangilanadi, so‘ngra tarmoqni sozlash boshlanadi.

def set\_level(self, level):

self.level\_name, self.b\_size, self.n\_mines = LEVELS[level]

self.setWindowTitle("Moonsweeper - %s" % (self.level\_name))

self.mines.setText("%03d" % self.n\_mines)

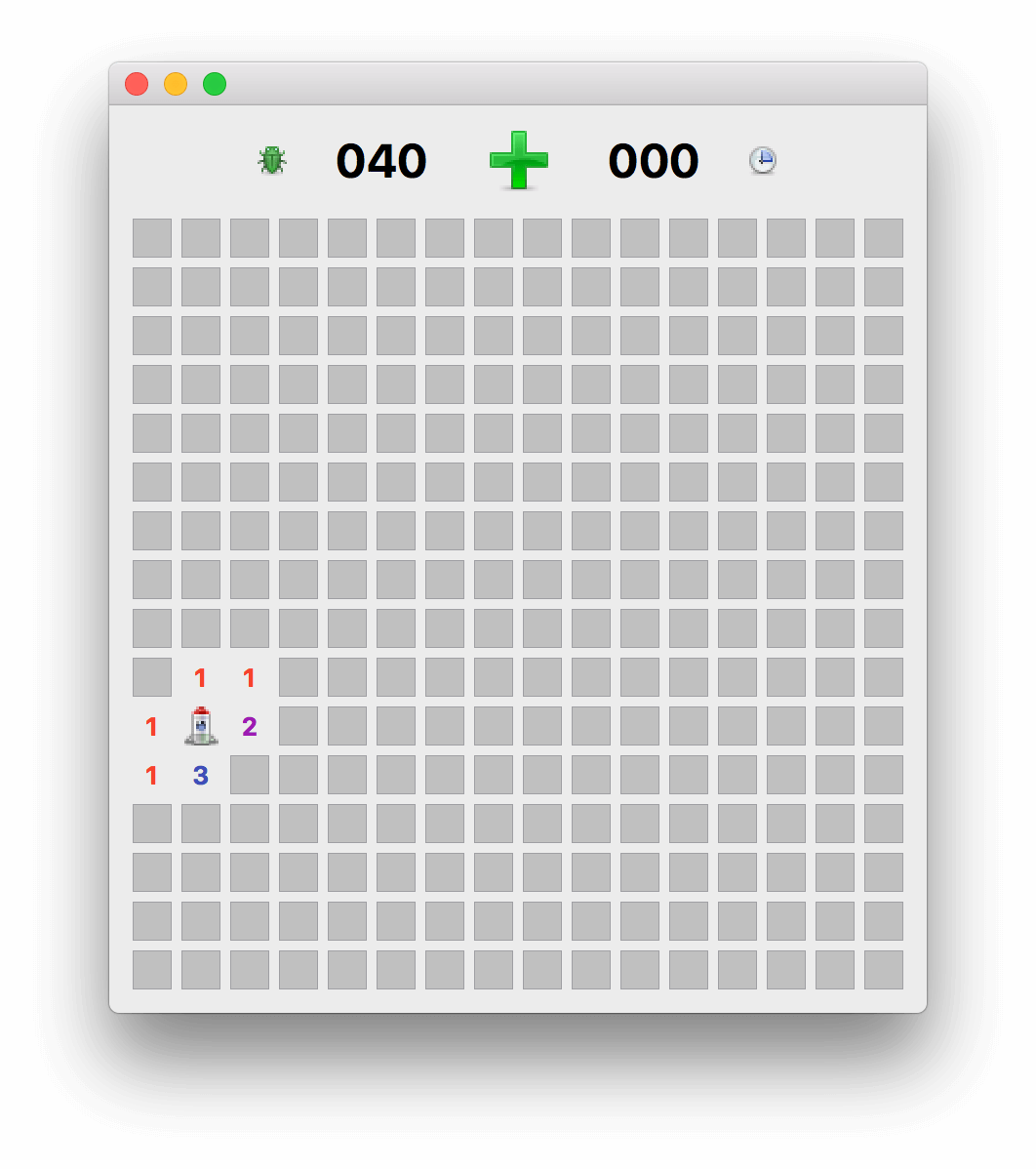
self.clear\_map()

self.init\_map()

self.reset\_map()

O‘rnatish funktsiyalari keyinroq ko‘rib chiqiladi.

Sinf Posplitkani ifodalaydi va uning xaritadagi tegishli o‘rni uchun barcha tegishli ma’lumotlarni o‘z ichiga oladi, jumladan, masalan, mina bo‘ladimi, aniqlanganmi, bayroqlanganmi va yaqin atrofdagi minalar soni.



**Nazorat savollari:**

1. Minesweeper o‘yini uchun qanday widgetlar kerak bo‘ladi?
2. Minesweeper o‘yinida rasmlarni joylashtirish qanday amalga oshiriladi?
3. Minalarni bosilganda sezishi qanday amalga oshiriladi?
4. Statusni ekranga chiqarish qanday amalga oshiriladi?
5. Random bilan minalarni joylashtirish qanday amalga oshiriladi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.

**9-mashg‘ulot.** PyQt5 da Minesweeper o‘yini dasturini yozish.

**O‘quv savollari:**

1. PyQt5 da Minesweeper o‘yini dizaynidagi tugmalarning funksiyalari uchun dastur yozish.
2. Dasturni testlash.
3. O‘yinni ishga tushirish.

**1. PyQt5 da Minesweeper o‘yini dizaynidagi tugmalarning funksiyalari uchun dastur yozish**

Har bir ob’ektda shuningdek, bosilgan, ochiladigan va kengaytiriladigan Pos 3 ta maxsus signal mavjud bo‘lib, biz ularni maxsus slotlarga ulaymiz. Nihoyat, biz oyna hajmini yangi tarkibga moslashtirish uchun o‘lchamini o‘zgartirishni chaqiramiz. Bu, aslida, faqat deraza qisqarganida kerak bo‘ladi - Qt uni avtomatik ravishda o‘stiradi.

def init\_map(self):

# Add positions to the map

for x in range(0, self.b\_size):

for y in range(0, self.b\_size):

w = Pos(x,y)

self.grid.addWidget(w, y, x)

# Connect signal to handle expansion.

w.clicked.connect(self.trigger\_start)

w.revealed.connect(self.on\_reveal)

w.expandable.connect(self.expand\_reveal)

QTimer.singleShot(0, lambda: self.resize(1,1)) # <1>

1. Taymer singleShot biz voqea tsikliga qaytganimizdan so‘ng va Qt yangi tarkibdan xabardor bo‘lganimizdan keyin o‘lchamni o‘zgartirishni ta’minlash uchun talab qilinadi.

Endi bizda pozitsion plitka ob’ektlari to‘plami mavjud, biz o‘yin taxtasining dastlabki sharoitlarini yaratishni boshlashimiz mumkin. Bu bir qator funktsiyalarga bo‘lingan. Biz ularni nomlaymiz \_reset(etakchi pastki chiziq tashqi foydalanish uchun mo‘ljallanmagan shaxsiy funktsiyani ko‘rsatadigan konventsiyadir). Asosiy funksiya reset\_mapuni sozlash uchun navbatma-navbat bu funksiyalarni chaqiradi.

Jarayon quyidagicha -

1. Maydondan barcha minalarni olib tashlang (va ma’lumotlarni qayta o‘rnating).
2. Maydonga yangi minalar qo‘shing.
3. Har bir pozitsiyaga ulashgan minalar sonini hisoblang.
4. Boshlang‘ich belgisini (raketa) qo‘shing va dastlabki tadqiqotni ishga tushiring.
5. Taymerni qayta o‘rnating.

Buni amalga oshirish uchun kod:

def reset\_map(self):

self.\_reset\_position\_data()

self.\_reset\_add\_mines()

self.\_reset\_calculate\_adjacency()

self.\_reset\_add\_starting\_marker()

self.update\_timer()

1 dan 5 gacha bo‘lgan alohida qadamlar quyida har bir qadam uchun kod bilan batafsil tavsiflanadi.

Birinchi qadam xaritadagi har bir pozitsiya uchun ma’lumotlarni qayta o‘rnatishdir. Biz doskadagi har bir pozitsiyani takrorlaymiz, .reset()har bir nuqtada vidjetni chaqiramiz. Funktsiya kodi .reset() bizning maxsus sinfimizda belgilangan Pos, biz keyinroq batafsil ko‘rib chiqamiz. Hozircha u minalarni, bayroqlarni tozalab, o‘z o‘rnini oshkor etilmagan holatga keltirishini bilish kifoya.

def \_reset\_position\_data(self):

# Clear all mine positions

for x in range(0, self.b\_size):

for y in range(0, self.b\_size):

w = self.grid.itemAtPosition(y, x).widget()

w.reset()

Endi barcha pozitsiyalar bo‘sh, biz xaritaga minalar qo‘shish jarayonini boshlashimiz mumkin. Minalar maksimal soni n\_minesyuqorida tavsiflangan daraja sozlamalari bilan belgilanadi.

def \_reset\_add\_mines(self):

# Add mine positions

positions = []

while len(positions) < self.n\_mines:

x, y = random.randint(0, self.b\_size-1), random.randint(0, self.b\_size-1)

if (x ,y) not in positions:

w = self.grid.itemAtPosition(y,x).widget()

w.is\_mine = True

positions.append((x, y))

# Calculate end-game condition

self.end\_game\_n = (self.b\_size \* self.b\_size) - (self.n\_mines + 1)

return positions

Minalar joyida bo‘lsa, endi biz har bir pozitsiya uchun "qo‘shni" raqamini hisoblashimiz mumkin - berilgan nuqta atrofida 3x3 o‘lchamdagi to‘rdan foydalanib, yaqin atrofdagi minalar sonini. Maxsus funksiya shunchaki ma’lum va joylashuv get\_surroundingatrofidagi pozitsiyalarni qaytaradi . Biz kon va do‘kon bo‘lgan bularning sonini hisoblaymiz .xyis\_mine == True

Bu yerda qo‘shni hisoblarni oldindan hisoblash keyinchalik ochish mantiqini soddalashtirishga yordam beradi. Ammo bu shuni anglatadiki, biz foydalanuvchiga o‘zining dastlabki harakatini tanlashiga ruxsat bera olmaymiz - biz buni "raketa atrofidagi dastlabki tadqiqot" deb tushuntirib, uni butunlay oqilona qilishimiz mumkin.

Hisobni kechiktirish orqali buni o‘zingiz hal qilishga harakat qiling!

def \_reset\_calculate\_adjacency(self):

def get\_adjacency\_n(x, y):

positions = self.get\_surrounding(x, y)

return sum(1 for w in positions if w.is\_mine)

# Add adjacencies to the positions

for x in range(0, self.b\_size):

for y in range(0, self.b\_size):

w = self.grid.itemAtPosition(y, x).widget()

w.adjacent\_n = get\_adjacency\_n(x, y)

*Birinchi harakat har doim* haqiqiy bo‘lishini ta’minlash uchun boshlang‘ich belgisi ishlatiladi . Bu biz mina bo‘lmagan pozitsiyani topmagunimizcha, tasodifiy pozitsiyalarni samarali sinab ko‘rish orqali tarmoq bo‘ylab *qo‘pol kuch* qidirish sifatida amalga oshiriladi . Buning uchun qancha urinishlar ketishini bilmasligimiz sababli, uni uzluksiz halqaga o‘rashimiz kerak.

Bu joy topilgach, biz uni boshlang‘ich joy sifatida belgilaymiz va keyin atrofdagi barcha pozitsiyalarni o‘rganishni ishga tushiramiz. Biz tsikldan chiqib, tayyor holatni tiklaymiz.

def \_reset\_add\_starting\_marker(self):

# Place starting marker.

# Set initial status (needed for .click to function)

self.update\_status(STATUS\_READY)

while True:

x, y = random.randint(0, self.b\_size - 1), random.randint(0, self.b\_size - 1)

w = self.grid.itemAtPosition(y, x).widget()

# We don’t want to start on a mine.

if not w.is\_mine:

w.is\_start = True

w.is\_revealed = True

w.update()

# Reveal all positions around this, if they are not mines either.

for w in self.get\_surrounding(x, y):

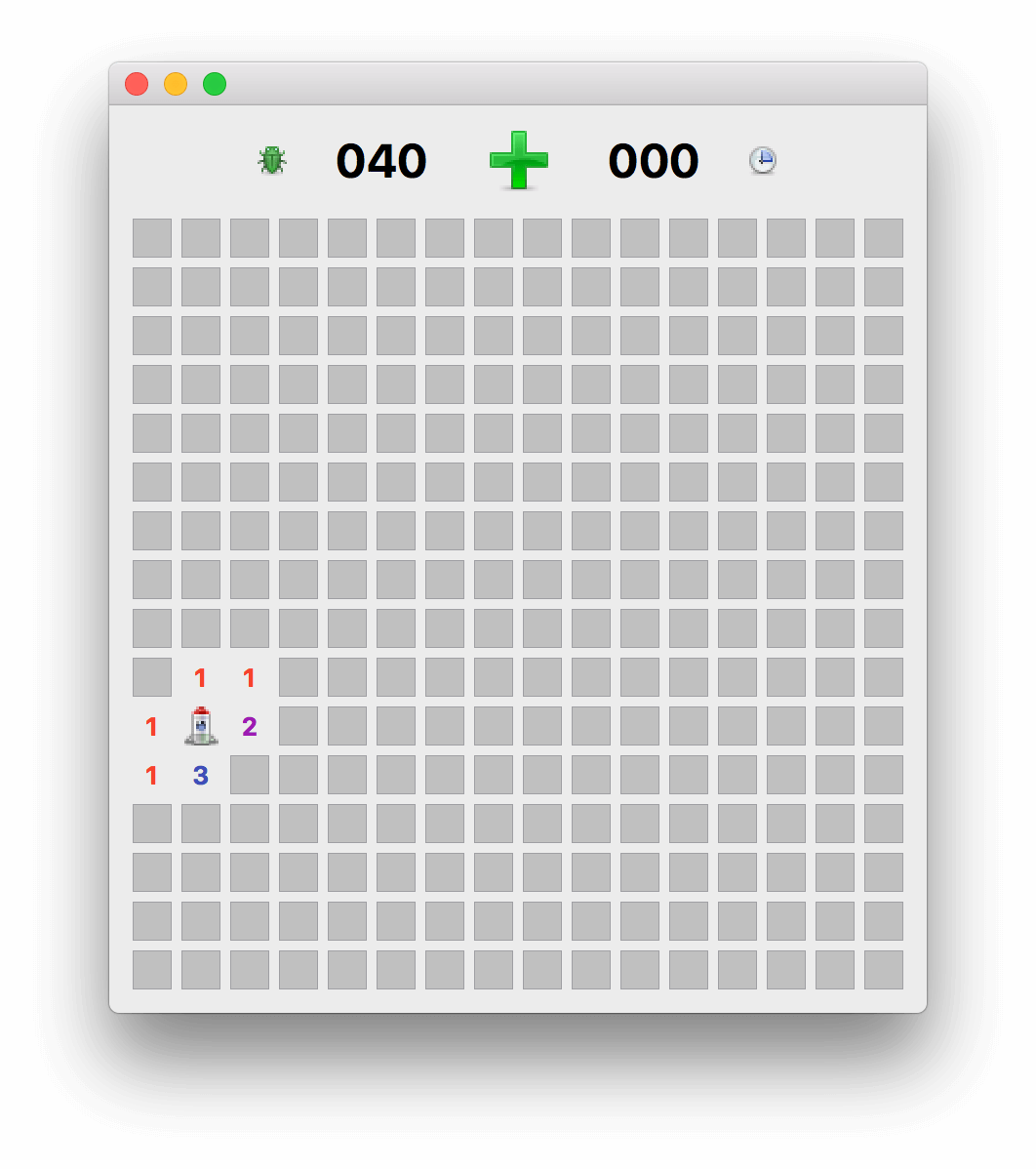
if not w.is\_mine:

w.click()

break

# Reset status to ready following initial clicks.

self.update\_status(STATUS\_READY)



**Plitkalarni joylashtirish**

O‘yin strukturaviy o‘yin bo‘lib, individual plitka pozitsiyalari o‘z holati haqida ma’lumotga ega bo‘ladi. Bu shuni anglatadiki, Posplitkalar o‘zlarining o‘yin mantiqlarini boshqarishga qodir.

Sinf nisbatan murakkab bo‘lganligi sababli Pos, u bu erda bir necha qismlarga bo‘linadi va ular o‘z navbatida muhokama qilinadi. Dastlabki o‘rnatish \_\_init\_\_bloki oddiy bo‘lib, xva ypozitsiyasini qabul qiladi va uni ob’ektda saqlaydi. Pospozitsiyalar yaratilgandan keyin hech qachon o‘zgarmaydi.

O‘rnatishni yakunlash uchun .reset()barcha ob’ekt atributlarini sukut bo‘yicha, nol qiymatlariga qaytaradigan funksiya chaqiriladi. Bu konni *boshlang‘ich pozitsiyasi emas* , *mina emas* , *oshkor qilinmagan* va *belgilanmagan* deb belgilaydi . Biz qo‘shni hisobni ham tiklaymiz.

class Pos(QWidget):

expandable = pyqtSignal(int,int)

revealed = pyqtSignal(object)

clicked = pyqtSignal()

def \_\_init\_\_(self, x, y, \*args, \*\*kwargs):

super(Pos, self).\_\_init\_\_(\*args, \*\*kwargs)

self.setFixedSize(QSize(20, 20))

self.x = x

self.y = y

self.reset()

def reset(self):

self.is\_start = False

self.is\_mine = False

self.adjacent\_n = 0

self.is\_revealed = False

self.is\_flagged = False

self.update()

O‘yin o‘yin maydonidagi plitkalar bilan sichqonchaning o‘zaro ta’siriga qaratilgan, shuning uchun sichqonchani bosishlarini aniqlash va ularga munosabat bildirish asosiy hisoblanadi. Qt da biz sichqonchani bosishlarini aniqlash orqali ushlaymiz mouseReleaseEvent. Buni bizning shaxsiy vidjetimiz uchun qilish uchun Posbiz sinfda ishlov beruvchini aniqlaymiz. Bu QMouseEventsodir bo‘lgan voqealarni o‘z ichiga olgan ma’lumot bilan qabul qilinadi. Bunday holda, biz faqat sichqonchaning chap tugmasi yoki o‘ng tugmasi bilan bo‘shatilganligi bilan qiziqamiz.

Sichqonchaning chap tugmachasini bosish uchun biz kafelning bayroqlanganligini yoki allaqachon ochilganligini tekshiramiz. Agar shunday bo‘lsa, biz chertishni e’tiborsiz qoldiramiz - bayroqchali plitalarni "xavfsiz" qilib, tasodifan bosish mumkin emas. Agar kafel belgilanmagan bo‘lsa, biz shunchaki .click()usulni ishga tushiramiz (keyinroq ko‘ring).

*Sichqonchaning o‘ng tugmasi bilan ko‘rsatilmagan* plitkalarni bosish uchun biz .toggle\_flag()bayroqni yoqish va o‘chirish usulini chaqiramiz.

def mouseReleaseEvent(self, e):

if(e.button()==Qt.RightButtonand not self.is\_revealed):

self.toggle\_flag()

elif (e.button() == Qt.LeftButton):

# Block clicking on flagged mines.

if not self.is\_flagged and not self.is\_revealed:

self.click()

Ishlovchi tomonidan chaqiriladigan usullar mouseReleaseEventquyida tavsiflanadi.

Ishlovchi .toggle\_flagshunchaki .is\_flaggedo‘ziga teskarisini o‘rnatadi ( Truebo‘ladi False, Falsebo‘ladi True) uni yoqish va o‘chirish effektiga ega. .update()E’tibor bering, biz holatni o‘zgartirib, qayta chizishga majburlash uchun qo‘ng‘iroq qilishimiz kerak . .clickedShuningdek, biz taymerni ishga tushirish uchun ishlatiladigan maxsus signalimizni chiqaramiz - chunki bayroqni qo‘yish kvadratni ochish emas, balki boshlang‘ich sifatida ham hisoblanishi kerak.

Usul .click()sichqonchaning chap tugmachasini bosadi va o‘z navbatida kvadratni ochishni boshlaydi. Agar bunga qo‘shni minalar soni Posnolga teng bo‘lsa, biz .expandable o‘rganilayotgan hududni avtomatik ravishda kengaytirish jarayonini boshlash uchun signalni ishga tushiramiz (keyinroq ko‘ring). Nihoyat, biz yana .clickedo‘yin boshlanishini bildirish uchun chiqaramiz.

Nihoyat, .reveal()usul kafel allaqachon aniqlanganligini tekshiradi va agar .is\_revealedbo‘lmasa True. .update()Vidjetni qayta bo‘yashni boshlash uchun yana qo‘ng‘iroq qilamiz .

Signalning ixtiyoriy emissiyasi .revealedfaqat o‘yin oxiri to‘liq xaritasini ochish uchun ishlatiladi. Har bir ochilish qaysi plitkalar aniqlanishi mumkinligini aniqlash uchun qo‘shimcha qidiruvni ishga tushirganligi sababli, butun xaritani ochish juda ko‘p ortiqcha qo‘ng‘iroqlarni keltirib chiqaradi. Bu erda signalni bostirish orqali biz bundan qochamiz.

def toggle\_flag(self):

self.is\_flagged = not self.is\_flagged

self.update()

self.clicked.emit()

def click(self):

self.reveal()

if self.adjacent\_n == 0:

self.expandable.emit(self.x, self.y)

self.clicked.emit()

def reveal(self, emit=True):

if not self.is\_revealed:

self.is\_revealed = True

self.update()

if emit:

self.revealed.emit(self)

Va nihoyat, biz vidjetimiz uchun joriy joylashuv holatini ko‘rsatishni boshqarish uchun maxsus paintEventusulni aniqlaymiz. Pos[bobda] tasvirlanganidek, vidjet tuvalini maxsus bo‘yashni amalga oshirish uchun biz chizishimiz kerak bo‘lgan chegaralarni - bu holda vidjetning tashqi chegarasini ko‘rsatadigan belgini QPainterolamiz .event.rect()Pos

Ochilgan plitkalar, kafelning *boshlang‘ich pozitsiyasi* , *bomba* yoki *bo‘sh joy* bo‘lishiga qarab turlicha chiziladi . Birinchi ikkitasi mos ravishda raketa va bomba piktogrammalari bilan ifodalanadi. QRectUlar yordamida plitka ichiga tortiladi .drawPixmap. QImageE’tibor bering, biz o‘tish orqali o‘tish orqali doimiylarni piksel xaritalariga aylantirishimiz kerak QPixmap.

Siz shunday deb o‘ylashingiz mumkin: "Nima uchun ularni faqat ob’ektlar sifatida saqlamaslik kerak, chunki biz aynan shu narsadan foydalanamiz? Afsuski, siz o‘zingiz ishga tushmasdan oldin ob’ektlar QPixmap yarata olmaysiz. QPixmapQApplication

Bo‘sh pozitsiyalar uchun (raketalar emas, bombalar emas) biz ixtiyoriy ravishda qo‘shni raqamni ko‘rsatamiz, agar u noldan katta bo‘lsa. Matnni matnga chizish uchun biz , hizalama bayroqlari va qator sifatida chizish uchun raqamni o‘tkazishdan QPainter foydalanamiz. Foydalanish qulayligi uchun biz har bir raqam uchun standart rangni belgilab oldik (da saqlangan )..drawText() QRectNUM\_COLORS

*Ochilmagan* plitkalar uchun biz to‘rtburchakni ochiq kul rang bilan to‘ldirib, plitka chizamiz va quyuqroq kul rangning 1 pikselli chegarasini chizamiz. Agar o‘rnatilgan bo‘lsa, biz va plitkadan .is\_flaggedfoydalanib, kafelning yuqori qismiga bayroq belgisini ham chizamiz .drawPixmapQRect

def paintEvent(self, event):

p = QPainter(self)

p.setRenderHint(QPainter.Antialiasing)

r = event.rect()

if self.is\_revealed:

if self.is\_start:

p.drawPixmap(r, QPixmap(IMG\_START))

elif self.is\_mine:

p.drawPixmap(r, QPixmap(IMG\_BOMB))

elif self.adjacent\_n > 0:

pen = QPen(NUM\_COLORS[self.adjacent\_n])

p.setPen(pen)

f = p.font()

f.setBold(True)

p.setFont(f)

p.drawText(r,Qt.AlignHCenter | Qt.AlignVCenter, str(self.adjacent\_n))

else:

p.fillRect(r, QBrush(Qt.lightGray))

pen = QPen(Qt.gray)

pen.setWidth(1)

p.setPen(pen)

p.drawRect(r)

if self.is\_flagged:

p.drawPixmap(r, QPixmap(IMG\_FLAG))

**Mexanika**

Biz odatda ma’lum bir nuqta atrofidagi barcha plitkalarni olishimiz kerak, shuning uchun bizda bu maqsad uchun maxsus funktsiya mavjud. U nuqta atrofida 3x3 o‘lchamdagi to‘r bo‘ylab oddiy takrorlanadi va biz to‘r chetlarida chegaradan chiqmasligimizni tekshirib ko‘ring ( 0 ≥ x ≤ self.b\_size). Qaytarilgan ro‘yxatda Poshar bir atrofdagi joydan vidjet mavjud.

def get\_surrounding(self, x, y):

positions = []

for xi in range(max(0, x - 1), min(x + 2, self.b\_size)):

for yi in range(max(0, y - 1), min(y + 2, self.b\_size)):

if not (xi == x and yi == y):

positions.append(self.grid.itemAtPosition(yi, xi).widget())

return positions

Usul expand\_revealnolga qo‘shni minalar bo‘lgan kafelni bosishga javoban ishga tushiriladi. Bunday holda, biz bosish atrofidagi maydonni nolga qo‘shni minalar bo‘lgan har qanday bo‘shliqqa kengaytirishni xohlaymiz, shuningdek, bu kengaytirilgan maydonning chegarasi atrofidagi har qanday kvadratlarni (bu minalar emas) aniqlamoqchimiz.

Biz keyingi iteratsiyada tekshiriladigan pozitsiyalarni o‘z ichiga olgan ro‘yxat, ko‘rsatiladigan plitka vidjetlarini o‘z ichiga olgan to\_expandro‘yxat va tsikldan qachon chiqishni aniqlash uchun bayroqdan boshlaymiz. Birinchi marta yangi vidjetlar qo‘shilmasa, tsikl to‘xtaydi .to\_revealany\_addedto\_reveal

Loop ichida biz qayta o‘rnatamiz any\_addedva Falsero‘yxatni bo‘shatamiz, takrorlash uchun to\_expandvaqtinchalik do‘konni saqlab qo‘yamiz .l

X Har bir joy va joy uchun ybiz 8 ta vidjetni olamiz. Agar ushbu vidjetlardan biri meniki bo‘lmasa va ro‘yxatda bo‘lmasa, to\_revealbiz uni qo‘shamiz. Bu kengaytirilgan maydonning barcha qirralari ochilishini ta’minlaydi. to\_expandAgar pozitsiyada qo‘shni minalar bo‘lmasa, biz keyingi iteratsiyada tekshiriladigan koordinatalarni qo‘shamiz .

-ga har qanday nodavlat plitkalarni qo‘shish to\_revealva faqat hali mavjud bo‘lmagan plitkalarni kengaytirish orqali to\_revealbiz kafelga bir necha marta tashrif buyurmasligimizga ishonch hosil qilamiz.

def expand\_reveal(self, x, y):

"""

Iterate outwards from the initial point, adding new locations to the

queue. This allows us to expand all in a single go, rather than

relying on multiple callbacks.

"""

to\_expand = [(x,y)]

to\_reveal = []

any\_added = True

while any\_added:

any\_added = False

to\_expand, l = [], to\_expand

for x, y in l:

positions = self.get\_surrounding(x, y)

for w in positions:

if not w.is\_mine and w not in to\_reveal:

to\_reveal.append(w)

if w.adjacent\_n == 0:

to\_expand.append((w.x,w.y))

any\_added = True

# Iterate an reveal all the positions we have found.

for w in to\_reveal:

w.reveal()

**2. Dasturni testlash**

O‘yin oxiri holati sarlavha ustiga bosilgandan so‘ng, oshkor qilish jarayonida aniqlanadi. Ikkita mumkin bo‘lgan natija bor -

1. Plitka - bu kon, o‘yin tugadi.
2. Plitka kon emas, uni kamaytiring self.end\_game\_n.

Bu nolga yetguncha davom etadi , bu esa yoki ga self.end\_game\_nqo‘ng‘iroq qilib g‘alaba qozonish jarayonini boshlaydi . Muvaffaqiyat/muvaffaqiyatsizlik har ikki holatda ham xaritani ochish va tegishli holatni o‘rnatish orqali boshlanadi.game\_overgame\_won

def on\_reveal(self, w):

if w.is\_mine:

self.game\_over()

else:

self.end\_game\_n -= 1 # decrement remaining empty spaces

if self.end\_game\_n == 0:

self.game\_won()

def game\_over(self):

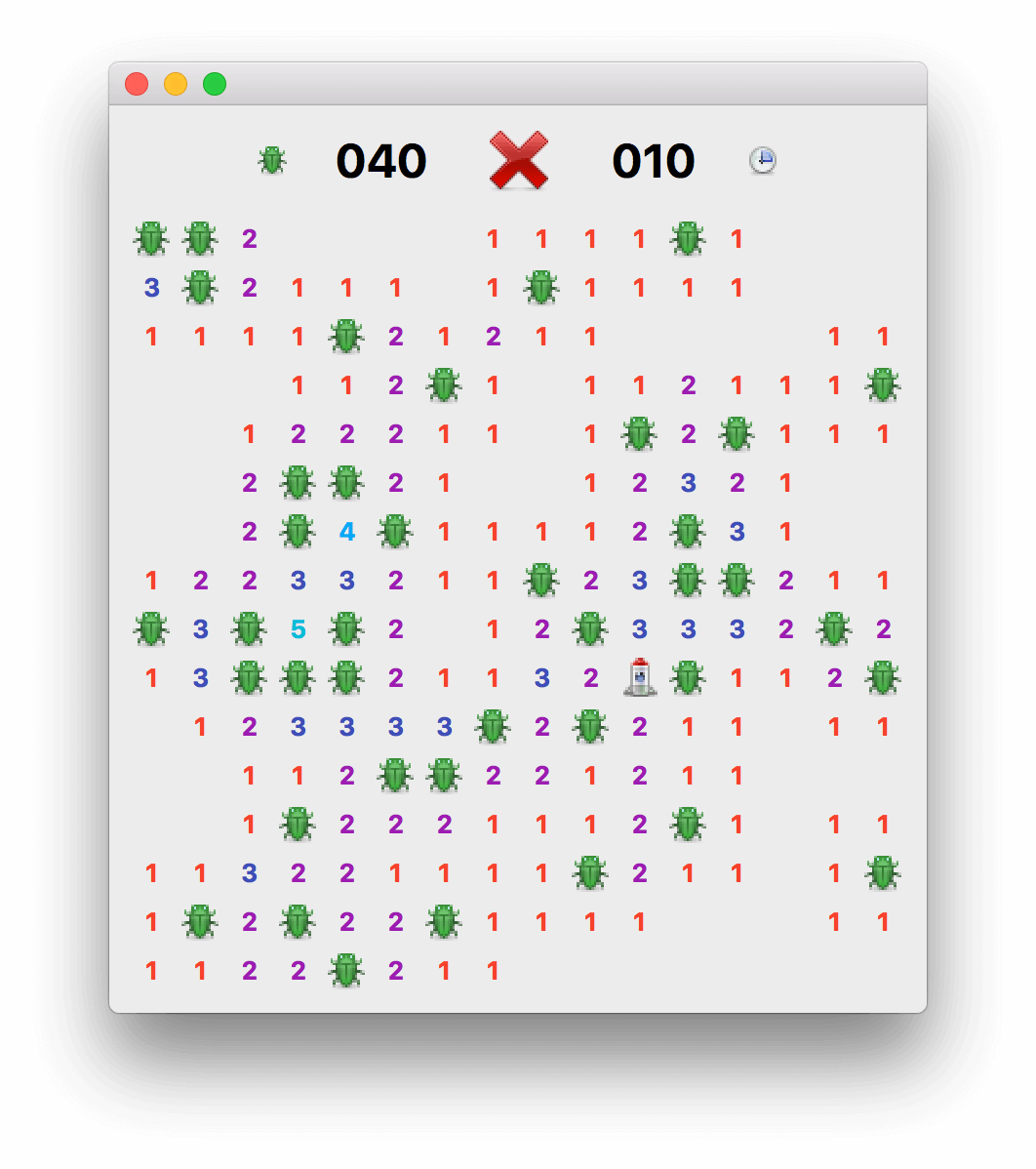
self.reveal\_map()

self.update\_status(STATUS\_FAILED)

def game\_won(self):

self.reveal\_map()

self.update\_status(STATUS\_SUCCESS)



**Nazorat savollari:**

1. Minesweeper o‘yini uchun qanday widgetlar kerak bo‘ladi?
2. Minesweeper o‘yinida rasmlarni joylashtirish qanday amalga oshiriladi?
3. Minalarni bosilganda sezishi qanday amalga oshiriladi?
4. Statusni ekranga chiqarish qanday amalga oshiriladi?
5. Random bilan minalarni joylashtirish qanday amalga oshiriladi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**1-mashg‘ulot.** Tarmoqda ma’lumot almashuvchi klient-server dasturini tuzish.

**O‘quv savollari:**

1. Socket modulining asosiy metodlari bilan tanishish.

**1. Socket modulining asosiy metodlari bilan tanishish**

Python tarmoq xizmatlariga kirishning ikki darajasini ta'minlaydi. Past darajada, siz asosiy operatsion tizimdagi asosiy rozetka yordamiga kirishingiz mumkin, bu sizga ulanishga yo‘naltirilgan va ulanishsiz protokollar uchun mijozlar va serverlarni amalga oshirish imkonini beradi.

Python, shuningdek, FTP, HTTP va boshqalar kabi ma'lum amaliy qatlam tarmoq protokollariga yuqori darajadagi kirishni ta'minlaydigan kutubxonalarga ega.

Ushbu bo‘lim sizga Internetdagi eng mashhur kontseptsiya - Socket Programming haqida tushuncha beradi.

**Soketlar nima?**

Soketlar ikki tomonlama aloqa kanalining so‘nggi nuqtalari hisoblanadi. Soketlar jarayon ichida, bitta mashinadagi jarayonlar o‘rtasida yoki turli qit'alardagi jarayonlar o‘rtasida aloqa qilishi mumkin.

Soketlar bir nechta turli turdagi quvurlar uchun amalga oshirilishi mumkin: Unix domen soketlari, TCP, UDP va boshqalar. *Soket* kutubxonasi umumiy transportlarni boshqarish uchun maxsus sinflarni, shuningdek, qolganlarini boshqarish uchun umumiy interfeysni taqdim etadi.

Soketlarning o‘z lug‘ati bor.

|  |  |
| --- | --- |
| **t/r** | **Muddati va tavsifi** |
| 1 | **Domen**  Transport mexanizmi sifatida foydalaniladigan protokollar oilasi. Bu qiymatlar AF\_INET, PF\_INET, PF\_UNIX, PF\_X25 va boshqalar kabi konstantalardir. |
| 2 | **turi**  Ikki soʻnggi nuqta oʻrtasidagi aloqa turi, odatda ulanishga yoʻnaltirilgan protokollar uchun SOCK\_STREAM va ulanishsiz protokollar uchun SOCK\_DGRAM. |
| 3 | **protokol**  Odatda nolga teng, bu domen va turdagi protokol variantini aniqlash uchun ishlatilishi mumkin. |
| 4 | **xost nomi**  Tarmoq interfeysi identifikatori -   * Xost nomi, toʻrt nuqtali manzil yoki ikki nuqta (va ehtimol nuqta) belgisidagi IPV6 manzili boʻlishi mumkin boʻlgan qator. * INADDR\_BROADCAST manzilini bildiruvchi “<broadcast>” qatori. * INADDR\_ANY ni belgilaydigan nol uzunlikdagi qator yoki * Xost bayt tartibida ikkilik manzil sifatida talqin qilingan butun son. |
| 5 | **port**  Har bir server bir yoki bir nechta portga qo‘ng‘iroq qilayotgan mijozlarni tinglaydi. Port Fixnum port raqami, port raqami qatori yoki xizmat nomi bo‘lishi mumkin. |

**Domen**

Transport mexanizmi sifatida foydalaniladigan protokollar oilasi. Bu qiymatlar AF\_INET, PF\_INET, PF\_UNIX, PF\_X25 va boshqalar kabi konstantalardir.

**type**

Ikki soʻnggi nuqta oʻrtasidagi aloqa turi, odatda ulanishga yoʻnaltirilgan protokollar uchun SOCK\_STREAM va ulanishsiz protokollar uchun SOCK\_DGRAM.

**protokol**

Odatda nolga teng, bu domen va turdagi protokol variantini aniqlash uchun ishlatilishi mumkin.

**xost nomi**

Tarmoq interfeysi identifikatori -

Xost nomi, toʻrt nuqtali manzil yoki ikki nuqta (va ehtimol nuqta) belgisidagi IPV6 manzili boʻlishi mumkin boʻlgan qator.

INADDR\_BROADCAST manzilini bildiruvchi “<broadcast>” qatori.

INADDR\_ANY ni belgilaydigan nol uzunlikdagi qator yoki

Xost bayt tartibida ikkilik manzil sifatida talqin qilingan butun son.

**port**

Har bir server bir yoki bir nechta portga qo‘ng‘iroq qilayotgan mijozlarni tinglaydi. Port Fixnum port raqami, port raqami qatori yoki xizmat nomi bo‘lishi mumkin.

***soket* moduli**

Soket yaratish uchun umumiy sintaksisga ega *socket* modulida mavjud *socket.socket() funksiyasidan foydalanishingiz kerak.*

s = socket.socket(socket\_family, socket\_type, protocol=0)

Bu erda variantlarning tavsifi:

* **socket\_family** AF\_UNIX yoki AF\_INET, avvalroq tushuntirilganidek.
* **socket\_type** - SOCK\_STREAM yoki SOCK\_DGRAM.
* **protokol** - odatda ko‘rsatilmaydi, standart 0.

**socket\_family** AF\_UNIX yoki AF\_INET, avvalroq tushuntirilganidek.

**socket\_type** - SOCK\_STREAM yoki SOCK\_DGRAM.

**protokol** - odatda ko‘rsatilmaydi, standart 0.

*Soket* ob'ektiga ega bo‘lganingizdan so‘ng , mijoz yoki server dasturini yaratish uchun kerakli funktsiyalardan foydalanishingiz mumkin. Quyida talab qilinadigan xususiyatlar ro‘yxati -

**Server soket usullari**

|  |  |
| --- | --- |
| **t/r** | **Usul va tavsif** |
| 1 | **s.bind()**  Ushbu usul manzilni (xost nomi, port raqamlari juftligi) rozetkaga bog‘laydi. |
| 2 | **s.tinglash()**  Ushbu usul TCP tinglovchisini o‘rnatadi va ishga tushiradi. |
| 3 | **s.accept()**  Bu passiv ravishda TCP mijoz ulanishini qabul qiladi, ulanish o‘rnatilguncha kutadi (blokirovka). |

**s.bind()**

Ushbu usul manzilni (xost nomi, port raqamlari juftligi) rozetkaga bog‘laydi.

**s.tinglash()**

Ushbu usul TCP tinglovchisini o‘rnatadi va ishga tushiradi.

**s.accept()**

Bu passiv ravishda TCP mijoz ulanishini qabul qiladi, ulanish o‘rnatilguncha kutadi (blokirovka).

**Mijoz soket usullari**

|  |  |
| --- | --- |
| **t/r** | **Usul va tavsif** |
| 1 | **s.connect()**  Ushbu usul TCP serveriga ulanishni faol ravishda boshlaydi. |

**s.connect()**

Ushbu usul TCP serveriga ulanishni faol ravishda boshlaydi.

**Umumiy soket usullari**

|  |  |
| --- | --- |
| **t/r** | **Usul va tavsif** |
| 1 | **s.recv()**  Ushbu usul TCP xabarini oladi |
| 2 | **s.send()**  Ushbu usul TCP xabarini yuboradi |
| 3 | **s.recvfrom()**  Bu usul UDP xabarini oladi |
| 4 | **s.sendto()**  Bu usul UDP xabarini yuboradi |
| 5 | **s.close()**  Ushbu usul rozetkani yopadi |
| 6 | **socket.getostname()**  Xost nomini qaytaradi. |

**s.recv()**

Ushbu usul TCP xabarini oladi

**s.send()**

Ushbu usul TCP xabarini yuboradi

**s.recvfrom()**

Bu usul UDP xabarini oladi

**s.sendto()**

Bu usul UDP xabarini yuboradi

**s.close()**

Ushbu usul rozetkani yopadi

**socket.getostname()**

Xost nomini qaytaradi.

**oddiy server.** Internet-serverlarni yozish uchun biz rozetka ob'ektini yaratish uchun rozetka modulida mavjud **soket funksiyasidan foydalanamiz.**Soket ob'ekti soket serverini sozlash uchun boshqa funktsiyalarni chaqirish uchun ishlatiladi.

Endi berilgan hostda xizmatingiz uchun *portni* belgilash uchun **bog‘lash (hostname, port) funksiyasini chaqiring.**

Keyin qaytarilgan ob'ektni *qabul qilish usulini chaqiring.*Ushbu usul mijoz siz ko‘rsatgan portga ulanishini kutadi va keyin ushbu mijozga ulanishni ifodalovchi *Connection ob'ektini qaytaradi.*

#!/usr/bin/python # This is server.py file

import socket # Import socket module

s = socket.socket() # Create a socket object

host = socket.gethostname() # Get local machine name

port = 12345 # Reserve a port for your service.

s.bind((host, port)) # Bind to the port

s.listen(5) # Now wait for client connection.

while True:

c, addr = s.accept() # Establish connection with client.

print 'Got connection from', addr

c.send('Thank you for connecting‘)

c.close() # Close the connection

**Oddiy mijoz.** Berilgan port 12345 va berilgan xostga ulanishni ochadigan juda oddiy mijoz dasturini yozamiz. *Python soket* moduli funksiyasidan foydalangan holda soket mijozini yaratish juda oddiy .

**Socket.connect(hostname, port)** portdagi *xost nomi bilan* TCP ulanishini ochadi . Ochiq rozetkaga ega bo‘lganingizdan so‘ng, siz undan istalgan kiritish-chiqarish ob'ekti kabi o‘qishingiz mumkin. Ishingiz tugagach, faylni yopganingizdek, uni yopishni unutmang.

Quyidagi kod juda oddiy mijoz bo‘lib, u berilgan xost va portga ulanadi, rozetkadan barcha mavjud ma'lumotlarni o‘qiydi va keyin chiqadi.

#!/usr/bin/python # This is client.py file

import socket # Import socket module

s = socket.socket() # Create a socket object

host = socket.gethostname() # Get local machine name

port = 12345 # Reserve a port for your service.

s.connect((host, port))

print s.recv(1024)

s.close() # Close the socket when done

Endi bu server.py ni fonda ishga tushiring va natijani ko‘rish uchun yuqoridagi client.py ni ishga tushiring.

# Following would start a server in background.

$ python server.py &

# Once server is started run client as follows:

$ python client.py

Bu quyidagi natijani beradi:

Got connection from ('127.0.0.1', 48437)

Thank you for connecting

**Python Internet modullari**

Python Network/Internet dasturlashdagi ba'zi muhim modullar ro‘yxati.

|  |  |  |  |
| --- | --- | --- | --- |
| **protokol** | **Umumiy funktsiya** | **Port raqami.** | **Python moduli** |
| HTTP | veb-sahifalar | 80 | httplib, urllib, xmlrpclib |
| NNTP | Usenet yangiliklari | 119 | nntplib |
| FTP | Fayl uzatish | 20 | ftplib, urllib |
| SMTP | Email yuborish | 25 | smtplib |
| POP3 | Email qabul qilinmoqda | 110 | poplib |
| IMAP4 | Email qabul qilinmoqda | 143 | imaplib |
| telnet | Buyruqlar qatorlari | 23 | telnetlib |
| gofer | Hujjatlarni topshirish | 70 | gopher, urlib |

Iltimos, FTP, SMTP, POP va IMAP protokollari bilan ishlash uchun yuqorida ko‘rsatilgan barcha kutubxonalarni tekshiring.

**Nazorat savollari:**

1. Tarmoq dasturlash deganda nimani tushunasiz?
2. Tarmoqda ishlash uchun python tilining qaysi paketi bilan ishlanadi?
3. Qanday metodalarni bilasiz?
4. Bind() metodining vazifasi qanday?
5. connect metodining vazifasi qanbday?
6. Protokol nima?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**2-mashg‘ulot.** Socket moduli bilan ishlash.

**O‘quv savollari:**

1. Socket modulining asosiy metodlari bilan tanishish.
2. .socket(), .bind, .listen, .accept(), .connect(), .send(), recv(), .close() metodlari bilan ishlash.

**1. Socket modulining asosiy metodlari bilan tanishish**

Soket - bu jarayonlar o‘rtasida ma’lumot almashinuvini ta’minlash uchun dasturlash interfeysi.

**Socket turlari:**

* Server - xabarlarni qabul qiluvchi socket.
* Mijoz - xabarlarni yuboradigan socket.

**Soketlar protokollarning transport qatlamida ishlaydi va shunga mos ravishda 2 tur mavjud:**

* **Oqim** (TCP asosida, kodda ko‘rsatilgan **SOCK\_STREAM**) - TCP protokoli asosida o‘rnatilgan ulanishga ega soketlar, ikki tomonlama bo‘lishi mumkin bo‘lgan baytlar oqimini uzatadi - ya’ni Ilova ma’lumotlarni qabul qilishi va yuborishi mumkin.
* **Datagram** (UDP asosida, kodda ko‘rsatilgan **SOCK\_DGRAM**) - ular o‘rtasida aniq ulanishni talab qilmaydigan socketlar. Xabar belgilangan soketga yuboriladi va shunga mos ravishda belgilangan socketdan qabul qilinishi mumkin.

Soket tarkibiasllida IP manzil va portdan iborat bo‘ladi.

**IP manzil** - IP protokoli yordamida qurilgan kompyuter tarmog‘idagi tugunning yagona tarmoq manzili. IPv4 protokoli versiyasida IP manzili uzunligi 4 bayt (masalan, 192.168.0.3), IPv6 protokoli versiyasida esa IP manzili 16 bayt (masalan, 2001:0db8:85a3:0000:0000: 8a2e: 0370: 7334). IP-manzil noyob bo‘lishi kerak.

**Port** - transport qatlami protokollari (TCP, UDP va boshqalar) sarlavhalarida yozilgan natural son. Port bir xil xost ichidagi paketni qabul qilish jarayonini aniqlash uchun ishlatiladi.

Python socketlar bilan ishlash uchun o‘rnatilgan socket kutubxonasidan foydalanadi. Modulning asosiy funktsiyalaridan biri bu socket()ulanish bilan ishlash uchun tegishli funktsiyalarga ega bo‘lgan socket tipidagi ob’ektni qaytaradigan funktsiyadir.:

class socket.socket

sock = socket.socket()

**2. .socket(), .bind, .listen, .accept(), .connect(), .send(), recv(), .close() metodlari bilan ishlash**

* **socket.bind(address)**- Soketni manzilga bog‘laydi (IP manzil va portni ishga tushiradi). Bundan oldin socket ulanmagan bo‘lishi kerak.
* **socket.listen([backlog])**- Ulanishlarni qabul qilish uchun serverni almashtiradi. “backlog (int)” parametri server qabul qiladigan ulanishlar sonidir.
* **socket.accept()**- Ulanishni qabul qiladi va mijozdan xabar kutayotganda dasturni bloklaydi. Natijada kortejni qaytaradi:
  + **conn**: ma’lumotlarni yuborish/qabul qilish uchun ishlatilishi mumkin bo‘lgan ulanish ob’ekti (socket);
  + **address**: mijozning manzili.
* **socket.recv(bufsize[, flags])**- socketdan ma’lumotlarni ikkilik formatda (baytlar to‘plami) o‘qiydi va qaytaradi. Parametr - bitta xabardagi baytlarning maksimal soni.**bufsize (int)**
* **socket.send(bytes[, flags])**- Mijozga ma’lumotlarni yuboradi va yuborilgan baytlar sonini qaytaradi. Parametr ikkilik ma’lumotlardir.**bytes (bytes)**
* **socket.close()**- Socketni yopadi.

Soket bilan ishlash ko‘p jihatdan fayl ob’yekti bilan ishlashga o‘xshaydi. Printsip - ochiq ulanish - ma’lumotlar - yopiq ulanish deb hisoblanadi.

**Nazorat savollari:**

* 1. port deb nimaga aytiladi?
  2. Socket nima ?
  3. Socketning qanday turlari mavjud?
  4. Socket ob’yektining qanday funksiyalari mavjud?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**3-mashg‘ulot.** Pythonda TCP klient-server dasturini tuzish.

**O‘quv savollari:**

1. Socket modulining asosiy metodlari yordamida client-server dasturini tuzish.

**1. Socket modulining asosiy metodlari yordamida client-server dasturini tuzish.**

Python-da o‘rnatilgan socket moduli tarmoq orqali muloqot qilish uchun funksionallikni ta’minlaydi. Ushbu modul soket sinfi shaklida so‘rovlarni yuborish va qabul qilish uchun past darajadagi interfeysda aniqlanadi. Va yaratayotgan narsadan qat’iy nazar - server yoki mijoz dasturi, so‘rovlarni yuborish uchun socket ob’yektini yaratish kerak bo‘ladi:

import socket

sock = socket.socket()

socket bilan ishlashni tugatgandan so‘ng, uni close() metodi yordamida yopish kerak.

import socket

client = socket.socket()

client.close()

Socket bilan keyingi harakatlar qanday rozetka yaratilayotganiga bog‘liq bo‘lib, server yoki mijoz uchun yaratiladi. Bunday holda, Python soketlarida eng oddiy mijozni yaratishni ko‘rib chiqiladi.

**Serverga bog‘lanish.** **connect()** metodi serverga ulanish uchun ishlatiladi.

socket.connect(address)

Ushbu **connect** metodi parametr sifatida server manzilini qabul qiladi. Manzil odatda kortej sifatida ifodalanadi

(host, port)

Birinchi element string sifatida xost hisoblanadi. Bu, masalan, "127.0.0.1" shaklidagi IP-manzil yoki lokol xost nomi bo‘lishi mumkin. Ikkinchi parametr raqamli port raqamidir. Port 0 dan 65535 gacha bo‘lgan 2 baytlik qiymatdir. Masalan:

import socket

client = socket.socket()

client.connect(("www.kun.uz", 80))

print("Connected...")

client.close()

Bu yerda birinchi parametr sifatida: "www.kun.uz" manziliga va ikkinchi parametr sifatida 80-portga ulanishga harakat qilmoqda. Ya’ni, www.kun.uz80-portda ishlaydigan oddiy veb-saytga ulanishga harakat qilinmoqda (odatda veb-server 80-portda ishlaydi). Ulangandan so‘ng, shunchaki konsolga satr chiqariladi va socketni yopiladi.

**Server bilan o‘zaro ulanish.**

Ulanish o‘rnatilgandan so‘ng serverga ma’lumotlarni yuborish va undan qabul qilish mumkin. Ma’lumotlarni yuborish uchun **socket.send()** usuli qo‘llaniladi, u parametr sifatida yuboriladigan ma’lumotlar to‘plamini oladi.

socket.send(bytes)

Soketdan ma’lumotlarni qabul qilish uchun **socket.recv()** usuli qo‘llaniladi.

bytes = socket.recv(bufsize[, flags])

Bu zarur parametr sifatida bir vaqtning o‘zida boshqa socketdan olinishi mumkin bo‘lgan baytlardagi maksimal bufer hajmini oladi. Bufer hajmini 2 ga karrali qilish yaxshidir, masalan, 512, 1024 va boshqalar. Qaytish qiymati boshqa socketdan olingan baytlar to‘plamidir.

Masalan, “www.kun.uz” serveriga ma’lumotlarni yuboriladi va undan qaytgan javobni qabul qilinadi:

import socket

client = socket.socket()

client.connect(("www.kun.uz", 80))

message = "GET / HTTP/1.1\r\nHost:www.kun.uz\r\n Connection: close\r\n\r\n"

print("Connecting...")

client.send(message.encode())

data = client.recv(1024)

print("Server sent: ", data.decode())

client.close()

Bunday holda, satr veb-sayt tushunadigan standart HTTP protokoli sarlavhalari bilan yuboriladi. "GET/HTTP/1.1\r\nHost: www.kun.uz HTTP so‘rov formati birinchi navbatda so‘rov turini, so‘ralgan manbaga yo‘lni va maxsus protokol versiyasidan iborat so‘rov qatorini () o‘z ichiga oladi . Ya’ni, bu yerda xabarda GET so‘rovi "/" yo‘li bo‘ylab yuborilganligini bildiriladi (ya’ni www.kun.uz saytining ildiziga). Bu holda HTTP / 1.1 protokoli qo‘llaniladi. So‘rov qatori ikki karet to‘plami va qaytariladigan belgilar qatori \r\n bilan tugashi kerak.

Bundan tashqari, HTTP so‘rovida sarlavhalar bo‘lishi mumkin. Shunday qilib, bu holda "Xost" sarlavhasini yuboriladi, bu esa xost manziliga ishora qiladi. Bu holda bu "www.kun.uz:80". Va shuningdek, bu holatda, "close" qiymatiga ega bo‘lgan "connection" sarlavhasini yuboriladi - bu qiymat serverga ulanishni yopishni buyuradi.

Bu yerda satrlarni emas, faqat baytlarni yuborish mumkinligi sababli, uni yuborilganida satrni baytlar to‘plamiga aylantiriladi. Buning uchun **bytes** sinf ob’yektini qaytaradigan **encode**() usuli qo‘llaniladi.

client.send(message.encode())

Serverga xabar yuborilganidan so‘ng, recv() funksiyasidan foydalanib ma’lumotlarni olishga harakat qilinadi. Bunday holda, olingan ma’lumotlar uchun bufer 1024 bayt hajmiga ega bo‘ladi. Usulning natijasi olingan ma’lumotlardir. Biroq, socket ma’lumotlarni baytlar to‘plami shaklida oladi (aniqrog‘i, ob’yekt bytes). Ularni satrga aylantirish uchun decode() usulidan foydalaning:

data = client.recv(1024) # serverdan ma’lumotlarni olish

print("Server sent: ", data.decode())

Va agar ushbu dasturni Python-da ishga tushirilsa, quyidagi kabi javob olish mumkin:

Connected...

Server sent: HTTP/1.1 301 Moved Permanently

Connection: close

Content-Length: 0

Server: Varnish

Retry-After: 0

Location: https://www.python.org/

Accept-Ranges: bytes

Date: Tue, 02 May 2023 17:52:32 GMT

Via: 1.1 varnish

X-Served-By: cache-bma1653-BMA

X-Cache: HIT

X-Cache-Hits: 0

X-Timer: S1683049953.637569,VS0,VE0

Strict-Transport-Security: max-age=63072000; includeSubDomains; preload

Shuningdek, server bizga http sarlavhalarida veb-sayt https://www.kun.uz/ manziliga ko‘chirilganligini ko‘rsatadigan qatorni yuboradi.

**Server dasturini yaratish**

Python-da serverni, shuningdek mijozni yaratish uchun soket sinfi ham ishlatiladi, ammo bu mijoz soketi bilan ishlashdan biroz farq qiladi.

**Serverni ulanish**

Server kiruvchi ulanishlarni kutib turadi (listining), ularni qandaydir tarzda qayta ishlaydi va javob yuboradi. Server o‘z ishini boshlashi uchun, avvalo, u ulanishlarni tinglaydigan manzilni aniqlab olishi kerak. Buning uchun **bind()** metodi qo‘llaniladi.

socket.bind(address)

Bu usul server ishga tushadigan manzilni qabul qiladi. Odatiy bo‘lib, manzil ikki elementdan iborat to‘plamdir:

(host, port)

Birinchi element string sifatida xost hisoblanadi. Bu, masalan, "127.0.0.1" shaklidagi IP-manzil yoki mahalliy xost nomi bo‘lishi mumkin. Ikkinchi parametr raqamli port raqamidir. Port 0 dan 65535 gacha bo‘lgan 2 baytlik qiymatni ifodalaydi. Bir xil manzilda (bir xil mashinada) bir nechta turli tarmoq ilovalari ishga tushirilishi mumkinligi sababli, port ushbu ilovalarni farqlash imkonini beradi. Masalan:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

Bu yerda server 12345-portda ishlaydi. E’tibor bering, barcha portlar ham bepul bo‘lmasligi mumkin. Ammo, qoida tariqasida, band bo‘lgan portlar unchalik ko‘p emas.

Manzil sifatida joriy xost nomidan foydalanamiz. Uni olish uchun funktsiyadan foydalaniladi **socket.gethostname()** (odatda bu joriy kompyuterning nomi).

**Ulanishlarni tinglash**

Serverni bog‘lagandan so‘ng ulanishlarni tinglash uchun uni ishga tushirish kerak. Buning uchun **listen()** metodi qo‘llaniladi.

socket.listen([backlog])

Bu backlog parametrini qabul qiladi. backlog – bu socket uchun ruxsat etilgan navbatdagi kiruvchi ulanishlarning maksimal soni. Ya’ni, mijozlar ulanganda, ular navbatda turishadi va server joriy mijozni qayta ishlaguncha kutishadi. Agar navbatda server tomonidan qayta ishlanishini kutayotgan mijozlar soni allaqachon belgilangan bo‘lsa, barcha yangi mijozlar rad etiladi. Masalan:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

**Mijozni qabul qilish va qayta ishlash**

**accept()** metodi kiruvchi ulanishlarni qabul qilish uchun ishlatiladi . Bu usul 2 ta tuple tipli qiymatni qaytaradi

(conn, address)

Birinchi element conn - server mijoz bilan aloqa qiladigan boshqa soket ob’ektini ifodalaydi. Ikkinchi element address - ulangan mijozning manzili hisoblanadi. Shuni ta’kidlash kerakki, mijoz bilan o‘zaro aloqalar tugagandan so‘ng, conn socketini close() usuli bilan yopish kerak bo‘ladi.

Tuplening birinchi elementi - conn yordamida mijozga xabar yuborish yoki aksincha ma’lumotlarni qabul qilish mumkin. Soketdan ma’lumotlarni qabul qilish uchun **socket.recv()** usuli qo‘llaniladi.

bytes = socket.recv(bufsize)

Bu zarur parametr sifatida bir vaqtning o‘zida boshqa socketdan olinishi mumkin bo‘lgan baytlardagi maksimal bufer hajmini oladi. Qaytish qiymati boshqa socketdan olingan baytlar to‘plamidir.

Ma’lumotlarni yuborish uchun **socket.send()** metodi qo‘llaniladi, u parametr sifatida yuboriladigan ma’lumotlar to‘plamini oladi.

socket.send(bytes)

Endi bir misolni ko‘rib chiqaylik. Server.py faylida serverni quyidagi kod bilan aniqlaylik:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

print("Server starts")

con, addr = server.accept()

print("connection: ", conn)

print("client address: ", addr)

message = "Hello Client!"

con.send(message.encode())

con.close()

print("Server ends")

server.close()

Bu yerda server mijozni qabul qiladi, uning ulanishi haqidagi ma’lumotlarni chiqaradi va mijozga “Hello Client!” qatorini qaytarib yuboradi.

Va client.py faylida quyidagi mijoz kodini aniqlanadi:

import socket

client = socket.socket()

hostname = socket.gethostname()

port = 12345

client.connect((hostname, port))

data = client.recv(1024)

print("Server sent: ", data.decode())

client.close()

Bizning serverimiz va mijozimiz bitta kompyuterda ishlaganligi sababli, **socket.gethostname()** server kodidagi kabi ulanish uchun server manzilini aniqlash uchun funksiya va port 12345 ishlatiladi. Serverga ulangandan so‘ng undan ma’lumotlarni olinadi va uni konsolda ko‘rsatadi.

Avval server kodini ishga tushiraylik. Va keyin mijoz kodini ishga tushiriladi. Natijada, server ulanishni qabul qiladi, u haqidagi ma’lumotlarni konsolda ko‘rsatadi va mijozga xabar yuboradi:

c:\python>python server.py

Server starts

connection: <socket.socket fd=432, family=2, type=1, proto=0, laddr=(‘192.168.0.102’, 12345), raddr=(‘192.168.0.102’, 61824)>

client address: (‘192.168.0.102’, 61824)

Server ends

Xususan, bu yerda server va mijoz 192.168.0.102 da ishlayotganini va mijoz 61824 portidan foydalanayotganini ko‘ramiz.

Va mijoz serverdan xabar oladi va uni konsolga chop etadi:

c:\python>python client.py

Server sent: Hello Client!

**Bir nechta mijozlar bilan ishlash**

Bunday holda, server bitta mijozga xizmat qiladi va ishlashni to‘xtatadi. Agar biz server ko‘p mijozlar bilan ishlashni istasak, unda cheksiz tsikldan foydalanishga to‘g‘ri keladi:

import socket

from datetime import datetime

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

print("Server running")

while True:

con, addr = server.accept()

print("client address: ", addr)

message = datetime.now().strftime("%H:%M:%S")

con.send(message.encode())

con.close()

Bunday holda, masalan, o‘rnatilgan modul datetime va datetime.now().strftime() funksiyasidan foydalanib, joriy vaqtni satr sifatida olish, keyinchalik uni mijozga yuborish mumkin. Natijada, mijoz so‘rov bilan joriy vaqthaqidagi ma’lumotlarni ham oladi.

**Ikki tomonlama aloqa**

Yuqoridagi misollarda aloqa bir tomonlama edi - server ma’lumotlarni yuboradi va mijoz uni qabul qiladi. Keling, mijoz ham, server ham ma’lumotlarni jo‘natgan va qabul qilganda eng oddiy ikki tomonlama aloqani ko‘rib chiqaylik. Serverga mijozdan bir nechta satr olishiga ruxsat beraylik, uni o‘zgartirib so‘ngra mijozga qaytarib yuboraylik:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

print("Server running")

while True:

con, \_ = server.accept()

data = con.recv(1024)

message = data.decode()

print(f"Client sent: {message}")

message = message[::-1]

con.send(message.encode())

con.close()

Mijoz konsoldan satr kiritish va uni serverga yuborish uchun kodni aniqlasin:

import socket

client = socket.socket()

hostname = socket.gethostname()

port = 12345

client.connect((hostname, port))

message = input("Input a text: ")

client.send(message.encode())

data = client.recv(1024)

print("Server sent: ", data.decode())

client.close()

Yuqoridagi ishning natijasi:

**Client**:

c:\python>python client.py

Input a text: hello

Server sent: olleh

c:\python>

**Server**:

c:\python>python server.py

Server running

Client sent: hello

**Nazorat savollari:**

1. Server bilan aloqani qanday amalga oshiriladi?
2. socket.bind() metodi nima amalni bajaradi?
3. socket.gethostname() metodi qanday vazifani bajaradi?
4. listen() metodining ishlash tamoyili qanday?
5. socket.recv() metodidan nima maqsadda foydalaniladi?
6. Socket nima?
7. decode() funksiyasi nima maqsadda foydalaniladi?
8. Serverga ma’lumot jo‘natish qanday amalga oshiriladi?
9. Serverdan ma’lumotni qanday qabul qilin olinadi?
10. Connect() metodi qanday parametrlarni qabul qiladi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**4-mashg‘ulot.** TCP klient-server dasturini testlash.

O‘quv savollari:

1. TCP client-server dasturi yordamida ma’lumot almashish.

**1. TCP client-server dasturi yordamida ma’lumot almashish.**

**Serverni ulanish**

Server kiruvchi ulanishlarni kutib turadi (listining), ularni qandaydir tarzda qayta ishlaydi va javob yuboradi. Server o‘z ishini boshlashi uchun, avvalo, u ulanishlarni tinglaydigan manzilni aniqlab olishi kerak. Buning uchun **bind()** metodi qo‘llaniladi.

socket.bind(address)

Bu usul server ishga tushadigan manzilni qabul qiladi. Odatiy bo‘lib, manzil ikki elementdan iborat to‘plamdir:

(host, port)

Birinchi element string sifatida xost hisoblanadi. Bu, masalan, "127.0.0.1" shaklidagi IP-manzil yoki mahalliy xost nomi bo‘lishi mumkin. Ikkinchi parametr raqamli port raqamidir. Port 0 dan 65535 gacha bo‘lgan 2 baytlik qiymatni ifodalaydi. Bir xil manzilda (bir xil mashinada) bir nechta turli tarmoq ilovalari ishga tushirilishi mumkinligi sababli, port ushbu ilovalarni farqlash imkonini beradi. Masalan:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

Bu yerda server 12345-portda ishlaydi. E’tibor bering, barcha portlar ham bepul bo‘lmasligi mumkin. Ammo, qoida tariqasida, band bo‘lgan portlar unchalik ko‘p emas.

Manzil sifatida joriy xost nomidan foydalanamiz. Uni olish uchun funktsiyadan foydalaniladi **socket.gethostname()** (odatda bu joriy kompyuterning nomi).

**Ulanishlarni tinglash**

Serverni bog‘lagandan so‘ng ulanishlarni tinglash uchun uni ishga tushirish kerak. Buning uchun **listen()** metodi qo‘llaniladi.

socket.listen([backlog])

Bu backlog parametrini qabul qiladi. backlog – bu socket uchun ruxsat etilgan navbatdagi kiruvchi ulanishlarning maksimal soni. Ya’ni, mijozlar ulanganda, ular navbatda turishadi va server joriy mijozni qayta ishlaguncha kutishadi. Agar navbatda server tomonidan qayta ishlanishini kutayotgan mijozlar soni allaqachon belgilangan bo‘lsa, barcha yangi mijozlar rad etiladi. Masalan:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

**Mijozni qabul qilish va qayta ishlash**

**accept()** metodi kiruvchi ulanishlarni qabul qilish uchun ishlatiladi . Bu usul 2 ta tuple tipli qiymatni qaytaradi

(conn, address)

Birinchi element conn - server mijoz bilan aloqa qiladigan boshqa soket ob’ektini ifodalaydi. Ikkinchi element address - ulangan mijozning manzili hisoblanadi. Shuni ta’kidlash kerakki, mijoz bilan o‘zaro aloqalar tugagandan so‘ng, conn socketini close() usuli bilan yopish kerak bo‘ladi.

Tuplening birinchi elementi - conn yordamida mijozga xabar yuborish yoki aksincha ma’lumotlarni qabul qilish mumkin. Soketdan ma’lumotlarni qabul qilish uchun **socket.recv()** usuli qo‘llaniladi.

bytes = socket.recv(bufsize)

Bu zarur parametr sifatida bir vaqtning o‘zida boshqa socketdan olinishi mumkin bo‘lgan baytlardagi maksimal bufer hajmini oladi. Qaytish qiymati boshqa socketdan olingan baytlar to‘plamidir.

Ma’lumotlarni yuborish uchun **socket.send()** metodi qo‘llaniladi, u parametr sifatida yuboriladigan ma’lumotlar to‘plamini oladi.

socket.send(bytes)

Endi bir misolni ko‘rib chiqaylik. Server.py faylida serverni quyidagi kod bilan aniqlaylik:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

print("Server starts")

con, addr = server.accept()

print("connection: ", conn)

print("client address: ", addr)

message = "Hello Client!"

con.send(message.encode())

con.close()

print("Server ends")

server.close()

Bu yerda server mijozni qabul qiladi, uning ulanishi haqidagi ma’lumotlarni chiqaradi va mijozga “Hello Client!” qatorini qaytarib yuboradi.

Va client.py faylida quyidagi mijoz kodini aniqlanadi:

import socket

client = socket.socket()

hostname = socket.gethostname()

port = 12345

client.connect((hostname, port))

data = client.recv(1024)

print("Server sent: ", data.decode())

client.close()

Bizning serverimiz va mijozimiz bitta kompyuterda ishlaganligi sababli, **socket.gethostname()** server kodidagi kabi ulanish uchun server manzilini aniqlash uchun funksiya va port 12345 ishlatiladi. Serverga ulangandan so‘ng undan ma’lumotlarni olinadi va uni konsolda ko‘rsatadi.

Avval server kodini ishga tushiraylik. Va keyin mijoz kodini ishga tushiriladi. Natijada, server ulanishni qabul qiladi, u haqidagi ma’lumotlarni konsolda ko‘rsatadi va mijozga xabar yuboradi:

c:\python>python server.py

Server starts

connection: <socket.socket fd=432, family=2, type=1, proto=0, laddr=(‘192.168.0.102’, 12345), raddr=(‘192.168.0.102’, 61824)>

client address: (‘192.168.0.102’, 61824)

Server ends

Xususan, bu yerda server va mijoz 192.168.0.102 da ishlayotganini va mijoz 61824 portidan foydalanayotganini ko‘ramiz.

Va mijoz serverdan xabar oladi va uni konsolga chop etadi:

c:\python>python client.py

Server sent: Hello Client!

**Bir nechta mijozlar bilan ishlash**

Bunday holda, server bitta mijozga xizmat qiladi va ishlashni to‘xtatadi. Agar biz server ko‘p mijozlar bilan ishlashni istasak, unda cheksiz tsikldan foydalanishga to‘g‘ri keladi:

import socket

from datetime import datetime

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

print("Server running")

while True:

con, addr = server.accept()

print("client address: ", addr)

message = datetime.now().strftime("%H:%M:%S")

con.send(message.encode())

con.close()

Bunday holda, masalan, o‘rnatilgan modul datetime va datetime.now().strftime() funksiyasidan foydalanib, joriy vaqtni satr sifatida olish, keyinchalik uni mijozga yuborish mumkin. Natijada, mijoz so‘rov bilan joriy vaqthaqidagi ma’lumotlarni ham oladi.

**Ikki tomonlama aloqa**

Yuqoridagi misollarda aloqa bir tomonlama edi - server ma’lumotlarni yuboradi va mijoz uni qabul qiladi. Keling, mijoz ham, server ham ma’lumotlarni jo‘natgan va qabul qilganda eng oddiy ikki tomonlama aloqani ko‘rib chiqaylik. Serverga mijozdan bir nechta satr olishiga ruxsat beraylik, uni o‘zgartirib so‘ngra mijozga qaytarib yuboraylik:

import socket

server = socket.socket()

hostname = socket.gethostname()

port = 12345

server.bind((hostname, port))

server.listen(5)

print("Server running")

while True:

con, \_ = server.accept()

data = con.recv(1024)

message = data.decode()

print(f"Client sent: {message}")

message = message[::-1]

con.send(message.encode())

con.close()

Mijoz konsoldan satr kiritish va uni serverga yuborish uchun kodni aniqlasin:

import socket

client = socket.socket()

hostname = socket.gethostname()

port = 12345

client.connect((hostname, port))

message = input("Input a text: ")

client.send(message.encode())

data = client.recv(1024)

print("Server sent: ", data.decode())

client.close()

Yuqoridagi ishning natijasi:

**Client**:

c:\python>python client.py

Input a text: hello

Server sent: olleh

c:\python>

**Server**:

c:\python>python server.py

Server running

Client sent: hello

**Nazorat savollari:**

1. Server bilan aloqani qanday amalga oshiriladi?
2. socket.bind() metodi nima amalni bajaradi?
3. socket.gethostname() metodi qanday vazifani bajaradi?
4. listen() metodining ishlash tamoyili qanday?
5. socket.recv() metodidan nima maqsadda foydalaniladi?

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**5-mashg‘ulot.** PyQt5 paketidan foydalanib zamonaviy chat dasturini tuzish.

**O‘quv savollari:**

1. TCP dasturini client qismini GUI ko‘rinishda ishlab chiqish.

**1. TCP dasturini client qismini GUI ko‘rinishda ishlab chiqish.**

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Tue Jul 24 13:05:46 2018

@author: JC

"""

import socket

import sys

import threading

import time

import functools

from PyQt5 import QtCore, QtGui

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QMainWindow, QApplication, QWidget, QPushButton

from PyQt5.QtWidgets import QVBoxLayout, QHBoxLayout, QMessageBox, QTabWidget

from PyQt5.QtWidgets import QGridLayout, QScrollArea, QLabel, QListView

from PyQt5.QtWidgets import QLineEdit, QComboBox, QGroupBox, QAction

from PyQt5.QtGui import QStandardItemModel, QStandardItem, QFont

class MyTableWidget(QWidget):

def \_\_init\_\_(self, parent):

super(QWidget, self).\_\_init\_\_(parent)

#connecion

self.conn = socket.socket()

self.connected = False

#tab UI

self.layout = QVBoxLayout(self)

self.tabs = QTabWidget()

self.tabs.resize(300,200)

self.tab1 = QWidget()

self.tab2 = QWidget()

self.tabs.addTab(self.tab1, "Home")

self.tabs.addTab(self.tab2, "Chat Room")

self.tabs.setTabEnabled(1,False)

#<Home>

gridHome = QGridLayout()

self.tab1.setLayout(gridHome)

self.IPBox = QGroupBox("IP")

self.IPLineEdit = QLineEdit()

self.IPLineEdit.setText("127.0.0.1")

IPBoxLayout = QVBoxLayout()

IPBoxLayout.addWidget(self.IPLineEdit)

self.IPBox.setLayout(IPBoxLayout)

self.portBox = QGroupBox("port")

self.portLineEdit = QLineEdit()

self.portLineEdit.setText("33002")

portBoxLayout = QVBoxLayout()

portBoxLayout.addWidget(self.portLineEdit)

self.portBox.setLayout(portBoxLayout)

self.nameBox = QGroupBox("Name")

self.nameLineEdit = QtWidgets.QLineEdit()

nameBoxLayout = QVBoxLayout()

nameBoxLayout.addWidget(self.nameLineEdit)

self.nameBox.setLayout(nameBoxLayout)

self.connStatus = QLabel("Status", self)

font = QFont()

font.setPointSize(16)

self.connStatus.setFont(font)

self.connBtn = QPushButton("Connect")

self.connBtn.clicked.connect(self.connect\_server)

self.disconnBtn = QPushButton("Disconnect")

self.disconnBtn.clicked.connect(self.disconnect\_server)

gridHome.addWidget(self.IPBox,0,0,1,1)

gridHome.addWidget(self.portBox,0,1,1,1)

gridHome.addWidget(self.nameBox,1,0,1,1)

gridHome.addWidget(self.connStatus,1,1,1,1)

gridHome.addWidget(self.connBtn,2,0,1,1)

gridHome.addWidget(self.disconnBtn,2,1,1,1)

gridHome.setColumnStretch(0, 1)

gridHome.setColumnStretch(1, 1)

gridHome.setRowStretch(0, 0)

gridHome.setRowStretch(1, 0)

gridHome.setRowStretch(2, 9)

#</Home>

#<Chat Room>

gridChatRoom = QGridLayout()

self.tab2.setLayout(gridChatRoom)

self.messageRecords = QLabel("<font color=\"#000000\">Welcome to chat room</font>", self)

self.messageRecords.setStyleSheet("background-color: white;");

self.messageRecords.setAlignment(QtCore.Qt.AlignTop)

self.messageRecords.setAutoFillBackground(True);

self.scrollRecords = QScrollArea()

self.scrollRecords.setWidget(self.messageRecords)

self.scrollRecords.setWidgetResizable(True)

self.sendTo = "ALL"

self.sendChoice = QLabel("Send to :ALL", self)

self.sendComboBox = QComboBox(self)

self.sendComboBox.addItem("ALL")

self.sendComboBox.activated[str].connect(self.send\_choice)

self.lineEdit = QLineEdit()

self.lineEnterBtn = QPushButton("Enter")

self.lineEnterBtn.clicked.connect(self.enter\_line)

self.lineEdit.returnPressed.connect(self.enter\_line)

self.friendList = QListView()

self.friendList.setWindowTitle('Room List')

self.model = QStandardItemModel(self.friendList)

self.friendList.setModel(self.model)

self.emojiBox = QGroupBox("Emoji")

self.emojiBtn1 = QPushButton("ก็ʕ•͡ᴥ•ʔ ก้")

self.emojiBtn1.clicked.connect(functools.partial(self.send\_emoji, "ก็ʕ•͡ᴥ•ʔ ก้"))

self.emojiBtn2 = QPushButton("(｡◕∀◕｡)")

self.emojiBtn2.clicked.connect(functools.partial(self.send\_emoji, "(｡◕∀◕｡)"))

self.emojiBtn3 = QPushButton("( ˘･з･)")

self.emojiBtn3.clicked.connect(functools.partial(self.send\_emoji, "( ˘･з･)"))

self.emojiBtn4 = QPushButton("ᕦ(ò\_óˇ)ᕤ")

self.emojiBtn4.clicked.connect(functools.partial(self.send\_emoji, "ᕦ(ò\_óˇ)ᕤ"))

emojiLayout = QHBoxLayout()

emojiLayout.addWidget(self.emojiBtn1)

emojiLayout.addWidget(self.emojiBtn2)

emojiLayout.addWidget(self.emojiBtn3)

emojiLayout.addWidget(self.emojiBtn4)

self.emojiBox.setLayout(emojiLayout)

gridChatRoom.addWidget(self.scrollRecords,0,0,1,3)

gridChatRoom.addWidget(self.friendList,0,3,1,1)

gridChatRoom.addWidget(self.sendComboBox,1,0,1,1)

gridChatRoom.addWidget(self.sendChoice,1,2,1,1)

gridChatRoom.addWidget(self.lineEdit,2,0,1,3)

gridChatRoom.addWidget(self.lineEnterBtn,2,3,1,1)

gridChatRoom.addWidget(self.emojiBox,3,0,1,4)

gridChatRoom.setColumnStretch(0, 9)

gridChatRoom.setColumnStretch(1, 9)

gridChatRoom.setColumnStretch(2, 9)

gridChatRoom.setColumnStretch(3, 1)

gridChatRoom.setRowStretch(0, 9)

#</Chat Room>

#Initialization

self.layout.addWidget(self.tabs)

self.setLayout(self.layout)

def enter\_line(self):

#assure the person still in rooom before send out

if self.sendTo != self.sendComboBox.currentText():

self.message\_display\_append("The person left. Private message not delivered")

self.lineEdit.clear()

return

line = self.lineEdit.text()

if line == "":#prevent empty message

return

if self.sendTo != "ALL":#private message, send to myself first

#this is a trick leverage the server sending back a copy to myself

send\_msg = bytes("{"+self.userName+"}"+line, "utf-8")

self.conn.send(send\_msg)

time.sleep(0.1) #this is important for not overlapping two sending

send\_msg = bytes("{"+self.sendTo+"}"+line, "utf-8")

self.conn.send(send\_msg)

self.lineEdit.clear()

self.scrollRecords.verticalScrollBar().setValue(self.scrollRecords.verticalScrollBar().maximum())

def send\_emoji(self, emoji):

#assure the person still in rooom before send out

if self.sendTo != self.sendComboBox.currentText():

self.message\_display\_append("The person left. Private message not delivered")

return

if self.sendTo != "ALL":#private message, send to myself first

#this is a trick leverage the server sending back a copy to myself

send\_msg = bytes("{"+self.userName+"}"+emoji, "utf-8")

self.conn.send(send\_msg)

time.sleep(0.1) #this is important for not overlapping two sending

send\_msg = bytes("{"+self.sendTo+"}"+emoji, "utf-8")

self.conn.send(send\_msg)

def message\_display\_append(self, newMessage, textColor = "#000000"):

oldText = self.messageRecords.text()

appendText = oldText+"<br /><font color=\""+textColor+"\">"+newMessage+"</font><font color=\"#000000\"></font>"

self.messageRecords.setText(appendText)

time.sleep(0.2) #this helps the bar set to bottom, after all message already appended

self.scrollRecords.verticalScrollBar().setValue(self.scrollRecords.verticalScrollBar().maximum())

def updateRoom(self):

while self.connected:

data = self.conn.recv(1024)

data = data.decode("utf-8")

print(data)

if data != "":

if "{CLIENTS}" in data:

welcome = data.split("{CLIENTS}")

self.update\_send\_to\_list(welcome[1])

self.update\_room\_list(welcome[1])

if not welcome[0][5:] == "":

self.message\_display\_append(welcome[0][5:])

self.scrollRecords.verticalScrollBar().setValue(self.scrollRecords.verticalScrollBar().maximum())

elif data[:5] == "{MSG}": #{MSG} includes broadcast and server msg

self.message\_display\_append(data[5:], "#006600")

self.scrollRecords.verticalScrollBar().setValue(self.scrollRecords.verticalScrollBar().maximum())

else: #private messgage is NONE format

self.message\_display\_append("{private}"+data, "#cc33cc")

self.scrollRecords.verticalScrollBar().setValue(self.scrollRecords.verticalScrollBar().maximum())

time.sleep(0.1) #this is for saving thread cycle time

def connect\_server(self):

if self.connected == True:

return

name = self.nameLineEdit.text()

if name == "":

self.connStatus.setText("Status :"+"Please enter your name")

return

self.userName = name

IP = self.IPLineEdit.text()

if IP == "":

IP = "127.0.0.1"

port = self.portLineEdit.text()

if port == "" or not port.isnumeric():

self.portLineEdit.setText("33002")

self.connStatus.setText("Status :"+"Port format invalid")

return

else:

port = int(port)

try:

self.conn.connect((IP, port))

except:

self.connStatus.setText("Status :"+" Refused")

self.conn = socket.socket()

return

send\_msg = bytes("{REGISTER}"+name, "utf-8")

self.conn.send(send\_msg)

self.connected = True

self.connStatus.setText("Status :"+" Connected")

self.nameLineEdit.setReadOnly(True) #This setting is not functional well

self.tabs.setTabEnabled(1,True)

self.rT = threading.Thread(target= self.updateRoom)

self.rT.start()

def disconnect\_server(self):

if self.connected == False:

return

send\_msg = bytes("{QUIT}", "utf-8")

self.conn.send(send\_msg)

self.connStatus.setText("Status :"+" Disconnected")

self.nameLineEdit.setReadOnly(False)

self.nameLineEdit.clear()

self.tabs.setTabEnabled(1,False)

self.connected = False

self.rT.join()

self.conn.close()

self.conn = socket.socket()

def update\_room\_list(self, strList):

L = strList.split("|")

self.model.clear()

for person in L:

item = QStandardItem(person)

item.setCheckable(False)

self.model.appendRow(item)

def update\_send\_to\_list(self, strList):

L = strList.split("|")

self.sendComboBox.clear()

self.sendComboBox.addItem("ALL")

for person in L:

if person != self.userName:

self.sendComboBox.addItem(person)

previous = self.sendTo

index = self.sendComboBox.findText(previous)

print("previous choice:",index)

if index != -1:

self.sendComboBox.setCurrentIndex(index) #updating, maintain receiver

else:

self.sendComboBox.setCurrentIndex(0) #updating, the receiver left, deafault to "ALL"

def send\_choice(self,text):

self.sendTo = text

print(self.sendTo)

self.sendChoice.setText("Send to: "+text)

class Window(QMainWindow):

def \_\_init\_\_(self):

super(Window, self).\_\_init\_\_()

self.setGeometry(50, 50, 500, 300)

self.setWindowTitle("Chat-Client")

self.table\_widget = MyTableWidget(self)

self.setCentralWidget(self.table\_widget)

self.show()

def closeEvent(self, event):

close = QMessageBox()

close.setText("You sure?")

close.setStandardButtons(QMessageBox.Yes | QMessageBox.Cancel)

close = close.exec()

if close == QMessageBox.Yes:

self.table\_widget.disconnect\_server() #disconnect to server before exit

event.accept()

else:

event.ignore()

def run():

app = QApplication(sys.argv)

GUI = Window()

sys.exit(app.exec\_())

if \_\_name\_\_ == "\_\_main\_\_":

run()

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**6-mashg‘ulot.** Pythonda GUI paketidan foydalanib chat dasturini tuzishni yakunlash

**O‘quv savollari:**

1. TCP client-server dasturini GUI ko‘rinishda ishlab chiqish.

**1. TCP client-server dasturini GUI ko‘rinishda ishlab chiqish.**

import argparse

from socket import AF\_INET, socket, SOCK\_STREAM

from threading import Thread

def accept\_incoming\_connections():

"""Sets up handling for incoming clients."""

while True:

client, client\_address = SERVER.accept()

print("%s:%s has connected." % client\_address)

addresses[client] = client\_address

Thread(target=handle\_client, args=(client,)).start()

def handle\_client(client): # Takes client socket as argument.

"""Handles a single client connection."""

name = ""

prefix = ""

while True:

msg = client.recv(BUFSIZ)

if not msg is None:

msg = msg.decode("utf-8")

if msg == "":

msg = "{QUIT}"

# Avoid messages before registering

if msg.startswith("{ALL}") and name:

new\_msg = msg.replace("{ALL}", "{MSG}"+prefix)

send\_message(new\_msg, broadcast=True)

continue

if msg.startswith("{REGISTER}"):

name = msg.split("}")[1]

welcome = '{MSG}Welcome %s!' % name

send\_message(welcome, destination=client)

msg = "{MSG}%s has joined the chat!" % name

send\_message(msg, broadcast=True)

clients[client] = name

prefix = name + ": "

send\_clients()

continue

if msg == "{QUIT}":

client.close()

try:

del clients[client]

except KeyError:

pass

if name:

send\_message("{MSG}%s has left the chat." % name, broadcast=True)

send\_clients()

break

# Avoid messages before registering

if not name:

continue

# We got until this point, it is either an unknown message or for an

# specific client...

try:

msg\_params = msg.split("}")

dest\_name = msg\_params[0][1:] # Remove the {

dest\_sock = find\_client\_socket(dest\_name)

if dest\_sock:

send\_message(msg\_params[1], prefix=prefix, destination=dest\_sock)

else:

print("Invalid Destination. %s" % dest\_name)

except:

print("Error parsing the message: %s" % msg)

def send\_clients():

send\_message("{CLIENTS}" + get\_clients\_names(), broadcast=True)

def get\_clients\_names(separator="|"):

names = []

for \_, name in clients.items():

names.append(name)

return separator.join(names)

def find\_client\_socket(name):

for cli\_sock, cli\_name in clients.items():

if cli\_name == name:

return cli\_sock

return None

def send\_message(msg, prefix="", destination=None, broadcast=False):

send\_msg = bytes(prefix + msg, "utf-8")

if broadcast:

"""Broadcasts a message to all the clients."""

for sock in clients:

sock.send(send\_msg)

else:

if destination is not None:

destination.send(send\_msg)

clients = {}

addresses = {}

parser = argparse.ArgumentParser(description="Chat Server")

parser.add\_argument(

'--host',

help='Host IP',

default="127.0.0.1"

)

parser.add\_argument(

'--port',

help='Port Number',

default=33002

)

server\_args = parser.parse\_args()

HOST = server\_args.host

PORT = int(server\_args.port)

BUFSIZ = 2048

ADDR = (HOST, PORT)

stop\_server = False

SERVER = socket(AF\_INET, SOCK\_STREAM)

SERVER.bind(ADDR)

if \_\_name\_\_ == "\_\_main\_\_":

try:

SERVER.listen(5)

print("Server Started at {}:{}".format(HOST, PORT))

print("Waiting for connection...")

ACCEPT\_THREAD = Thread(target=accept\_incoming\_connections)

ACCEPT\_THREAD.start()

ACCEPT\_THREAD.join()

SERVER.close()

except KeyboardInterrupt:

print("Closing...")

ACCEPT\_THREAD.interrupt()

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**7-mashg‘ulot.** PyQt5 paketidan foydalanib zamonaviy chat dasturini tuzish.

**O‘quv savollari:**

1. GUI TCP klient-server dasturi yordamida ma’lumot almashinuvini testlash.

**1. GUI TCP klient-server dasturi yordamida ma’lumot almashinuvini testlash.**

Bu darsda oldingi mavzulardagi GUI TCP klient-server dasturini ishga tushirib, kursantlar bir-birlari bilan chat orqali suhbatlashishadi, hamda ushbu dasturni testlashadi.

**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**3-Mavzu:** Pythonda tarmoq dasturlashga kirish.

**8-mashg‘ulot.** **Python ilovasini kompilyatsiya qilish**.

O‘quv savollari:

1. Python ilovasini kompilyatsiya qilish.

**1. Python ilovasini kompilyatsiya qilish.**

Ushbu bo‘limda pythonda yaratilgan loyihalarning yuklanuvchi faylini hosil qilish bo‘yicha bilim va ko‘nikmalarga berib o‘tiladi.

Dasturchi yaratgan dasturini kimgadir taqdim etishi uchun albatda butun bir loyihasini emas abalki butun loyihani bitta yuklanuvchi faylga birlashtirib, keyin taqdim etadi. Bunday bitta faylga birlashtirishning turli usullari mavjud.

**Pythonning PYINSTALLER paketi va uning imkoniyatlari**

*PyInstaller* paketi yordamida yuklanuvchi faylni hosil qilishni bir nechta qadamlarga bo‘lish mumkin:

**1-qadam: PyInstaller paketini kompyuterga o‘rnatish.**

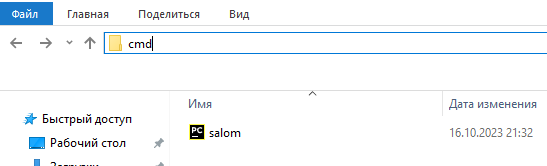
*PyInstaller* paketini komputerga o‘rnatish uchun quyidagi buyruqni buyruklar satriga kiritiladi:

pip install pyinstaller

Shundan so‘ng ushbu paketdan fodalanish mumkin bo‘ladi.

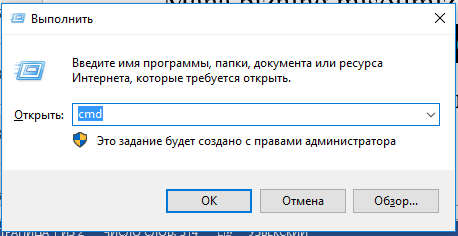
**2-qadam: PyInstaller yordamida bajariladigan fayl hosil qilish.**

Endi PyInstaller paketi yordamida Python tilida yozilgan loyihani bitta faylga jamlash kerak. Buning uchun project joylashgan papkaga kirish va buyruq qatoriga **CMD** buyrug‘ini kiritib, enter tugmasini bosiladi:



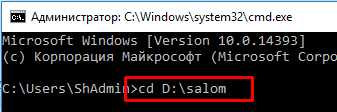
*80-rasm*. *Buyruq qatoriga CMD buyrug‘ini kiritish*

Yoki ikkinchi usuli kompyuteringinzdan PUSK+R tugmasi bosiladi va hosil bo‘lgan «**Выполнить»** oynasiga “CMD” buyrug‘i kiritiladi.



*81-rasm. «Выполнить»**oynasi*

Va hosil bo‘lgan konsol oynasiga **cd** buyrug‘i orqali kerakli papkaga kirish mumkin. Buning uchun **cd** va undan keyin Python skriptingiz saqlanadigan joy quyidagi ko‘rinishda kiritiladi va enter tugmasi bosiladi va ko‘rsatilgan kotalokga kiriladi:



*82-rasm*. ***cd*** *buyrug‘i orqali kerakli papkaga kirish*

Keyin bajariladigan amal faylni yaratish uchun quyidagi shablondan to‘g‘ri foydalanish qilishdan iborat:

pyinstaller --onefile pythonFileName.py

Ushbu shablonda *pythonFileName* "**salom**" (va fayl kengaytmasi albatta .py**)** bo‘lgani uchun bajariladigan faylni yaratish buyrug‘i quyidagicha bo‘ladi:

pyinstaller --onefile salom.py

ENTER tugmasini bosiladi va pyinstaller paketi avtomatik ravishda ushbu papka ichida dist papkasini hosil qiladi. Yuklanuvchi exe fayl esa ushbu papkaning ichida joylashgan bo‘ladi. Endi ushbu exe faylni python dasturi o‘rnatilmagan kompyuterlarda ham ishga tushirish mumkin.

**Pythonning *AUTO-PY-TO-EXE* paketi va uning imkoniyatlari**

Pythonda yaratilgan loyihalarni yagona .exe faylga yig‘ishning yana bir usuli – bu auto-py-to-exe paketidan foydalangan holda judayam funksiyanal va visual tarzda amalga oshirish mumkin. Buning uchun ushbu paketni kompyuterga o‘rnatish kerak bo‘ladi.

**auto-py-to-exe paketini kompyuterga o‘rnatish.** Kompyuterga python paketlarini o‘rnatishning bir nechta usullari mavjud. Bullardan **pip**, **github** hamda **ofline** holda o‘rnatish mumkin. Odatda pip orqali o‘rnatiladi. Chunki bu usul qulay hisoblanadi.

**Pip orqali o‘rnatish.** *Auto-py-to-exe*paketini kompyuterga Pip orqali o‘rnatish quyidagicha amalga oshiriladi: Birinchi navbatda **“Команда строка”** oynasi ochib olinadi va u joyga pip install auto-py-to-exe buyrug‘i kiritiladi va enter tugmasi bosiladi. Shundan so‘ng avtomatik ravishda kompyuterga ushbu paket o‘rnatiladi:

$ pip install auto-py-to-exe

**GitHubdan o‘rnatish.** Ushbu paketnito‘g‘ridan-to‘g‘ri GitHubdan o‘rnatish mumkin. GitHubdan *auto-py-to-exe* paketini o‘rnatish uchun avval GitHub omborini klonlash kerak bo‘ladi. U quyidagi kod orqali amalga oshiriladi:

git **clone** https://github.com/brentvollebregt/auto-py-to-exe.git

Keyin auto-py-to-exe papkasiga cd buyrug‘i orqali o‘tish kerak.

$ cd auto-py-to-exe

Endi python buyrug‘i orqali setup.py faylini ishga tushirish kerak.

$ python setup.py install

Quyidagi buyruq yordamida versiyani ham tekshirish mumkin:

auto-py-to-exe --version



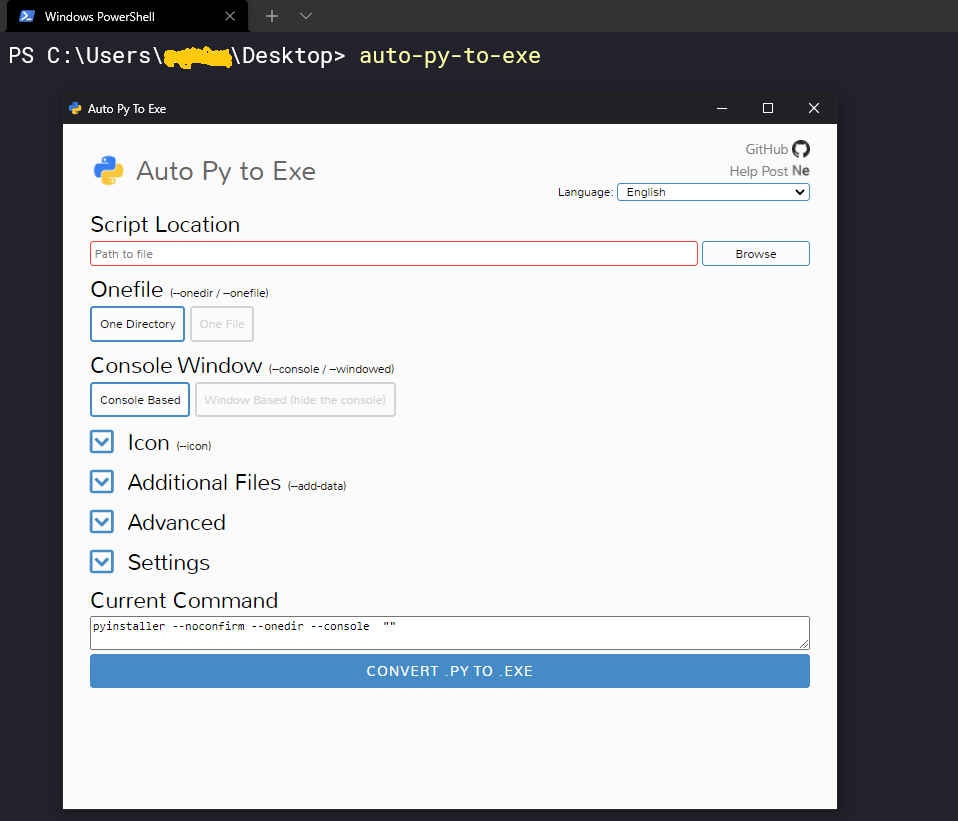
*83-rasm. Paketning versiyasini bilish*

Auto py to exe ning joriy versiyasi 2.42.0 va u endi kompyuterga o‘rnatilgan.

**Ilovani ishga tushirish.** auto-py-to-exe ilivasini ishga tushirish uchun terminalda quyidagi buyruqni bajarish kerak bo‘ladi:

$ auto-py-to-exe

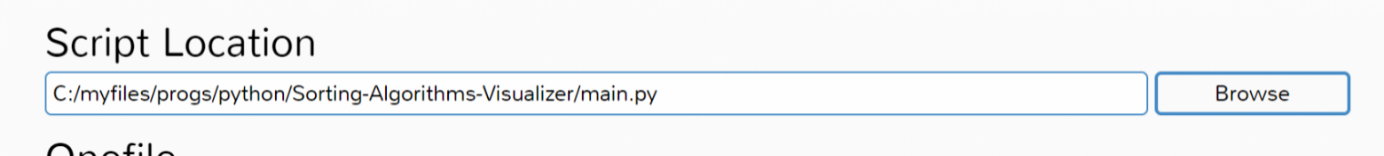
Shundan so‘ng quyidagi “Auto py to exe” dasturi ishga tushadi:



*84-rasm.* ***avto-py-to-exe GUI interfeysi***

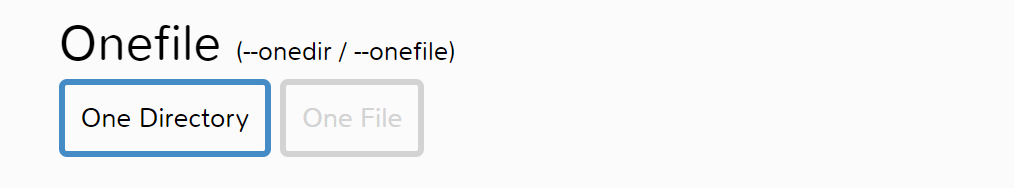
Endi ushbu interfeysdan .py faylni yuklanuvchi .exe fayliga aylantirish uchun foydalaniladi.

**Konvertatsiya jarayoni. 1-qadam**. **Fayl joylashuvini qo‘shish.** Python faylini .exe fayliga o‘zgartirish uchun avval uning yo‘lini ko‘rsatish kerak bo‘ladi. Buning uchun konvertatsiya qilinadigan faylning manziliga o‘tiladi va keyin yo‘lni ko‘rsatib, qo‘shib qo‘yiladi:



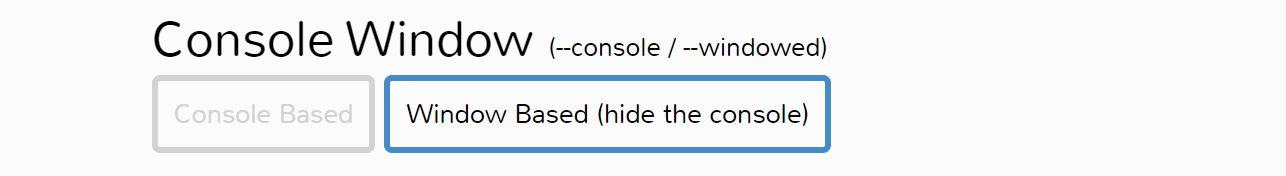
*85-rasm. Faylni yo‘lini ko‘rsatish*

**2-qadam: "One Directory" yoki "One File" ni tanlash.** Interfeysda "**One Directory**" yoki "**One File**" ni tanlash imkoniyati mavjud. Odatda ko‘pchilik katta Python loyihalar bir nechta fayllardan tashkil topgan bo‘ladi, shuning uchun "One Directory" ni tanlash afzalroq. Ushbu parametr barcha kerakli fayllar bilan papkani yaratadi, shuningdek **.exe** faylini ham yaratib qo‘yadi.



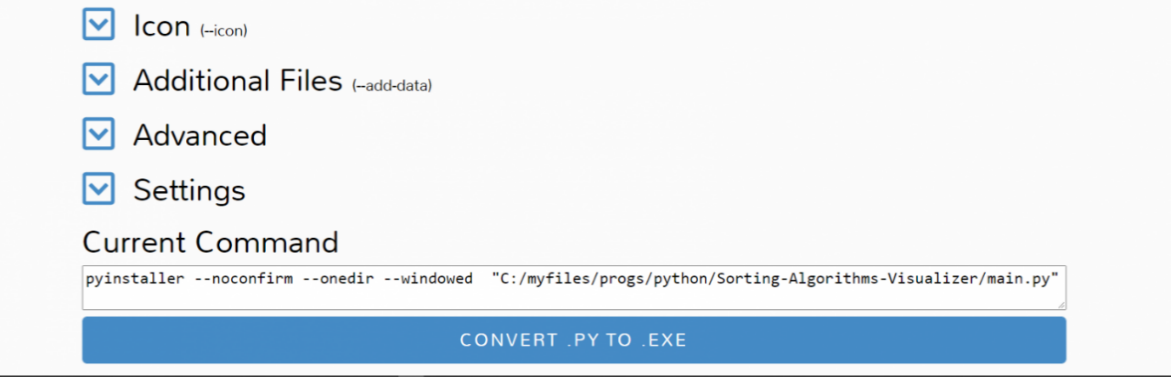
*86-rasm. "One Directory" yoki "One File" ni tanlash*

**3-qadam. “Console Based” yoki “Windows Based” ni tanlash.** Yuqoridagi amallarni bajarilganidan so‘ng dastur turini tanlash kerak: **Console** yoki **Windows** asoslangan. Agar "*Windows Based*" ni tanlansa, bu dasturning barcha konsol natijalarini yashiradi. Agar yaratilgan dastur konsol chiqishiga asoslangan holda ishlaydigan bo‘lsa, unda "*Console Based*" ni tanlash maqsadga muvofiqq bo‘ladi. Agar dasturda GUI ilovasi bo‘lsa yoki foydalanuvchiga konsol chiqishini ko‘rsatish shart bo‘lmasa, unda "Windows Based" ni tanlash tavsiya qilinadi. Bu yerda ikkinchi ikkinchi variantni tanlangan, chunki ushbu dasturda GUI mavjud.



*87-rasm. Konsolga asoslangan yoki oynaga asoslangan*

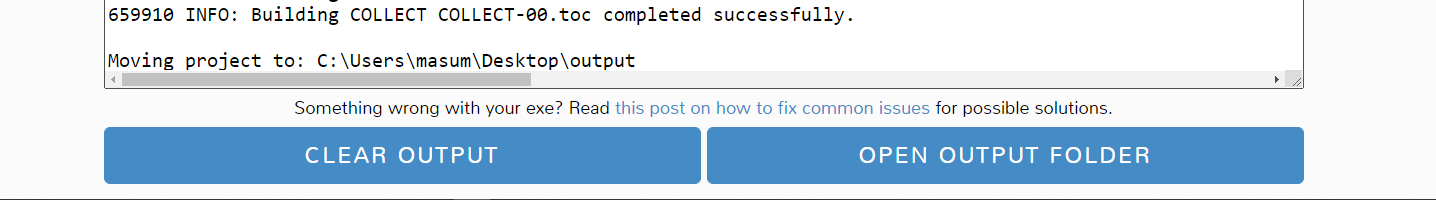
**4-qadam. Konvertatsiya qilish.** Yuqoridagibarcha amallarni bajarilganidan keyin qolgan sozlamalari masalan, piktogramma, qo‘shimcha fayllar va hk. ni o‘z standart holatida qoldirib, konvertatsiya qilish mumkin. Buning uchun faqat **CONVERT .PY TO .EXE** tugmasini bosish kifoya.



*88-rasm. Konvertatsiya qilish uchun tugma.*

**Jarayonni yakunlash uchun biroz kutishingiz kerak bo‘ladi.**

**Open output folder.** Konvertatsiya jarayoni tugaganidan so‘ng, “Open output folder” tugmasini bosish orqali konvertatsiya qilingan fayl joylashgan papkani ochish mumkin.

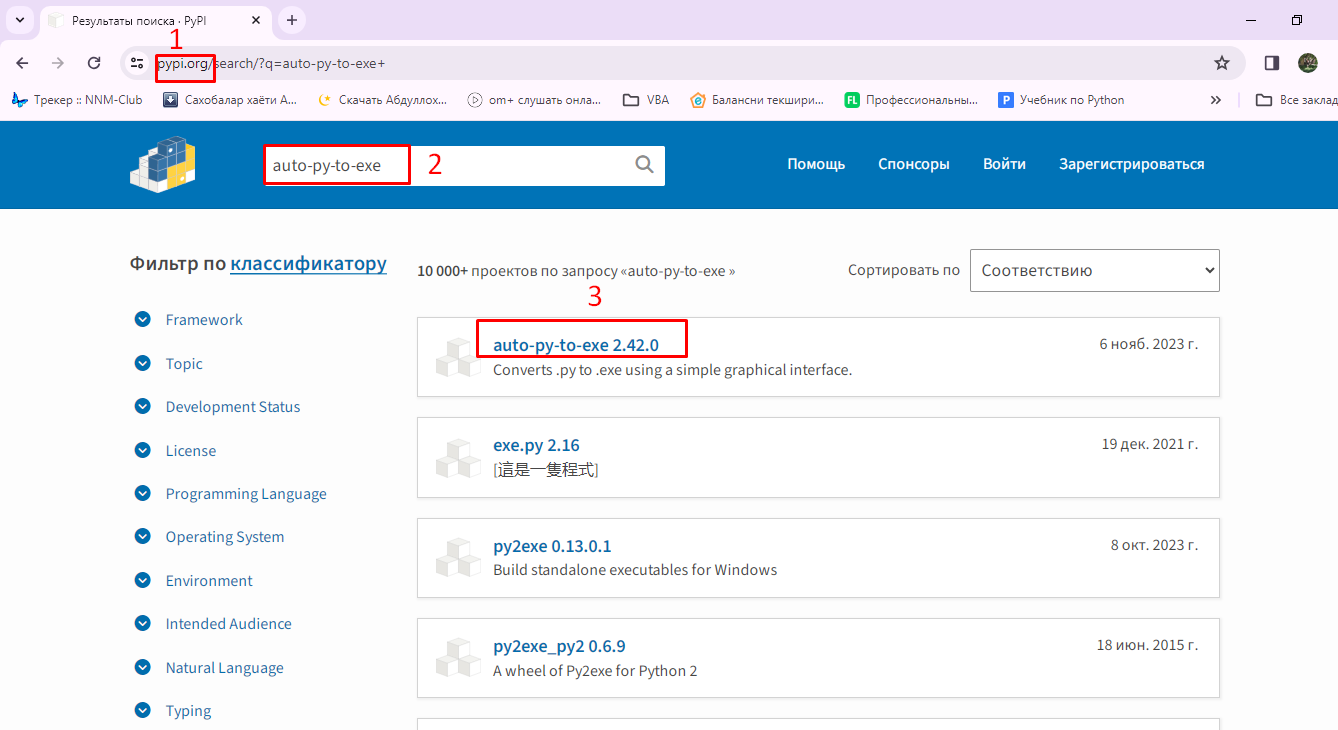


*89-rasm. Open output folder**tugmasi*

Shundan so‘ng .exe fayl joylashgan papka ochiladi. Endi uni Python-ni o‘rnatmasdan boshqa kompyuterlarda ham bemalol ishga tushirib foydalanish mumkin.

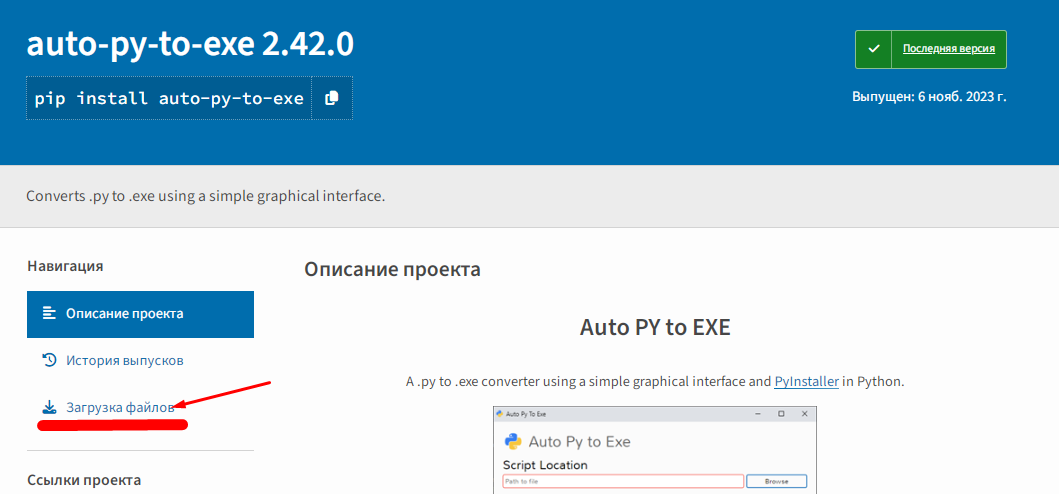
**Python paketlarini offline holatda kompyuterga o‘rnatish**

Ba’zi hollarda internet tarmog‘iga ulanmagan kompyuterlarga ham python paketlarini o‘rnatishga zarurat paydo bo‘lib qoladi. Bunday vaziyatlarda kompyuterga pythonning kerakli paketini olib kelib, o‘rnatiladi. Buning uchun boshqa bir internetga ulangan kompyuterdan ***pypi.org*** saytiga kirib, kerakli paketni qidirib topiladi va yuklab olinadi. Masalan:



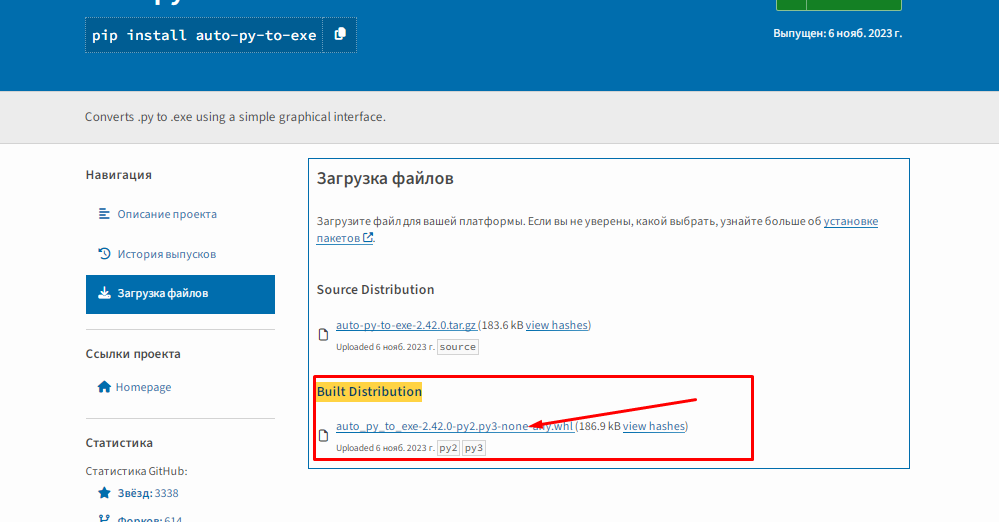
*90-rasm. Pypi.org sayti bilan ishlash*

Ushbu kerakli paketni topib, uni tanlanganda paket haqida ko‘plab kerakli ma’lumotlar beruvchi oyna ochiladi. U oynada paketni yuklab olish tugmasi ham mavjud va yuklab olish “Загрузка файлов” bo‘limini tanlanlash orqali amalga oshiriladi.



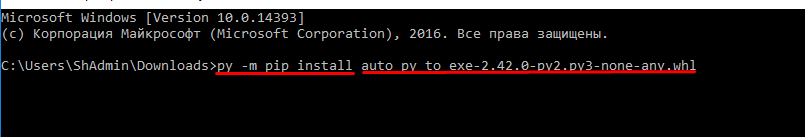
*91-rasm. Pypi.org saytidan kerakli paketni yuklab olish*

Shundan so‘ng paketni yuklab olish uchun “**Загрузка файлов**” oynasi ochiladi. U yerdan “**Built Distribution**” bo‘limidagi faylni yuklab olish uchun maxsus ssilkani bosiladi va avtomatik ravishda fayl yuklab olinadi.



*92-rasm. Kerakli paketni yuklab olish*

Ushbu yuklab olingan faylni boshqa bir kompyuterga ko‘chirib o‘tkaziladi. Endi ushbu kerakli yukanuvchi faylni kompyuterga o‘rnatish kerak. Buning uchun fayl oddiy .exe fayllar kabi emas balki maxsus usulda o‘rnatilishi kerak bo‘ladi. Ushbu amal “***py -m pip install file\_name***” shabloni yordamida amalga oshiriladi. Bizning misolimizda file Downloads papkasida joylashganligi uchun terminaldan osha papkaga kirib borib, undan so‘ng o‘rnatish kodini teriladi va enter tugmasi bosish orqali o‘rnatish amalga oshiriladi.



*93-rasm. Kerakli paketni lokal holatda o‘rnatish*

Shundan so‘ng, agarda hech qanday xatolik ro‘y bermasa, paket kompyuterga muvoffaqiyatli o‘tnatiladi.

Barcha paketlar kompyuterga lokal holatda o‘rnatilishi mumkin.

**Nazorat savollari:**

1. Pyinstaller paketining vazifasi qanday?
2. Pyinstaller paketini o‘rnatish qanday amalga oshiriladi?
3. Pyinstaller paketidan qanday foydalaniladi?
4. Python paketlarini kompyuterga qanday usullardan foydalanib, o‘rnatish mumkin?
5. Kerakli paketlarni lokal holda o‘rnatish qanday amalga oshiriladi?
6. Kompyuterga paketlarni lokal holada o‘rnatishning qanday afzalliklari bor?
7. Auto-py-to-exe paketi kompyuterga qanday o‘rnatiladi?
8. Auto-py-to-exe paketining qanday sozlamalari mavjud?
9. Auto-py-to-exe paketining PyInstaller paketidan qanday farqi mavjud?
10. Auto-py-to-exe paketi yordamida yaratilgan .exe fayl qaysi papkada yaratiladi?

**O‘ZBEKISTON RESPUBLIKASI MUDOFAA VAZIRLIGI**

**AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI**

**VA ALOQA HARBIY INSTITUTI**

**KIBERXAVFSIZLIK FAKULTETI AXBOROT TEXNOLOGIYALARI VA DASTURIY INJINIRING**

**kafedrasi**

****

**“PYTHON DASTURLASH TILI” FANIDAN**

GLOSSARIY

**Toshkent – 2024****GLOSSARIY**

|  |  |  |
| --- | --- | --- |
| **Termin** | **O‘zbek tilidagi sharhi** | **Ingliz tilidagi sharhi** |
| **!=** | Teng emas operatori; mantiqiy inkor amali qiymati bilan birhil. | The [inequality operator](http://www.stroustrup.com/glossary.html#Ginequality-operator); compares [value](http://www.stroustrup.com/glossary.html#Gvalue)s for inequality returning a [bool](http://www.stroustrup.com/glossary.html" \l "Gbool). |
| **import** | Kutubxonalarni chaqirish direktivasi |  |
| **+=** | add-and-assign operatori; masalan a+=b vazifazi jihatdan a=a+b bilan bir xil | add-and-assign [operator](http://www.stroustrup.com/glossary.html#Goperator); a+=b is roughly equivalent to a=a+b. |
| **address** | Hotira manzili | a [memory](http://www.stroustrup.com/glossary.html#Gmemory) location |
| **aggregate** | Konstruktorsiz massiv yoki struktura | an [array](http://www.stroustrup.com/glossary.html#Garray) or a [struct](http://www.stroustrup.com/glossary.html#Gstruct) without a [constructor](http://www.stroustrup.com/glossary.html#Gconstructor) |
| **algorithm** | Hisoblashning aniq ketma-ketligi | a precise definition of a computation. |
| **and** | && mantiqiy “va” (ko‘paytirish) operatori sinonimi | synonym for &&, the logical and [operator](http://www.stroustrup.com/glossary.html#Goperator). |
| **ANSI** | Amerika milliy standart agentligi. | The American national standards organization. |
| **application** | Umumiy maqsadga ega dasturlar to‘plami | a [collection](http://www.stroustrup.com/glossary.html#Gcollection) of [program](http://www.stroustrup.com/glossary.html#Gprogram)s seen as serving a common purpose (usually providing a common [interface](http://www.stroustrup.com/glossary.html#Ginterface) to their users) |
| **bit** | 0 yoki 1 qiymatga ega birlik hotira | a unit of [memory](http://www.stroustrup.com/glossary.html#Gmemory) that can hold 0 or 1. |
| **bool** | Mantiqiy tip. Bu tip faqat rost yoki yolg‘on qiymat qabul qiladi | the built-in Boolean [type](http://www.stroustrup.com/glossary.html#Gtype). A bool can have the [value](http://www.stroustrup.com/glossary.html#Gvalue)s [true](http://www.stroustrup.com/glossary.html#Gtrue) and [false](http://www.stroustrup.com/glossary.html#Gfalse). |
| **bug** | Xatolik termini | colloquial term for error. |
| **byte** | Xotiradagi bir nechta belgilar yig‘indisi | a unit of [memory](http://www.stroustrup.com/glossary.html#Gmemory) that can hold a character of the [C++](http://www.stroustrup.com/glossary.html#GC++) representation [character set](http://www.stroustrup.com/glossary.html#Gcharacter-set). |
| **Python** | Tizimli dasturlashni protsedurali qo‘llab quvvatlovchi dasturlash tili. | a [general-purpose programming language](http://www.stroustrup.com/glossary.html#Ggeneral-purpose-programming-language) with a bias towards systems programming that supports [procedural programming](http://www.stroustrup.com/glossary.html#Gprocedural-programming), [data abstraction](http://www.stroustrup.com/glossary.html#Gdata-abstraction), [object-oriented programming](http://www.stroustrup.com/glossary.html#Gobject-oriented-programming), and [generic programming](http://www.stroustrup.com/glossary.html#Ggeneric-programming).[C](http://www.stroustrup.com/glossary.html#GC)++ was designed and originally implemented by [Bjarne Stroustrup](http://www.stroustrup.com/glossary.html#GBjarne-Stroustrup). |
| **char** | Belgili tip. Har bir belgi 8 bit, ya’ni baytga teng. | [character type](http://www.stroustrup.com/glossary.html#Gcharacter-type); typically an 8-[bit](http://www.stroustrup.com/glossary.html#Gbit) [byte](http://www.stroustrup.com/glossary.html#Gbyte). |
| **input** | Standart kiritish oqimi | standard [istream](http://www.stroustrup.com/glossary.html" \l "Gistream). |
| **class** | Foydalanuvchi belgilaydigan tur, sinf. Sinf foydalanuvchi funksiyasi, foydalanuvchi ma’lumotlari va kontentlari bo‘lishi mumkin. | a [user-defined type](http://www.stroustrup.com/glossary.html#Guser-defined-type). A class can have [member function](http://www.stroustrup.com/glossary.html#Gmember-function)s, [member data](http://www.stroustrup.com/glossary.html#Gmember-data), [member constant](http://www.stroustrup.com/glossary.html#Gmember-constant)s, and [member type](http://www.stroustrup.com/glossary.html#Gmember-type)s. |
| **interpritator** | Python da yozilgan buyruqlarni mashina tiliga o‘girib beruvchi vosita. | the part of a [python](http://www.stroustrup.com/glossary.html#GC++) implementation that produces [object code](http://www.stroustrup.com/glossary.html#Gobject-code) from a [translation unit](http://www.stroustrup.com/glossary.html#Gtranslation-unit). |
| **copy()** | Nusxa olish operatori | Copy operator |
| **UML** | Унификatsiya qilingan modellashtirish tili | Unified Modeling Language |
| **OMG** | Ob’ektlarni boshqarish guruhi | Object Management Group |
| **4GL** | To‘rtinchi avlod tili | Fourth-Generation Language |
| **ANSI** | Amerika milliy standartlash instituti | American National Standards Institute |
| **AMPS** |  | Advanced Mobile Phone Service |
| **ERP** | Korxona resurslarini rejalashtirish | Enterprise Resource Planning |
| **CRM** | Mijozlar bilan o‘zaro munosabatlarni boshqarish | Customer Relations Management |
| **SQL** | Tuzilmalashgan so‘rovlar tili | Structured Query Language |
| **OLAP** | Xaqiqiy vaqtda ma’lumotlarga analitik ishlov berish | On-Line Analytical Processing |
| **OLTP** | Xaqiqiy vaqtda tranzaksiyalarga ishlov berish | On-Line Transaction Processing |
| **TCO** | Egalik qilishning yalpi qiymati | Total Cost of Ownership |
| **JIT** | Ayni vaqtida | Just-In-Time |
| **LAN** | Lokal hisoblash tarmog‘i | Local Area Network |
| **MAN** | Maxalliy hisoblash tarmog‘i | Metropolitan Area Network |
| **WAN** | Xududiy hisoblash tarmog‘i | Wide Area Network |
| **ISO** | Halqaro standartlashtirish tashkiloti | International Organization for Standardization |
| **API** | amaliy dasturlashtirish maxsus interfeysi | Application Programming Interface |
| **WWW** | Umumjahon o‘rgamchak to‘ri | World Wide Web |
| **ASCII** | Axborot almashishning Amerika standarti | American Standard Code for Information Interchange |
| **LIFO** | «Oxirida keldi, birinchi ketdi» prinsipi | Last In, First Out |
| **FIFO** | «Birinchi keldi, birinchi ketdi» prinsipi | First In, FirstOut |
| **PDA** | personalraqamlikotib | Personal Digital Assistant |
| **Axborot** | muayyan ob’ekt xususidagi bilimlarimizning noaniqlik darajasini pasaytirishga imkon beruvchi har qanday ma’lumot. | data that reduces the uncertainty degree of knowledge about certain object. |
| **Avtomatlashtirilgan axborot tizimi** | ma’lumotlarni va axborotni yaratish, uzatish, ishlash, tarqatish, saqlash va/yoki boshqarishga va hisoblashlarni amalga oshirishga mo‘ljallangan dasturiy va apparat vositalar. | a set of software and hardware designed for the creation, transmission, processing, distribution, storage and/or data and information management and production calculations. |
| **Xavf-xatarni agregirlash** | birlashgan xavf-xatarni teranrok tushunishga mo‘ljallangan bir necha xavf-xatarni bitta xavf-xatarga birlashtirish. | sombine multiple risks in a risk, aimed at a better understanding of the overall risk. |
| **Xavf - xatar taxlili** | nomuvofik hodisalar paydo bo‘lish holida kutiladigan zararni aniqlash maqsadida, ehtimollik hisoblashlardan foydalanib, tizim xarakteristikalarini va salbiy tomonlarini o‘rganish jarayoni. Xavf – xatarni tahlillash masalasi u yoki bu xavf – xatarning maqbullik darajasini aniqlashdan iborat. | the process of studying the characteristics and weaknesses of the system, conducted using a probabilistic calculations in order to determine the expected damage in case of adverse events. The task of risk analysis is to determine the acceptability of a risk to the system. |
| **Xavfsizlik xizmati ma’muri** | xavfsizlikni ta’minlashning bir yoki bir necha tizimi hamda loyihalashni nazoratlash va ulardan foydalanish xususida to‘liq tasavvurga ega shaxs (yoki shaxslar guruhi). | person (or group of people) having (s) complete understanding of one or more security systems and controls (s) design and use. |
| **Faol taxdid** | tizim holatiga atayin ruxsatsiz o‘zgartirish kiritish tahdidi. | the threat of a deliberate unauthorized system state changes. |
| **Trafik taxlili** | Trafik oqimini kuzatish (borligi, yukligi xajmi, yunalishi va chastotasi) asosida axborat xolati xususida xulosa qilish. 2. Deshifrlanishga sabab bo‘lmaydigan, ammo g‘animga yoki buzg‘unchiga uzatilayotgan ochik matn va, umuman, kuzatilayotgan aloqa tizimining ishlashi xususidagi bilvosita axborotni olishiga imkon beruvchi aloqa tizimi orqali uzatiluvchi shifrlangan xabarlar majmuining tahlili. Trafik tahlili shifrlangan xabarlarning rasmiylashtirish xususiyatlaridan, ularning uzunligi, uzatilish vaqti, uzatuvchi va qabul qiluvchi xususidagi malumotlardan foydalanadi. | Report on the state information based on observation of traffic flows (presence , absence, amount , direction and frequency . 2 . Analysis of all encrypted messages sent over the communication system does not lead to decrypt , but allowing the opponent and / or the offender obtain indirect information about the transmitted Post and generally observed on the functioning of the communication system. A. that uses features of registration messages encrypted , and their length , the transmission time , the data sender and recipient , etc. |
| **Tarmoq taxlillagichlari (sniffer)** | tarmoqtrafigini “tinglash”ni va tarmoq trafigidan avtomatik tarzda foydalanuvchilar ismini, parollarni, kredit kartalar nomerini, shu kabi boshqa axborotni ajratib olishni amalga oshiruvchi dasturlar. | programs, asking for “listening” network traffic and automatically selects the network traffic of user names, passwords, credit card numbers, other similar information. |
| **Apparat himoya** | kompyuterda ma’lumotlarni himoyalashda apparat vositalardan, masalan, chegara registrlaridan yoki qulflardan va kalitlardan foydalanish. | the use of hardware, for example, registers boundaries or locks and keys to protect data in computers. |
| **Faol xujum** | kriptotizimga yoki kriptografik protokolga hujum bo‘lib, unga binoan dushman va/yoki buzg‘unchi konuniy foydalanuvchi xarakatiga ta’sir etishi, masalan, qonuniy foydalanuvchi xabarini almashtirishi yoki yo‘q qilishi va xabarni yaratib uning nomidan uzatishi va h. mumkin. | attack on a cryptosystem or cryptographic protocol in which the offender or the enemy and can affect the legitimate user actions, for example, replace or remove legitimate posts, create and send messages on his behalf, etc. |
| **Xizmat qilishdan voz kechishga undaydigan atayin qilinmaydigan hujum** | tasodifiy yuz bergan holat natijasida (reklama natijasida qiziqishning keskin oshishi, to‘satdan va kutilmagan kamdan – kam bo‘ladigan mashhurlik va h.) DOS hujumning mavaffaqiyatli o‘tkazilgani xususida taassurot tug‘diruvchi qandaydir servis uchun xizmat qilishdan voz kechish. Ko‘pincha faollikning avj onlarida yangilik saytlariga qiziqarli axborotni joylashtirish natijasida, xamda qidiruv tizimlari yordamida ommabop URL – havolalarni o‘tkazish qobiliyati cheklangan foydalanish kanallariga ega media – resurslariga indekslash natijasida paydo bo‘ladi. | denial of service for any service which has come as a result of chance ( sharply increased interest in the result of the promotion , the sudden and unexpected exception pop, etc. ) , giving the impression of a successful DoS attack . Often occurs in times of peak activity as a result of placement of interesting information on news sites , as well as from popular search engines indexing URL- links to various media resources with limited bandwidth channel access. |
| **Xavfsizlik auditi** | kompyuter tizimi xavfsizligiga ta’sir etuvchi bo‘lishi mumkin bo‘lgan xavfli harakatlarni xarakterlovchi, oldindan aniqlangan hodisalar to‘plamini ro‘yxatga olish(audit faylida qaydlash) yo‘li bilan himoyalanishni nazoratlash. | maintain security control by registering (fixation in the audit file) a predetermined set of events that characterize the potentially dangerous actions in the computer affecting its security. |
| **Axborot tarmog‘i xavfsizligi** | axborot tarmog‘ini ruxsatsiz foydalanishdan, me’yoriy harakatiga tasodifan aralashishdan yoki komponentlarini buzishga urinishdan saqlash choralari. | measures designed to protect network information from unauthorized access, accidental or intentional interference with normal activities or attempts to destroy its components. |
| **Kompyuter tizimi xavfsizligi** | destruktiv harakatlarga va yolg‘on axborotni zo‘rlab qabul qilinishiga olib keluvchi ishlanadigan va saqlanuvchi axborotdan ruxsatsiz foydalanishga urinishlarga kompyuter tizimining qarshi tura olish hususiyati. | property computer systems to resist attempts of unauthorized access to information processed and stored, the input of information, leading to destructive actions, and the imposition of false information. |
| **Xodim xavfsizligi** | qandaydir jiddiy axborotdan foydalanish imkoniyatiga ega barcha xodimlarning kerakli avtorizatsiyaga va barcha kerakli ruxsatnomalarga egalik kafolatini ta’minlovchi usul. | method of ensuring that all personnel having access to some sensitive information has the necessary authorization, as well as all the necessary permissions. |
| **Korxona xavfsizligi** | korxona o‘z faoliyatini buzilishsiz va to‘xtalishsiz yurgiza oladigan vaqt bo‘yicha barqaror bashoratlanuvchi atrof-muhit holati. | stable condition projected time environment in which the company can carry out their activities without disturbances and interruptions. |
| **Axborotning blokirovka qilinishi** | axborotning texnik vositalar yordamida ishlanishida uning foydalanuvchanlik xususiyatining yo‘qolishi. Natijada tanishish, xujjatlash, modifikatsiyalash yoki yo‘q qilish bo‘yicha vakolatli amallarni bajarish qiyinlashadi yoki to‘xtatiladi. | the loss of information when it is processed by technical means accessibility property, expressed in difficulty or termination of authorized access to it for authorized operations familiarization, documentation, modification, or destruction. |
| **Xavf-xatarga ta’sir** | havf-xatarni modifikatsiyalash (o‘zgartirish) jarayoni. Havf-xatarga ta’sir quyidagilarni o‘z ichiga olishi mumkin: faoliyatini boshlamaslik yoki davom ettirmaslik qarori orqali xavf-xatarni oldini olish natijasida xavf-xatar paydo bo‘ladi; qulay imkoniyatdan foydalanish uchun xavf-xatarni qabul qilish yoki ko‘paytirish; xavf-xatar manbaini yo‘q qilish; imkoniyatini o‘zgartirish; oqibatlarni o‘zgartirish; xavf-xatarni boshqa taraf yoki taraflar bilan bo‘lishish (jumladan, shartnomalar va xavf-xatarni moliyalash); xavf-xatarni tushungan holda ushlab qolish. | the modification (changing) risk. Effects on risk may include: - avoid the risk by decision not to commence or to continue, resulting in the risk; - Adoption or increase the risk for a favorable opportunity; - Eliminate the source of the risk; - Changing opportunities; - Change impacts; - Sharing the risk with another party or parties (including contracts and risk financing); - Conscious retention risk. |
| **Konfidensial axborotdan foydalanish** | muayyan shaxsga tarkibida konfidensial xarakterli ma’lumot bo‘lgan axborot bilan tanishishga vakolatli mansabdor shaxsning ruxsati. | authorized authorized official introduction of a particular person with the information containing confidential information. |
| **Axborotdan ruxsatsiz foydalanish** | manfaatdor sub’ektning huquqiy hujjatlarni yoki mulkdor, axborot egasi tomonidan o‘rnatilgan himoyalanuvchi axborotdan foydalanish huquqlari yoki qoidalarini buzib himoyalanuvchi axborotga ega bo‘lishi. | preparation of protected information interested entity in violation of the legal instruments or by the owner, the owner of the information or rights of access to protected information. |
| **Tizimning osilib qolishi** | yangi topshiriqlar kiritilishini bostirish yo‘li bilan multidastur tizimini to‘xtatish (“yaxlatish”). | system hang - stop ("freezing") multiprogramming system by inhibiting the entry of new jobs. |
| **Axborotni fosh qilinishdan himoyalash** | himoyalanuvchi axborotni, ushbu axborotdan foydalanish xuquqiga ega bo‘lmagan manfaatdor sub’ektlarga (iste’molchilarga) ruxsatsiz yetkazishni bartaraf etishga yo‘naltirilgan axborot himoyasi. | the information security directed on prevention of unauthorized finishing of protected information to interested subjects (consumers), not having right of access to this information. |
| **Ijtimoiy injeneriya** | xizmatchi xodimlar va foydalanuvchilar bilan, turli nayrang, aldash va h. orqali chalg‘itish asosidagi muloqotdan olinadigan axborot yordamida axborot tizimining xavfsizlik tizimini chetlab o‘tish. | round of system of safety of system information by means of information received from contacts with the service personnel and users on the basis of their introduction in delusion at the expense of various tricks, deception and so forth. |
| **Insayder** | guruxga tegishli yashirin axborotdan foydalanish xuquqiga ega gurux a’zosi. Odatda, axborot sirqib chiqish bilan bog‘liq mojoroda muhim shaxs hisoblanadi. Shu nuqtai nazaridan, insayderlarning quyidagi xillari farqlanadi: beparvolar, manipulyatsiyalanuvchilar, ranjiganlar, qo‘shimcha pul ishlovchilar va h. | the member of group of the people having access to the classified information, belonging this group. As a rule, is the key character in the incident, connected with information leakage. From this point of view distinguish the following types of insiders: negligent, manipulated, offended, disloyal, earning additionally, introduced, etc. |
| **Insident** | ruxsatsiz foydalanish xuquqiga ega bo‘lishga yoki kompyuter tizimiga xujum o‘tkazishga urinishning qayd etilgan xoli. | the recorded case of attempt of receiving unauthorized access or carrying out attack to computer system. |
| **Tashqi insident** | manba jabrlanuvchi tomon bilan bevosita bog‘lanmagan buzg‘unchi bo‘lgan mojoro. | the incident which source is the violator who hasn't been connected with an affected party directly. |
| **Ichki insident** | manba jabrlanuvchi tomon bilan bevosita bog‘langan (mehnat shartnomasi yoki boshqa usullar bilan) buzg‘unchi bo‘lgan mojoro. | the incident which source is the violator connected with an affected party directly (the employment contract or other ways). |
| **Axborotdan noqonuniy foydalanish** | qonuniy yo‘l bilan olingan ma’lumotlarni, ushbu ma’lumotlarga va bunday harakatlarga qo‘l uruvchi sub’ektlarga o‘rnatilgan qoidalarni va vakolatlarni (sanksiyalarni) buzgan holda, uzatish, tarqatish (chop etish) va qo‘llash. | transmission, distribution(publishing), the use of information obtained through legal means in violation of the rules and powers(sanctions) established forthis information and subject,to take such action. |
| **Suqilib kirishga sinash** | tizimni uning himoyalash vositasini (xususan, ruxsatsiz foydalanishdan himoyalash vositasini) tekshirish maqsadida sinash. | system test for the purpose of check of means of its protection (in particular from illegally go access). |
| **Yashirin kanal** | xavfsizlik siyosatini buzish uchun ishlatilishi mumkin bo‘lgan, axborot texnologiyasi tizimini va avtomatlashtirilgan tizimlarni ishlab chiqaruvchilar ko‘zda tutmagan kommunikatsiya kanali. Axborot uzatish mexanizmi bo‘yicha yashirin kanallar xotira bo‘yicha yashirin kanalga, vaqt bo‘yicha yashirin kanalga va statistik yashirin kanalga bo‘linadi. | unforeseen the developer of system of technologies information and systems automated the communication channel which can be applied to security policy violation. On the information transfer mechanism c.s subdivide on: c.s. on memory c.s. on time; hidden statistical channels. |
| **Axborot sirqib chiquvchi kanal** | ko‘riklanuvchi ma’lumotlardan (tijorat siri, fizik muhit va sanoat ayg‘oqchilik vositalari majmui) ruxsatsiz foydalanishga imkon beruvchi konfidensial axborot manbaidan to niyati buzuqgacha bo‘lgan fizik yo‘l. | actual path from a source of confidential information to the malefactor, on which probably unauthorized obtaining protected data (set of a source of a trade secret, the physical environment and means of industrial espionage). |
| **Keylogger** | klaviaturali kiritishni ushlab qolishga mo‘ljallangan dastur yoki apparat vosita. Bosilgan klavishlar skan-kodlarini aniqlashni va ularni yashirincha saqlashni va/yoki yashirincha qandaydir kanal orqali uzatishni amalga oshiradi. | the program or the hardware device intended for interception of keyboard input. Carries out recognition of scan-codes of the pressed keys and the hidden preservation and/or their hidden transfer on any channel. |
| **Kompyuter jinoyatchiligi** | o‘zi yoki uchinchi shaxs uchun mulkiy foyda olish hamda raqibiga mulkiy ziyon yetkazish maqsadida axborot resursidan ruxsatsiz foydalanishni, uni modifikatsiyalashni yoki yo‘qotishni amalga oshirish. | implementation of unauthorized access to information resource, its modification (fake) or destruction for the purpose of obtaining property benefits for itself or for the third party, and also for causing property damage to the competitor. |
| **Mantiqiy “bomba”** | axborotdan ruxsatsiz foydalanish uchun ma’lum vaqtiy va axborot sharoitlarida ishga tushiriluvchi dastur. | the program which is started under certain temporary or information conditions for implementation of unauthorized access to information. |

**O‘ZBEKISTON RESPUBLIKASI MUDOFAA VAZIRLIGI**

**AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI**

**VA ALOQA HARBIY INSTITUTI**

**KIBERXAVFSIZLIK FAKULTETI AXBOROT TEXNOLOGIYALARI VA DASTURIY INJINIRING kafedrasi**

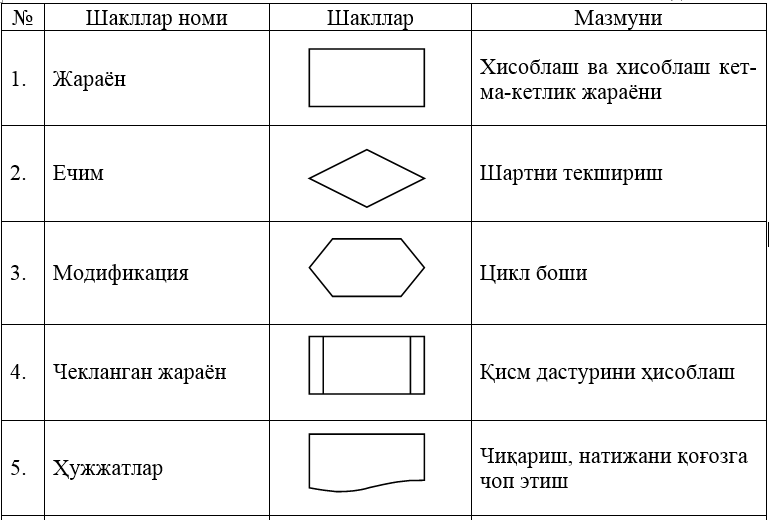
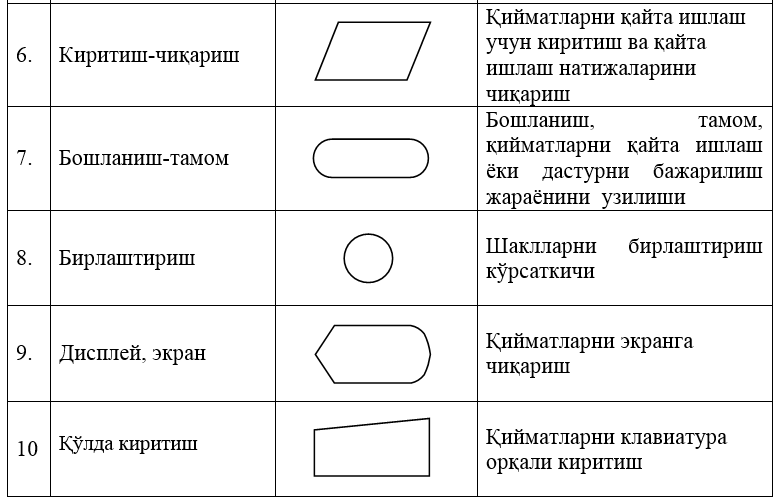
****

**“PYTHON DASTURLASH TILI” FANIDAN**

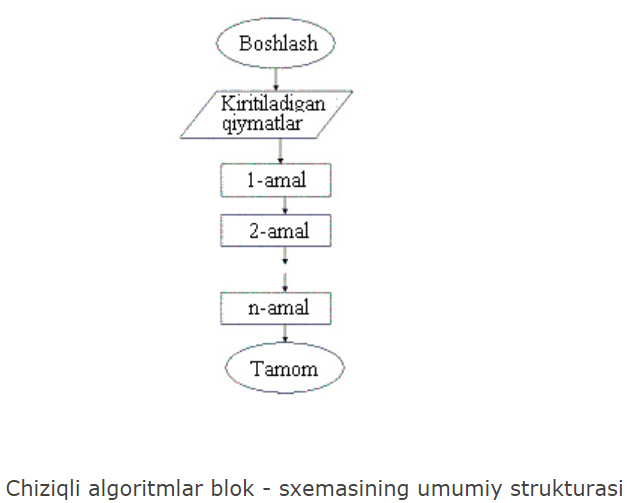
TARQATMA MATERIALLAR TO'PLAMI

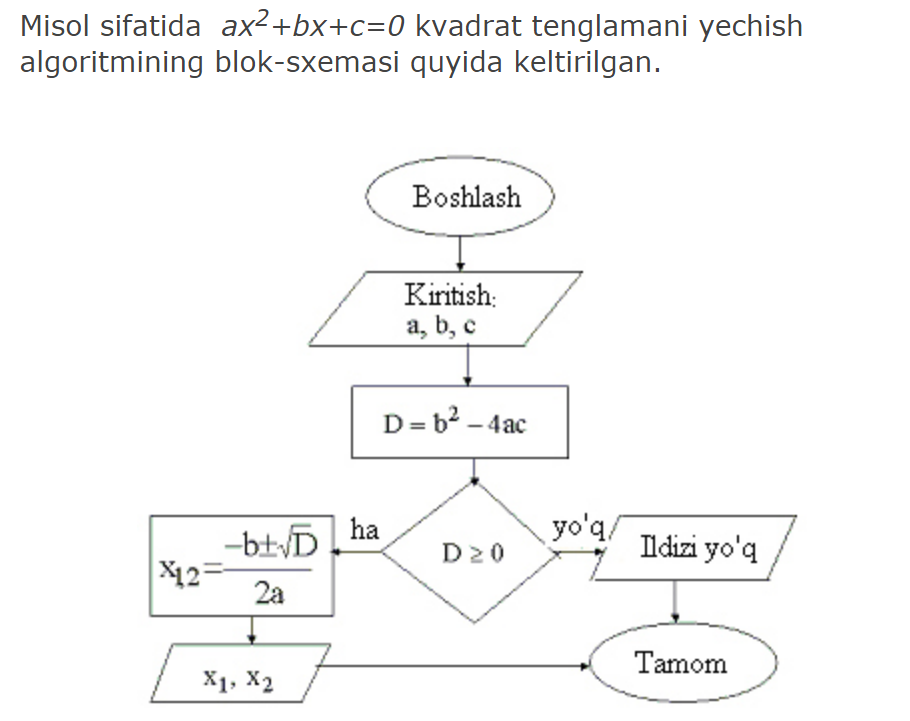
(1-ilova)

**Toshkent – 2025**

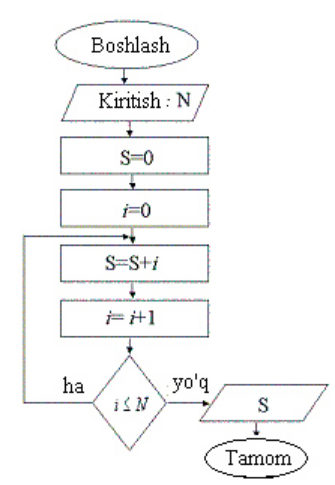
**

*1-rasm. Algoritm blok-sxemasida qo‘llaniladigan shakllar*





*2-rasm. Algoritm blok-sxemasida qo‘llaniladigan shakllar*

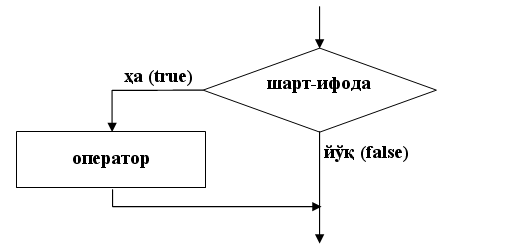


|  |  |  |  |
| --- | --- | --- | --- |
| Funksiya | Ifodalanishi | Funksiya | Ifodalanishi |
| Sin x | sin(x) |  | sqrt(x); pow(x,1/2.) |
| Cos x | cos(x) |  | abs(x) yoki fabs(x) |
| Tg x | tan(x) | Arctan x | atan(x) |
| ex | exp(x) | Arcsin x | asin(x) ? |
| Ln x | log(x) | Arccos x | acos(x) ? |
| Lg x | log10(x) |  | pow(x,2/3.) |
| xa | pow(x,a) | Log2x | log(x)/log(2) |

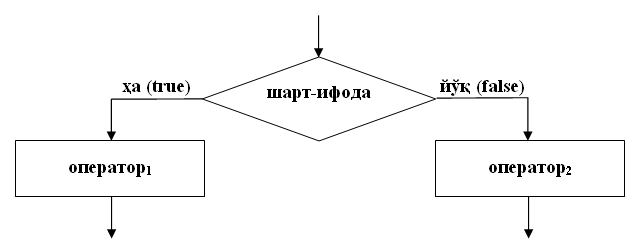
*4-rasm. Python dasturlash tilida standart funksiyalarning yozilishi*

|  |  |  |  |
| --- | --- | --- | --- |
| **Escape**  **belgilari** | **Ichki kod  (16 son)** | **Nomi** | **Amal** |
| \\ | 0x5S | \ | Teskari yon chiziqni chop etish |
| \’ | 0x27 | ‘ | Apostrofni chop etish |
| \” | 0x22 | “ | Qo‘shtirnoqni chop etish |
| \? | 0x3F | ? | So‘roq belgisi |
| \a | 0x07 | bel | Tovush signalini berish |
| \b | 0x08 | bs | Kursorni 1 belgi o‘rniga orqaga qaytarish |
| \f | 0x0C | ff | Sahifani o‘tkazish |
| \n | 0x0A | lf | Qatorni o‘tkazish |
| \r | 0x0D | cr | Kursorni ayni qatorning boshiga qaytarish |
| \t | 0x09 | ht | Navbatdagi tabulyatsiya joyiga o‘tish |
| \v | 0x0D | vt | Vertikal tabulyatsiya (pastga) |
| \000 | 000 |  | Sakkizlik kodi |
| \xNN | 0xNN |  | Belgi o‘n oltilik kodi bilan berilgan |

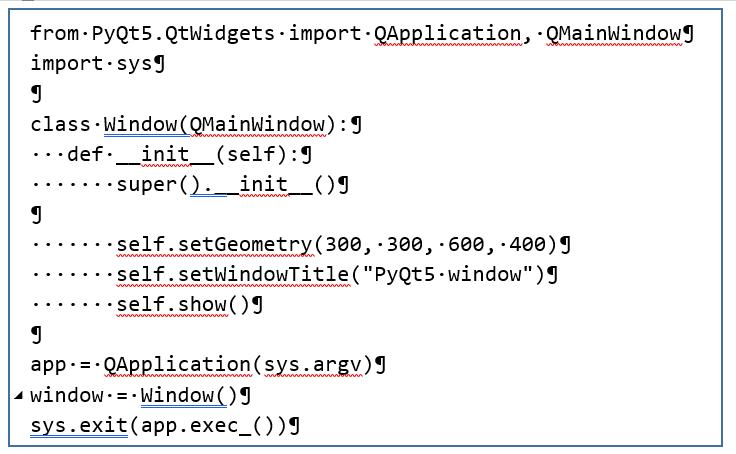
*5-rasm. PYTHON tilida escape -belgilar jadvali*



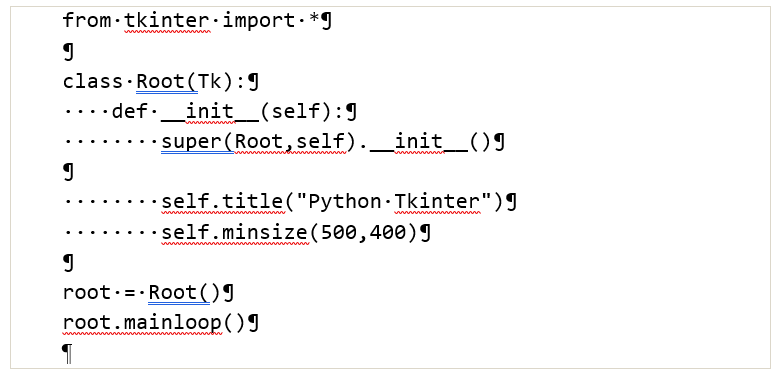
*if () shart operatorining blok sxemasi*



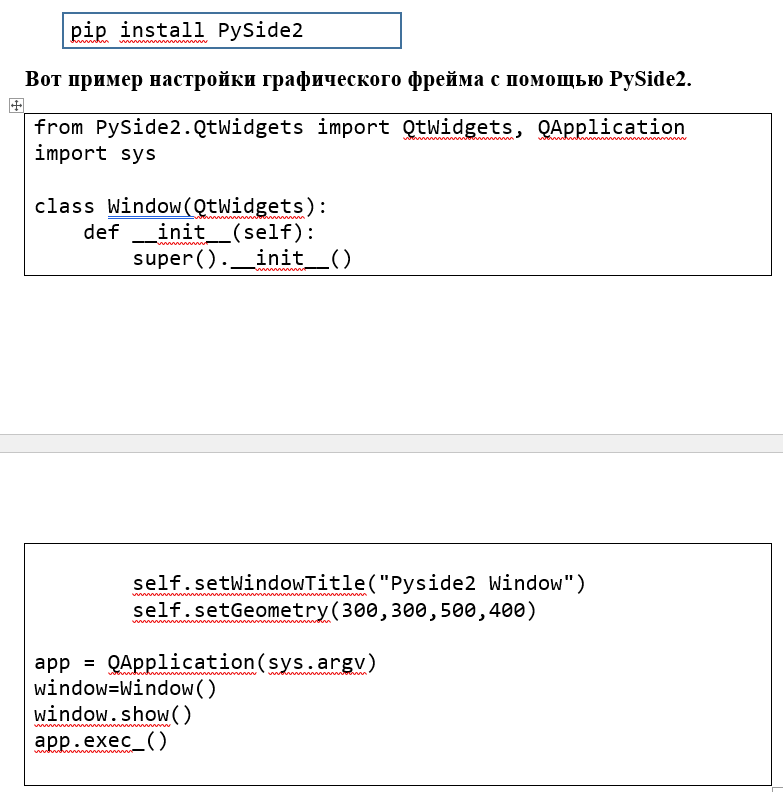
*if(); else shart operatorining blok sxemasi*



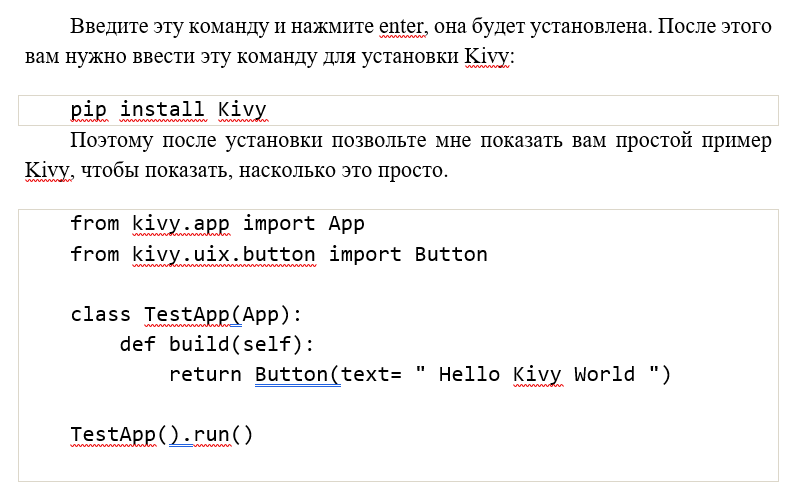
*PyQt5 bilan iashlash*



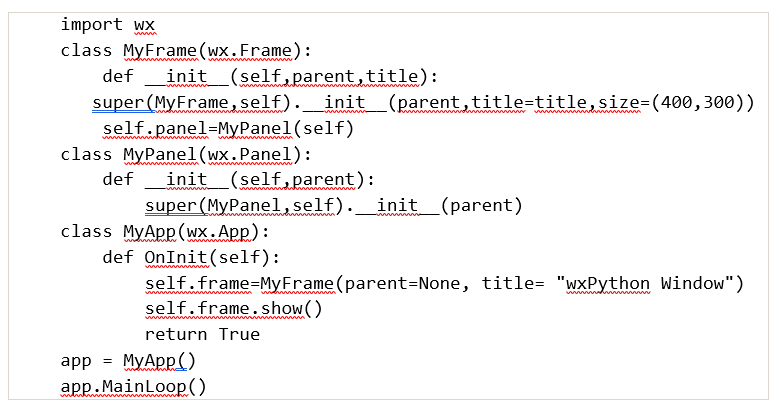
*PyQt5 paketida dastur tuzish*



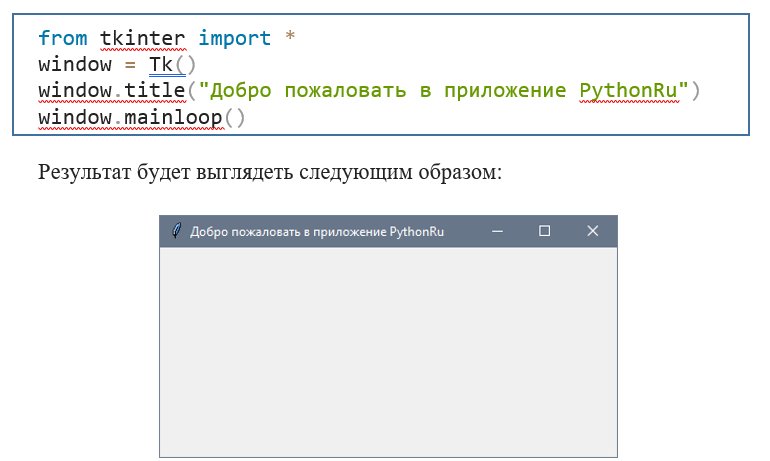
*Pip installer bilan ishlash*

*kivy paketini o‘rnatish va ishlash*

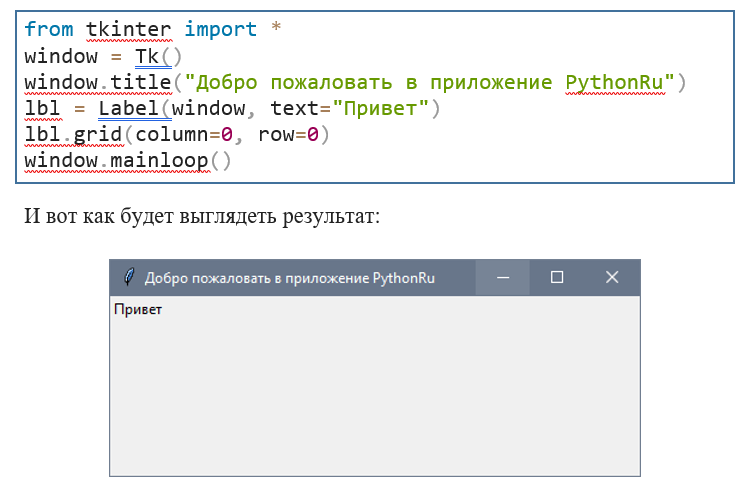




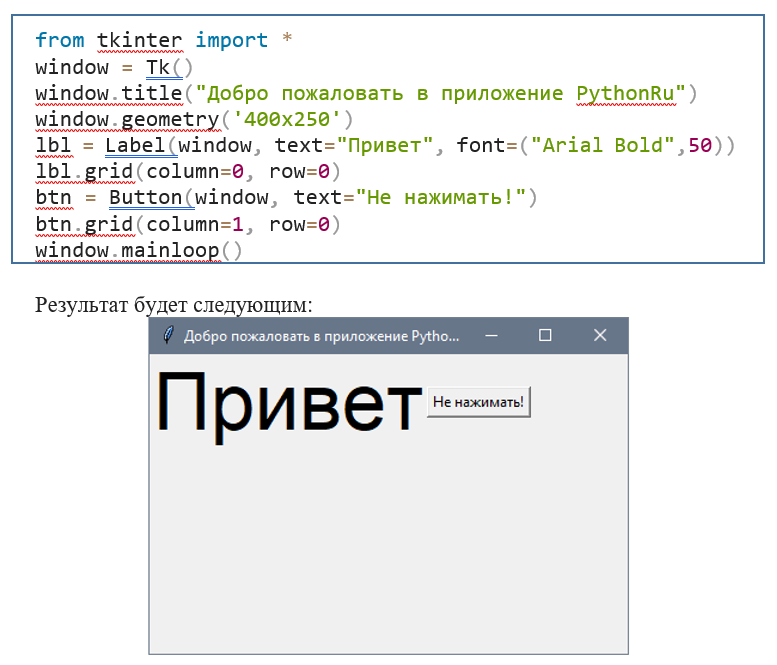
*wxPython paketini o‘rnatish va ishlash*



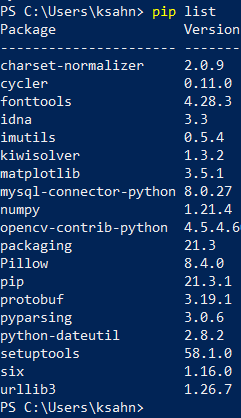
*Tkinter paketi bilan ishlash va uning natijasi*



*Tkinter paketi bilan ishlash va uning natijasi*



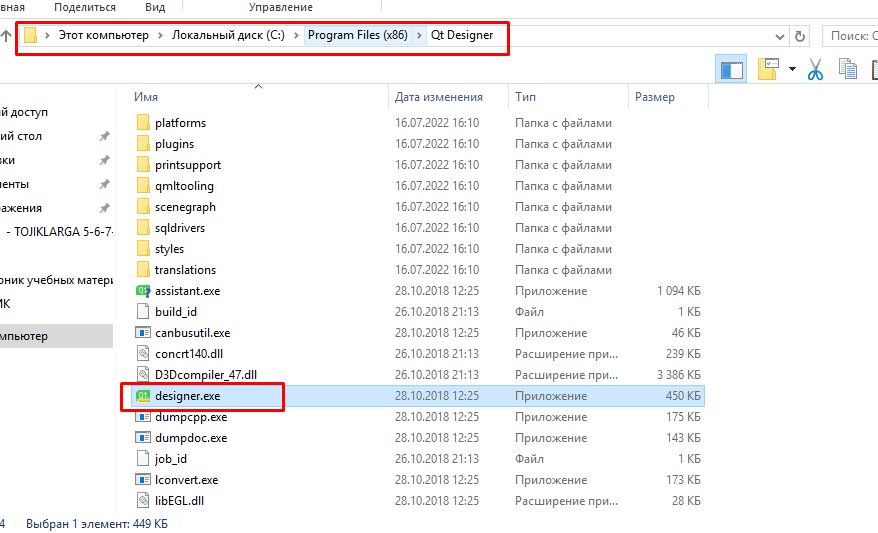
*Tkinter paketi bilan ishlash va uning natijasi*



*O‘rnatilgan paketlarning ro‘yxatini chiqarish*



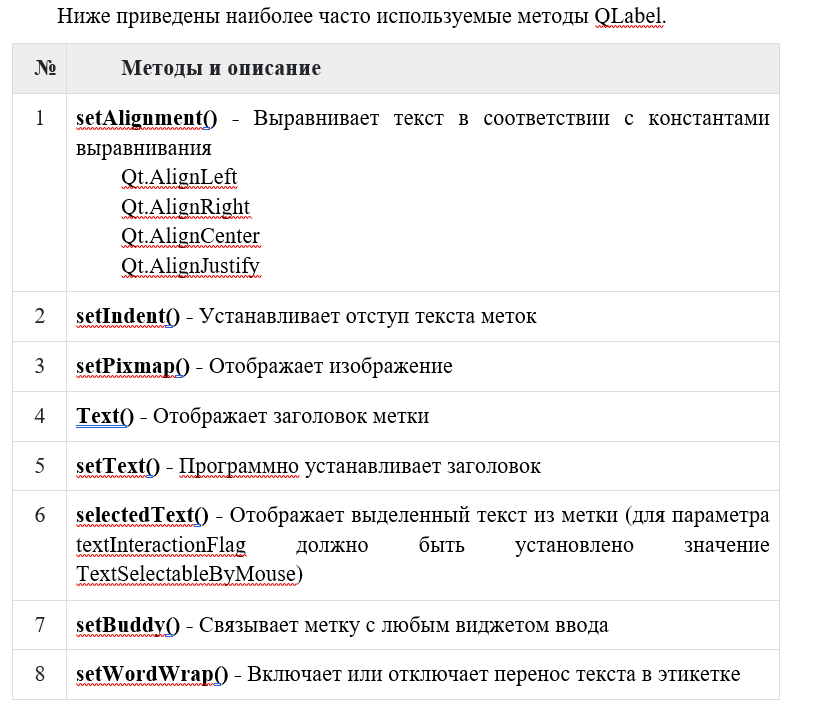
*PyQt5 paketini o‘rnatish*



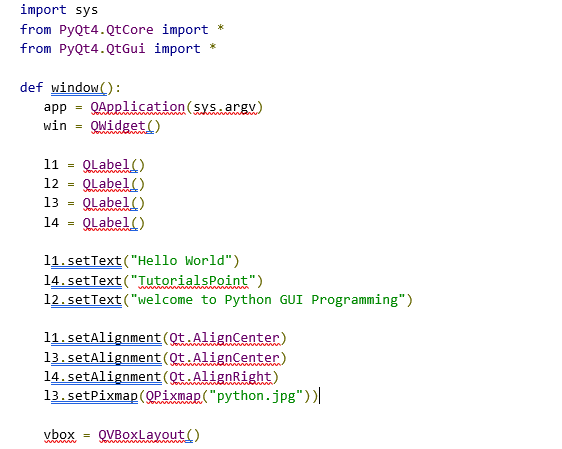
*Qt Designer dasturini o‘rnatish*



*Qt Designer dasturi yarliqini o‘rnatish*

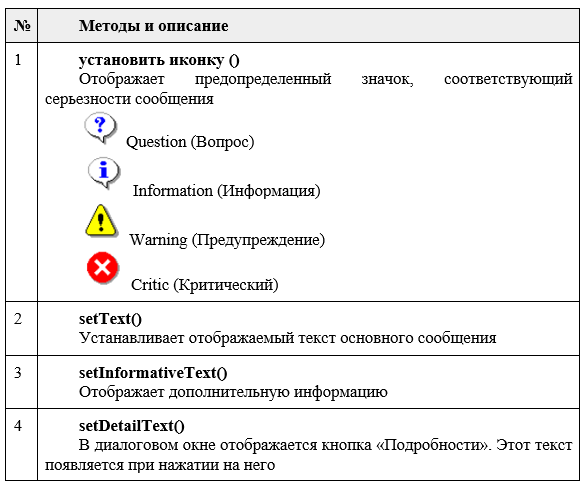
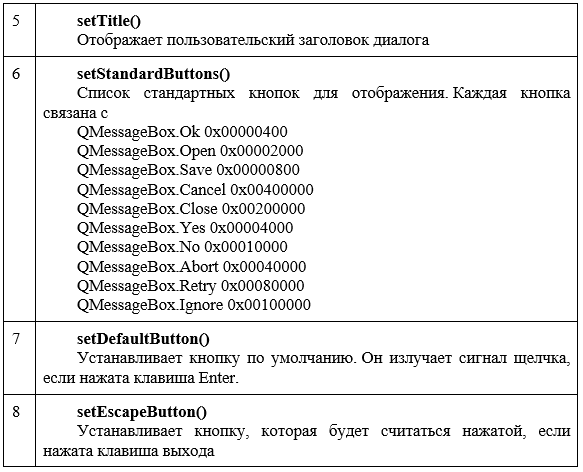


Qlabel vidgetining metodlari





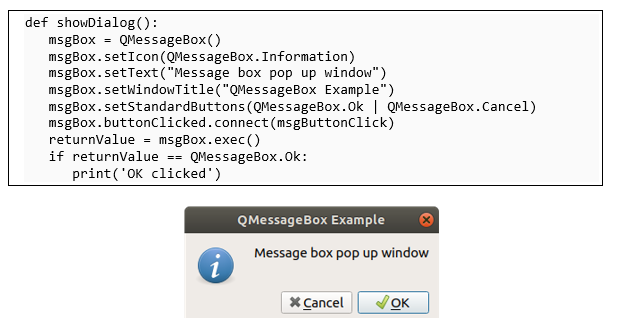
*QLabel widgeti bilan ishlash*

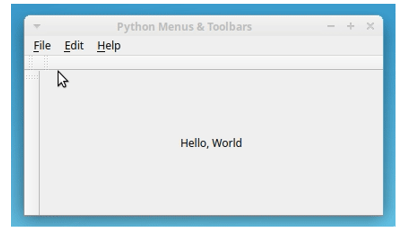
*.*



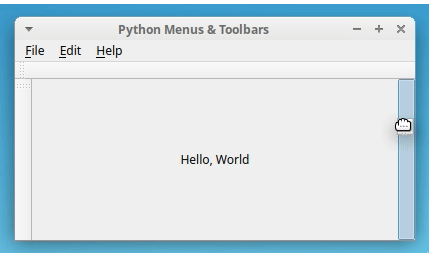
*buttonClicked() signalining funksiya bilan bog‘lanishi*



*Diolog oyna hosil qilish*



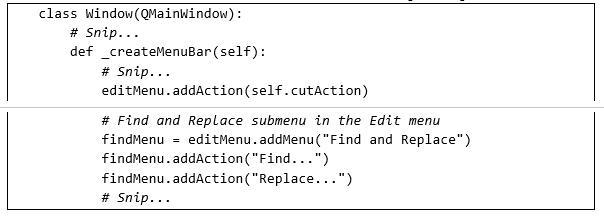
*Panel instrument hosil qilish*

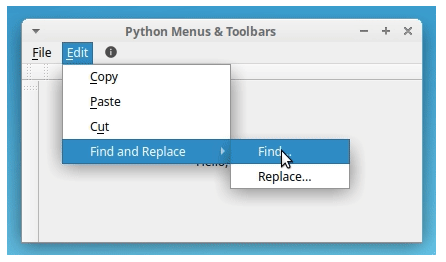


*Panel instrument hosil qilish*



*Panel instrument hosil qilish*





*Panel instrument hosil qilish*

**O‘ZBEKISTON RESPUBLIKASI MUDOFAA VAZIRLIGI**

**AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI**

**VA ALOQA HARBIY INSTITUTI**

**KIBERXAVFSIZLIK FAKULTETI AXBOROT TEXNOLOGIYALARI VA DASTURIY INJINIRING kafedrasi**

****

**“PYTHON DASTURLASH TILI” FANIDAN**

Fan bo'yicha imtihon qabul qilish uchun uslubiy ishlanma

(2-ilova)

**Toshkent** – **2025**

AXBOROT TEXNOLOGIYALARI VA SUN’IY INTELLEKT kafedrasi

“Python dasturlash tili” o‘quv fanidan \_\_\_\_\_\_\_\_ o‘quv guruhi kursantlaridan baholi sinov va imtihon qabul qilish uchun

**USLUBIY ISHLANMA**

**O‘quv-tarbiyaviy maqsadi**: Kursantlarning “Python dasturlash tili” o‘quv fanidan o‘quv dasturlariga ko‘ra o‘tilgan mavzular va mashg‘ulotlar bo‘yicha olgan nazariy bilim va amaliy ko‘nikmalarini tekshirish, dasturlash tillari va texnologiyalari bo‘yicha amaliy ko‘nikmalarini mustahkamlash.

**O‘tkazish joyi:** \_\_\_\_ - o‘quv sinflari

**O‘tkazish usuli:** og‘zaki, yozma va amaliy

**Vaqti:** 6 o‘quv soati (240 daqiqa)

|  |  |  |
| --- | --- | --- |
| O‘quv guruhi | O‘tkaziladigan sana | O‘tkaziladigan vaqti |
|  |  |  |

**“Python dasturlash tili” o‘quv fanidan o‘tkaziladigan imtihon mavzular rejasiga asosan 104 ta savolni tashkil etadi**

1.     Python paketini qanday o‘rnatish mumkin ?

2.     Python kodini oddiy matnli faylga saqlash orqali ishga tushirish uchun qanday qadamlarni bajarishim kerak ?

3.     Python dasturlash tili bilan ishlash uchun qanday tarjimonlardan foydalanish mumkin ?

4.     Python dasturlash tilining nomi qayerdan kelib chiqqan ?

5.     Python dasturlash tilining tarixi haqida gapirib bering .

6.     Python dasturlash tili logotipini kim ixtiro qilgan .

7.     Pythonni oddiy va tushunarli dasturlash tili nima qiladi ?

8.     Pythonning boshqa dasturlash tillaridan qanday afzalliklari bor ?

9.     Dasturlash tili yordamida qanday dasturlar yaratish mumkin

10. Python ? \_ GUI deganda nimani tushunasiz?

11. Tkinter paketi va uning xususiyatlari.

12. Qanday grafik paketlarni bepul o‘rnatish mumkin?

13. Tkinter paketi va uning xususiyatlari haqida gapirib bering.

14. PySide2 paketining xususiyatlari qanday?

15. Python PyQt 5 paketining xususiyatlarini aytib bering .

16. Windows muhiti uchun PyQt 5 paketi qanday o‘rnatiladi ?

17. QtDesigner dasturining imkoniyatlari haqida gapirib bering.

18. QtDesignerni qanday o‘rnataman?

19. QLabel vidjetining xususiyatlari qanday ?

20. QLabel vidjetining o‘lchamini qanday o‘zgartirish mumkin ?

21. QLineEdit vidjetining xususiyatlari qanday ?

22. QLineEdit vidjetining asosiy xususiyatlari qanday ?

23. QLabel vidjetining xususiyatlari qanday?

24. QmessageBox vidjet funksiyasi?

25. QmessageBox vidjetining asosiy xususiyatlari nimalardan iborat ?

26. Belgilar va bitmaplarning funksiyalari ?

27. Rasmni joylashtirish uchun qaysi PyQt vidjetidan foydalaniladi?

28. Tasvirni joylashtirishda Label vidjetining qaysi xususiyatidan foydalaniladi?

29. Menyuda joylashtirish uchun qanday vidjet mavjud?

30. Menyu vidjetining xususiyatlari nimalardan iborat

31. Uskunalar paneli qanday amallar bilan to‘ldiriladi?

32. PyQt da asboblar paneliga variantlar qanday qo‘shiladi?

33. Pythonda submenyu qanday yaratiladi?

34. Menyu amallar bilan qanday to‘ldiriladi?

35. PyQt da piktogramma va resurslardan qanday foydalaniladi?

36. Menyu qatorlari qanday yaratiladi?

37. Pythonda qanday arifmetik operatorlar mavjud?

38. Butun sonlarga bo‘lish qanday ishlaydi?

39. Ko‘rsatkichlar qanday ishlaydi?

40. Bu kommutativ amal nima?

41. Pythonda qanday son turlari mavjud?

42. Mantiqiy ma'lumotlar bilan qanday ishlaymiz?

43. Pythonda qanday arifmetik operatorlar mavjud?

44. Butun sonlarga bo‘lish qanday ishlaydi?

45. Ko‘rsatkichlar qanday ishlaydi?

46. ​​Bu kommutativ operatsiya nima ?

47. Pythonda son turlari qanday?

48. Mantiqiy ma'lumotlar bilan qanday ishlaymiz?

49. Argumentlar format() parametrlari sifatida?

50. Satr qo‘shish operatori ?

51. Satrlarni ko‘paytirish operatori?

52. dagi substring a'zolik operatori ?

53. Pythonda o‘rnatilgan string funktsiyalari.

54. Funktsiya ord(c) .

55. Satrning registrini o‘zgartirish .

56. string.swapcase() .

57. Satrdagi pastki qatorni toping va almashtiring.

58. String tasnifi .

59. string.islower() .

60. Chiziqlarni tekislash, chekinish

61. Qaysi operator shart operatori deyiladi?

62. If operatorining ishlash printsipi nimadan iborat ?

63. Python tilidagi if , if - else va if - elif - else iboralarining farqi nimada ?

64. Berilgan sonlarning eng kattasini topish dasturi qanday?

65. Qanday operatorlar mantiqiy operatorlar deb ataladi?

66. Mantiqiy ko`paytirish, qo`shish operatorlari nimalardan iborat?

67. Berilgan sonlardan avval kichikroq, keyin esa kattasini ko‘rsatadigan dastur qanday tuzilishga ega bo‘ladi?

68. Pythonda sikl operatori nima?

69. While Loop zanjiri printsipi nimadan iborat?

70. 1 dan 50 gacha tub sonlarni topuvchi dastur qanday tuziladi?

71. While siklida break operatoridan foydalanish sababi nima?

72. Kartej nima?

73. Bo‘g‘imli tipning ishlash printsipi nimadan iborat

74. Toya qanday qadoqlanadi va qanday yo‘llari bor?

75. Nima uchun u "obyektga mo‘ljallangan dasturlash" deb ataladi?

76. Ob'ekt nima?

77. Sinf nima?

78. Klasslar va ob'ektlar qanday yaratiladi?

79. Initsializatsiya usuli nima?

80. \_\_del\_\_() usuli qanday vazifani bajaradi?

81. Meros nima ?

82. Qanday maxsus usullar mavjud?

83. \_\_del\_\_ usuli qanday vazifani bajaradi ?

84. \_\_init\_\_ usuli qanday vazifani bajaradi ?

85. \_\_add\_\_ usuli qanday vazifani bajaradi?

86. SQLite moduli qanday chaqiriladi ?

87. SQLite moduli yordamida ma'lumotlar bazasi qanday yaratiladi ?

88. Connect () funksiyasi qanday prinsipga asoslangan ?

89. Jadval qanday buyruqlar yordamida tuziladi?

90. INSERT buyrug‘ining vazifasi va printsipi nimadan iborat ?

91. Execute () funksiyasi qanday ishlashini tushuntiring .

92. JB ning vazifasi nimadan iborat . yaqin ()?

93. JB ning vazifasi nimadan iborat . majburiyat ()?

94. Dump buyrug‘i qanday vazifani bajaradi?

95. SQLite modulidagi WHILE bandining printsipi nimadan iborat ?

96. SELECT operatorining printsipi nima ?

97. UPDATE bayoni qanday prinsipga asoslangan ?

98. DELETE operatorining printsipi nimadan iborat ?

99. HAVING operatorining ishlash printsipi nimadan iborat ?

100.   LIKE operatorining ishlash printsipi nimadan iborat ?

101.   FROM operatorining ishlash printsipi nimadan iborat ?

102.   GROUP BY buyrug‘ining maqsadi nima ?

103.   ORDER BY buyrug‘ining maqsadi nima ?

104.   Arifmetik operatorlarga ta’rif bering.

**VAQT TAQSIMOTI VA USLUBIY TIZIMI**

| **T/r** | **Imtihonning olib borilishi** | **Vaqt**  **(daq)** | **O‘qituvchining harakati** |
| --- | --- | --- | --- |
| 1. | **I. KIRISH QISMI**  - guruhni tavsiya qiluvchidan bildiruv qabul qilish, salomlashish;  - guruhning davomatini va kursantlarning imtihonga tayyorligini tekshirish (tashqi ko‘rinishi, daftarlari, sinov daftarchalari, guruh jurnali, yozuv-chizuv anjomlari va h.k.);  - imtihonni maqsadini va qabul qilish tartibini etkazish. | 5 | Guruh ikki sherengaga saflanadi. Aniqlangan kamchiliklar bartaraf etiladi. Imtihondan ozod qilingan kursantlar e’lon qilinadi  va taqdirlanadi. Imtihonga kirishga ruxsat berilmagan kursantlar e’lon qilinadi. Javob berishga tayyorlanish uchun vaqt 30 daqiqa ekanligi e’lon qilinadi, dastlabki topshiruvchi 5 ta kursant bittadan o‘quv sinfiga kiritiladi. Qolgan kursantlar kafedraning bo‘sh o‘quv sinfiga kirib tayyorlanib, kutishadi. |
| 2. | **II. ASOSIY QISM**  *Imtihonning borishi.*  Imtihon fandan o‘quv dasturlariga ko‘ra o‘tilgan mavzular va mashg‘ulotlar bo‘yicha kursantlarning olgan nazariy bilim va amaliy ko‘nikmalarini tekshirib ko‘rish va baholash maqsadida o‘tkaziladi. Chaqirilgan kursant imtihon topshirishga kelgani to‘g‘risida imtihon qabul qiluvchiga bildiruv beradi, masalan: “*Ustoz! Kursant G‘ulomov “Python dasturlash tili” fanidan imtihon topshirish uchun yetib keldi!”*, sinov daftarchasini o‘qituvchiga taqdim qiladi, so‘ngra bilet oladi, uning raqamini aytadi, bilet savollari bilan tanishib chiqadi va savollarni tushungani yoki tushunmagani to‘g‘risida bildiruv beradi, zarur bo‘lganda savollarni aniqlashtiradi, javoblarni yozish uchun o‘quv bo‘limidan ro‘yxatdan o‘tgan toza varaq oladi, so‘ngra ko‘rsatilgan joyga borib o‘tiradi va javob berishga tayyorgarlik ko‘radi, zarur bo‘lganda, o‘qituvchining ruxsati bilan kompyuterda mashq qilib tayyorlanadi.  Javob berishga tayyorlanayotgan kursant reja tuzib olishi yoki javobni yozishi mumkin, zarur hollarda sinf doskasiga yoki varaqqa chizmalarni, sxemalarni, hisoblarni va shunga o‘xshashlarni tushirishi va bunda ruxsat etilgan materiallardan, o‘quv plakatlaridan, sxemalardan foydalanishi mumkin.  Javob berishga tayyor yoki belgilangan vaqti tugagan kursant o‘qituvchining ruxsati bilan yoki uning chaqiruvi bo‘yicha etib kelib, bildiruv beradi, masalan: “*Ustoz! Kursant G‘ulomov 5-bilet savollariga javob berish uchun yetib keldi!»* va biletda berilgan savollarga javob bera boshlaydi.  Bilet savoliga javobni tugatgan kursant sinov qabul qiluvchiga bildiruv beradi, masalan: «*Ustoz! Kursant Abdullaev 1-savolga javobni tugatdi!»*.  Imtihonda har bir topshiruvchiga faqat bitta bilet olishga ruxsat beriladi. Imtihon topshiruvchi bilet savolariga javob bera olmasligi to‘g‘risida bildiruv bergan hollarda u imtihondan o‘tmagan hisoblanadi.  Imtihon topshirish vaqtida ruxsat etilmagan har xil materiallardan foydalayotgan yoki imtihonda belgilangan tartib qoidani buzayotgan kursantlar intizomiy javobgarlikka tortiladilar. Imtihon qabul qiluvchi qaroriga ko‘ra, bunday kursantlar biletsiz, butun o‘quv fani bo‘yicha imtihon topshirishi mumkin.  Javob berishga tayyorlanish uchun 30 daqiqagacha, javob berish uchun esa o‘quv me’yorini bajarish vaqtidan kelib chiqib, vaqt ajratiladi. | 220 | O‘qituvchi imtihon topshirgan kursantni baholashi tartibi:  **«a’lo» -** agarda kursant o‘quv dasturi bo‘yicha teran va puxta bilimini ko‘rsatsa, uni aniq va to‘g‘ri ifoda eta olsa, tezda to‘g‘ri qaror qabul qila olsa, javoblarda jiddiy xatolarga yo‘l qo‘ymasa, kompyuterda dastur kodlarini yoza olsa va maxsus amaliy dasturlarda to‘g‘ri ishlay olsa;  **«yaxshi» -** agarda kursant o‘quv dasturi bo‘yicha puxta bilimini ko‘rsatsa, uni aniq va to‘g‘ri ifoda eta olsa, to‘g‘ri qaror qabul qila olsa, javoblarda jiddiy xatolarga yo‘l qo‘ymasa, kompyuterda dastur kodlarini yoza olsa va maxsus amaliy dasturlarda to‘g‘ri ishlay olsa;  **«qoniqarli» -** agarda kursant o‘quv dasturini faqat asosiy qisimi bo‘yicha bilimini ko‘rsatsa, mashg‘ulotning har bir qismini to‘liq o‘zlashtirmagan bo‘lsa, javoblarda jiddiy xatolarga yo‘l qo‘ymasa, ayrim savollarga to‘g‘ri javob uchun yunaltiruvchi savollar berishni talab qilsa, kompyuterda dasturlar kodini qisman yoza olsa;  **«qoniqarsiz»** – agarda kursant javob berishda qo‘pol xatolarga yo‘l qo‘ysa, olgan bilimlari natijasida kompyuterda dastur kodlarini yoza olmasa.  Topshiruvchi bilet savoliga javobni tugatgach, imtihon qabul qiluvchi unga qo‘shimcha va aniqlashtiruvchi savollar berishi mumkin. Imtihon natijasi bilet bo‘yicha va berilgan qo‘shimcha savollarga javoblarni tugatgach, kursantga e’lon qilinadi.  Imtihonda o‘quv me’yorini bajarish nazariy (o‘quv me’yori nomi, bajariladigan ishlar hajmi va baholanish tartibini aytadi) va amaliy qismlardan iborat. O‘quv me’yorini bajarishda **umumiy baho** O‘R MV 2013-yildagi 105-sonli buuyrug‘i talablariga asosan baholanadi.  Imtihon topshirish natijasi imtihon qabul qilgan o‘qituvchi tomonidan topshiruvchiga e’lon qilinadi, imtihon vedomostiga va kursantning sinov daftarchasiga qo‘yiladi, bunda sinovdan o‘tolmasa sinov daftarchasiga qo‘yilmaydi. Har bir belgi imtihon qabul qiluvchi imzosi bilan tasdiqlanadi. |
| 3. | **III.YAKUNIY QISM:**  - imtihon natijasi e’lon qilinadi;  -imtihon natijalari tahlil qilinadi, a’lo va yomon tayyorgarlik ko‘rgan kursantlar ta’kidlab o‘tiladi, umumiy kamchiliklar ko‘rsatiladi va ularni bartaraf qilish bo‘yicha ko‘rsatmalar beriladi;  - tug‘ilgan savollarga javob beriladi;  - imtihon tugaganligi e’lon qilinadi. | 15 | Guruhning barcha shaxsiy tarkibi ikki sherengaga saflanadi. Imtihon natijalari, umumiy hatolar aytiladi, kelgusida ushbu hatolarni bartaraf etish bo‘yicha ko‘rsatmalar beriladi.  Ish joylari, kompyuter va boshqa jihozlar tartibga keltiriladi. |

**O‘ZBEKISTON RESPUBLIKASI HARBIY XAVFSIZLIK VA MUDOFAA UNIVERSITETINING AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI VA HARBIY ALOQA INSTITUTI**

**“AXBOROT TEXNOLOGIYALARI VA SUN’IY INTELLEKTI”**

**kafedrasi**

****

**«PYTHON DASTURLASH TILI» FANIDAN**

Fan bo'yicha baholi sinov va imtihon

qabul qilish uchun savollar to'plami

(3-ilova)

**Toshkent – 2025**

**“AXBOROT TEXNOLOGIYALARI VA SUN’IY INTELLEKT” KAFEDRASI**

“Python dasturlash tili” fanidan \_\_\_\_\_\_\_\_\_ guruh kursantlari uchun baholi sinov va imtihon

SAVOLLARI

1.     Python paketini qanday o‘rnatish mumkin ?

2.     Python kodini oddiy matnli faylga saqlash orqali ishga tushirish uchun qanday qadamlarni bajarishim kerak ?

3.     Python dasturlash tili bilan ishlash uchun qanday tarjimonlardan foydalanish mumkin ?

4.     Python dasturlash tilining nomi qayerdan kelib chiqqan ?

5.     Python dasturlash tilining tarixi haqida gapirib bering .

6.     Python dasturlash tili logotipini kim ixtiro qilgan .

7.     Pythonni oddiy va tushunarli dasturlash tili nima qiladi ?

8.     Pythonning boshqa dasturlash tillaridan qanday afzalliklari bor ?

9.     Dasturlash tili yordamida qanday dasturlar yaratish mumkin

10. Python ? \_ GUI deganda nimani tushunasiz?

11. Tkinter paketi va uning xususiyatlari.

12. Qanday grafik paketlarni bepul o‘rnatish mumkin?

13. Tkinter paketi va uning xususiyatlari haqida gapirib bering.

14. PySide2 paketining xususiyatlari qanday?

15. Python PyQt 5 paketining xususiyatlarini aytib bering .

16. Windows muhiti uchun PyQt 5 paketi qanday o‘rnatiladi ?

17. QtDesigner dasturining imkoniyatlari haqida gapirib bering.

18. QtDesignerni qanday o‘rnataman?

19. QLabel vidjetining xususiyatlari qanday ?

20. QLabel vidjetining o‘lchamini qanday o‘zgartirish mumkin ?

21. QLineEdit vidjetining xususiyatlari qanday ?

22. QLineEdit vidjetining asosiy xususiyatlari qanday ?

23. QLabel vidjetining xususiyatlari qanday?

24. QmessageBox vidjet funksiyasi?

25. QmessageBox vidjetining asosiy xususiyatlari nimalardan iborat ?

26. Belgilar va bitmaplarning funksiyalari ?

27. Rasmni joylashtirish uchun qaysi PyQt vidjetidan foydalaniladi?

28. Tasvirni joylashtirishda Label vidjetining qaysi xususiyatidan foydalaniladi?

29. Menyuda joylashtirish uchun qanday vidjet mavjud?

30. Menyu vidjetining xususiyatlari nimalardan iborat

31. Uskunalar paneli qanday amallar bilan to‘ldiriladi?

32. PyQt da asboblar paneliga variantlar qanday qo‘shiladi?

33. Pythonda submenyu qanday yaratiladi?

34. Menyu amallar bilan qanday to‘ldiriladi?

35. PyQt da piktogramma va resurslardan qanday foydalaniladi?

36. Menyu qatorlari qanday yaratiladi?

37. Pythonda qanday arifmetik operatorlar mavjud?

38. Butun sonlarga bo‘lish qanday ishlaydi?

39. Ko‘rsatkichlar qanday ishlaydi?

40. Bu kommutativ amal nima?

41. Pythonda qanday son turlari mavjud?

42. Mantiqiy ma'lumotlar bilan qanday ishlaymiz?

43. Pythonda qanday arifmetik operatorlar mavjud?

44. Butun sonlarga bo‘lish qanday ishlaydi?

45. Ko‘rsatkichlar qanday ishlaydi?

46. Bu kommutativ operatsiya nima ?

47. Pythonda son turlari qanday?

48. Mantiqiy ma'lumotlar bilan qanday ishlaymiz?

49. Argumentlar format() parametrlari sifatida?

50. Satr qo‘shish operatori ?

51. Satrlarni ko‘paytirish operatori?

52. dagi substring a'zolik operatori ?

53. Pythonda o‘rnatilgan string funktsiyalari.

54. Funktsiya ord(c) .

55. Satrning registrini o‘zgartirish .

56. string.swapcase() .

57. Satrdagi pastki qatorni toping va almashtiring.

58. String tasnifi .

59. string.islower() .

60. Chiziqlarni tekislash, chekinish

61. Qaysi operator shart operatori deyiladi?

62. If operatorining ishlash printsipi nimadan iborat ?

63. Python tilidagi if , if - else va if - elif - else iboralarining farqi nimada ?

64. Berilgan sonlarning eng kattasini topish dasturi qanday?

65. Qanday operatorlar mantiqiy operatorlar deb ataladi?

66. Mantiqiy ko`paytirish, qo`shish operatorlari nimalardan iborat?

67. Berilgan sonlardan avval kichikroq, keyin esa kattasini ko‘rsatadigan dastur qanday tuzilishga ega bo‘ladi?

68. Pythonda sikl operatori nima?

69. While Loop zanjiri printsipi nimadan iborat?

70. 1 dan 50 gacha tub sonlarni topuvchi dastur qanday tuziladi?

71. While siklida break operatoridan foydalanish sababi nima?

72. Kartej nima?

73. Bo‘g‘imli tipning ishlash printsipi nimadan iborat

74. Toya qanday qadoqlanadi va qanday yo‘llari bor?

75. Nima uchun u "obyektga mo‘ljallangan dasturlash" deb ataladi?

76. Ob'ekt nima?

77. Sinf nima?

78. Klasslar va ob'ektlar qanday yaratiladi?

79. Initsializatsiya usuli nima?

80. \_\_del\_\_() usuli qanday vazifani bajaradi?

81. Meros nima ?

82. Qanday maxsus usullar mavjud?

83. \_\_del\_\_ usuli qanday vazifani bajaradi ?

84. \_\_init\_\_ usuli qanday vazifani bajaradi ?

85. \_\_add\_\_ usuli qanday vazifani bajaradi?

86. SQLite moduli qanday chaqiriladi ?

87. SQLite moduli yordamida ma'lumotlar bazasi qanday yaratiladi ?

88. Connect () funksiyasi qanday prinsipga asoslangan ?

89. Jadval qanday buyruqlar yordamida tuziladi?

90. INSERT buyrug‘ining vazifasi va printsipi nimadan iborat ?

91. Execute () funksiyasi qanday ishlashini tushuntiring .

92. JB ning vazifasi nimadan iborat . yaqin ()?

93. JB ning vazifasi nimadan iborat . majburiyat ()?

94. Dump buyrug‘i qanday vazifani bajaradi?

95. SQLite modulidagi WHILE bandining printsipi nimadan iborat ?

96. SELECT operatorining printsipi nima ?

97. UPDATE bayoni qanday prinsipga asoslangan ?

98. DELETE operatorining printsipi nimadan iborat ?

99. HAVING operatorining ishlash printsipi nimadan iborat ?

100.   LIKE operatorining ishlash printsipi nimadan iborat ?

101.   FROM operatorining ishlash printsipi nimadan iborat ?

102.   GROUP BY buyrug‘ining maqsadi nima ?

103.   ORDER BY buyrug‘ining maqsadi nima ?

104.   Arifmetik operatorlarga ta’rif bering.

**O‘ZBEKISTON RESPUBLIKASI HARBIY XAVFSIZLIK VA MUDOFAA UNIVERSITETINING AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI VA HARBIY ALOQA INSTITUTI**

**“AXBOROT TEXNOLOGIYALARI VA SUN’IY INTELLEKT”**

**kafedrasi**

****

**«PYTHON DASTURLASH TILI» FANIDAN**

Fan bo'yicha mashg'ulotlarni

o'tishda foydalaniladigan

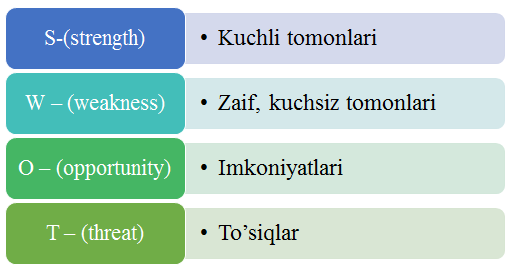
interfaol ta'lim metodlari

(4 ilova)

**Toshkent – 2025**

**“SWOT-tahlil” metodi**

**Metodning maqsadi:** mavjud nazariy bilimlar va amaliy tajribalarni tahlil qilish, taqqoslash orqali muammoni hal etish yo‘llarini topishga, bilimlarni mustahkamlash, takrorlash, baholashga, mustaqil, tanqidiy fikrlashni, nostandart tafakkurni shakllantirishga xizmat qiladi.



**Namuna:** Python dasturlash tilining SWOT tahlilini ushbu jadvalga tushiring.

|  |  |  |
| --- | --- | --- |
| **S** | Python dasturlash tilidan foydalanishning kuchli tomonlari | Ushbu dasturlash tili tizimli dasturlash tili bo‘lib, boshqa dasturlash tillarida ham tizimli masalalarni yechishda Python ga murojaat etiladi. |
| **W** | Python dasturlash tilidan foydalanishning kuchsiz tomonlari | Dasturlash tilining strukturasi murakkabligi. |
| **O** | Python dasturlash tilidan foydalanishning imkoniyatlari (ichki) | Python dasturlash tilini mukkammal o‘zlashtirgan dasturchilar muammoli masalalarni yechishda va yuzaga keladigan xatoliklarni bartaraf etishda mukammal muhit hisoblanadi. |
| **T** | To‘siqlar (tashqi) | Ma’lumotlar xavfsizligining to‘laqonli ta’minlanmaganligi |

**Metodning maqsadi:** Bu metod murakkab, ko‘ptarmoqli, mumkin qadar, muammoli xarakteridagi mavzularni o‘rganishga qaratilgan. Metodning mohiyati shundan iboratki, bunda mavzuning turli tarmoqlari bo‘yicha bir xil axborot beriladi va ayni paytda, ularning har biri alohida aspektlarda muhokama etiladi. Masalan, muammo ijobiy va salbiy tomonlari, afzallik, fazilat va kamchiliklari, foyda va zararlari bo‘yicha o‘rganiladi. Bu interfaol metod tanqidiy, tahliliy, aniq mantiqiy fikrlashni muvaffaqiyatli rivojlantirishga hamda o‘quvchilarning mustaqil g‘oyalari, fikrlarini yozma va og‘zaki shaklda tizimli bayon etish, himoya qilishga imkoniyat yaratadi. “Xulosalash” metodidan ma’ruza mashg‘ulotlarida individual va juftliklardagi ish shaklida, amaliy va seminar mashg‘ulotlarida kichik guruhlardagi ish shaklida mavzu yuzasidan bilimlarni mustahkamlash, tahlili qilish va taqqoslash maqsadida foydalanish mumkin.

**Metodni amalga oshirish tartibi:**

trener tinglovchilarni 5-6 kishidan iborat kichik guruhlarga ajratadi;

trening maqsadi, shartlari va tartibi bilan ishtirokchilarni tanishtirgach, har bir guruhga umumiy muammoni tahlil qilinishi zarur bo‘lgan qismlari tushirilgan tarqatma materiallarni tarqatadi;

har bir guruh o‘ziga berilgan muammoni atroflicha tahlil qilib, o‘z mulohazalarini tavsiya etilayotgan sxema bo‘yicha tarqatmaga yozma bayon qiladi;

navbatdagi bosqichda barcha guruhlar o‘z taqdimotlarini o‘tkazadilar. Shundan so‘ng, trener tomonidan tahlillar umumlashtiriladi, zaruriy axborotlr bilan to‘ldiriladi va mavzu yakunlanadi.

**Namuna:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dasturlash tillari** | | | | | |
| **C** | | **C++** | | **Visual C++** | |
| afzalligi | kamchiligi | afzalligi | kamchiligi | afzalligi | kamchiligi |
|  |  |  |  |  |  |
| **Xulosa:** | | | | | |

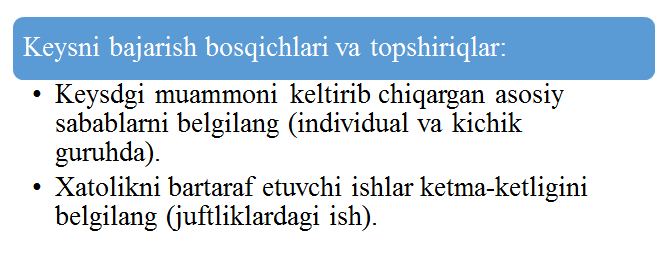
**“Keys-stadi” metodi**

**«Keys-stadi»** - inglizcha so‘z bo‘lib, («case» – aniq vaziyat, hodisa, «stadi» – o‘rganmoq, tahlil qilmoq) aniq vaziyatlarni o‘rganish, tahlil qilish asosida o‘qitishni amalga oshirishga qaratilgan metod hisoblanadi. Mazkur metod dastlab 1921 yil Garvard universitetida amaliy vaziyatlardan iqtisodiy boshqaruv fanlarini o‘rganishda foydalanish tartibida qo‘llanilgan. Keysda ochiq axborotlardan yoki aniq voqyea-hodisadan vaziyat sifatida tahlil uchun foydalanish mumkin. Keys harakatlari o‘z ichiga quyidagilarni qamrab oladi: Kim (Who), Qachon (When), Qayerda (Where), Nima uchun (Why), Qanday/ Qanaqa (How), Nima-natija (What).

**“Keys metodi” ni amalga oshirish bosqichlari**

|  |  |
| --- | --- |
| **Ish**  **Bosqichlari** | **Faoliyat shakli**  **va mazmuni** |
| **1-bosqich:** Keys va uning axborot ta’minoti bilan tanishtirish | * yakka tartibdagi audio-vizual ish; * keys bilan tanishish(matnli, audio yoki media shaklda); * axborotni umumlashtirish; * axborot tahlili; * muammolarni aniqlash |
| **2-bosqich:** Keysni aniqlashtirish va o‘quv topshirig‘ni belgilash | * individual va guruhda ishlash; * muammolarni dolzarblik iyerarxiyasini aniqlash; * asosiy muammoli vaziyatni belgilash |
| **3-bosqich:** Keysdagi asosiy muammoni tahlil etish orqali o‘quv topshirig‘ining yechimini izlash, hal etish yo‘llarini ishlab chiqish | * individual va guruhda ishlash; * muqobil yechim yo‘llarini ishlab chiqish; * har bir yechimning imkoniyatlari va to‘siqlarni tahlil qilish; * muqobil yechimlarni tanlash |
| **4-bosqich:** Keys yechimini yechimini shakllantirish va asoslash, taqdimot. | * yakka va guruhda ishlash; * muqobil variantlarni amalda qo‘llash imkoniyatlarini asoslash; * ijodiy-loyiha taqdimotini tayyorlash; * yakuniy xulosa va vaziyat yechimining amaliy aspektlarini yoritish |

**Keys.** Berilgan topshiriq asosida dastur algoritmi tuzilib C++ dasturlash tilida dastur matni yozildi. Dasturni acm.tuit.uz saytiga yuborilganda “kompilyatsiyada hatolik” habari chiqdi. Ya’ni Sistema yechimni qabul qilmadi.

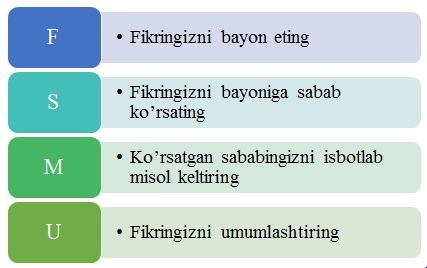


**«FSMU» metodi**

**Texnologiyaning maqsadi:** Mazkur texnologiya ishtirokchilardagi umumiy fikrlardan xususiy xulosalar chiqarish, taqqoslash, qiyoslash orqali axborotni o‘zlashtirish, xulosalash, shuningdek, mustaqil ijodiy fikrlash ko‘nikmalarini shakllantirishga xizmat qiladi. Mazkur texnologiyadan ma’ruza mashg‘ulotlarida, mustahkamlashda, o‘tilgan mavzuni so‘rashda, uyga vazifa berishda hamda amaliy mashg‘ulot natijalarini tahlil etishda foydalanish tavsiya etiladi.

**Texnologiyani amalga oshirish tartibi:**

* qatnashchilarga mavzuga oid bo‘lgan yakuniy xulosa yoki g‘oya taklif etiladi;
* har bir ishtirokchiga FSMU texnologiyasining bosqichlari yozilgan qog‘ozlarni tarqatiladi:



* ishtirokchilarning munosabatlari individual yoki guruhiy tartibda taqdimot qilinadi.

FSMU tahlili qatnashchilarda kasbiy-nazariy bilimlarni amaliy mashqlar va mavjud tajribalar asosida tezroq va muvaffaqiyatli o‘zlashtirilishiga asos bo‘ladi.

**Namuna.**

**Fikr:** “Polimarfizim obyektga yo‘naltirilgan dasturlashning asosiy tamoyillaridan biridir”.

**Topshiriq:** Mazkur fikrga nisbatan munosabatingizni FSMU orqali tahlil qiling.

**“Assesment” metodi**

**Metodning maqsadi:** mazkur metod ta’lim oluvchilarning bilim darajasini baholash, nazorat qilish, o‘zlashtirish ko‘rsatkichi va amaliy ko‘nikmalarini tekshirishga yo‘naltirilgan. Mazkur texnika orqali ta’lim oluvchilarning bilish faoliyati turli yo‘nalishlar (test, amaliy ko‘nikmalar, muammoli vaziyatlar mashqi, qiyosiy tahlil, simptomlarni aniqlash) bo‘yicha tashhis qilinadi va baholanadi.

**Metodni amalga oshirish tartibi:**

“Assesment” lardan ma’ruza mashg‘ulotlarida talabalarning yoki qatnashchilarning mavjud bilim darajasini o‘rganishda, yangi ma’lumotlarni bayon qilishda, seminar, amaliy mashg‘ulotlarda esa mavzu yoki ma’lumotlarni o‘zlashtirish darajasini baholash, shuningdek, o‘z-o‘zini baholash maqsadida individual shaklda foydalanish tavsiya etiladi. Shuningdek, o‘qituvchining ijodiy yondashuvi hamda o‘quv maqsadlaridan kelib chiqib, assesmentga qo‘shimcha topshiriqlarni kiritish mumkin.

**Namuna.** Har bir katakdagi to‘g‘ri javob 5 ball yoki 1-5 balgacha baholanishi mumkin.

**Test**

* 1.10-8 aniqlikda natijani chop etish qanday bajariladi?
* А. printf(“%.8f”,x)
* В. cout<<x
* С. cout<<(“%.8f”,x)

**Qiyosiy tahlil**

* Dasturiy mahsulotlardan foydalanish ko‘rsatkichlarini tahlil qiling?

**Tushuncha tahlili**

* STD qisqartmasini izohlang.

**Amaliy ko‘nikma**

* Grafik muhitida ishlash uchun qo‘shimcha kutubxona fayllarini o‘rnating va sozlang.

**“Insert” metodi**

**Metodning maqsadi:** Mazkur metod o‘quvchilarda yangi axborotlar tizimini qabul qilish va bilmlarni o‘zlashtirilishini engillashtirish maqsadida qo‘llaniladi, shuningdek, bu metod o‘quvchilar uchun xotira mashqi vazifasini ham o‘taydi.

**Metodni amalga oshirish tartibi:**

* o‘qituvchi mashg‘ulotga qadar mavzuning asosiy tushunchalari mazmuni yoritilgan input-matnni tarqatma yoki taqdimot ko‘rinishida tayyorlaydi;
* yangi mavzu mohiyatini yorituvchi matn talim oluvchilarga tarqatiladi yoki taqdimot ko‘rinishida namoyish etiladi;
* talim oluvchilar individual tarzda matn bilan tanishib chiqib, o‘z shaxsiy qarashlarini maxsus belgilar orqali ifodalaydilar. Matn bilan ishlashda kursantlar yoki qatnashchilarga quyidagi maxsus belgilardan foydalanish tavsiya etiladi:

|  |  |  |  |
| --- | --- | --- | --- |
| **Belgilar** | **1-matn** | **2-matn** | **3-matn** |
| “V” – tanish ma’lumot. |  |  |  |
| “?” – mazkur ma’lumotni tushunmadim, izoh kerak. |  |  |  |
| “+” bu ma’lumot men uchun yangilik. |  |  |  |
| “–” bu fikr yoki mazkur ma’lumotga qarshiman? |  |  |  |

Belgilangan vaqt yakunlangach, talim oluvchilar uchun notanish va tushunarsiz bo‘lgan ma’lumotlar o‘qituvchi tomonidan tahlil qilinib, izohlanadi, ularning mohiyati to‘liq yoritiladi. Savollarga javob beriladi va mashg‘ulot yakunlanadi.

**“Tushunchalar tahlili” metodi**

**Metodning maqsadi:** mazkur metod kursantlar yoki qatnashchilarni mavzu buyicha tayanch tushunchalarni o‘zlashtirish darajasini aniqlash, o‘z bilimlarini mustaqil ravishda tekshirish, baholash, shuningdek, yangi mavzu buyicha dastlabki bilimlar darajasini tashhis qilish maqsadida qo‘llaniladi.

Metodni amalga oshirish tartibi:

* ishtirokchilar mashg‘ulot qoidalari bilan tanishtiriladi;
* o‘quvchilarga mavzuga yoki bobga tegishli bo‘lgan so‘zlar, tushunchalar nomi tushirilgan tarqatmalar beriladi ( individual yoki guruhli tartibda);
* o‘quvchilar mazkur tushunchalar qanday ma’no anglatishi, qachon, qanday holatlarda qo‘llanilishi haqida yozma ma’lumot beradilar;
* belgilangan vaqt yakuniga etgach o‘qituvchi berilgan tushunchalarning tugri va tuliq izohini uqib eshittiradi yoki slayd orqali namoyish etadi;
* har bir ishtirokchi berilgan tugri javoblar bilan uzining shaxsiy munosabatini taqqoslaydi, farqlarini aniqlaydi va o‘z bilim darajasini tekshirib, baholaydi.

**Namuna:** “Moduldagi tayanch tushunchalar tahlili”

|  |  |  |
| --- | --- | --- |
| **Tushunchalar** | **Sizningcha bu tushuncha qanday ma’noni anglatadi?** | **Qo‘shimcha ma’lumot** |
| Registr | Foylanuvchini ro‘yxatga olish |  |
| Subcribe | Foydalanuvchini xizmatga a’zo bo‘lishini ta’minlash |  |
| Invite | Ulanish o‘rnatilishi uchun sofwswitch bilan bog‘lanish |  |
| Acknowedge | Aloqani tasdiqlash va foydalanuvchiga aloqa qilishini ta’minlash |  |
| 200 Ok | Softswitch foydalanuvchining so‘roviga ijobiy javob qaytarishi |  |
| Release | Aloqa tugatilganida kanalni bo‘sh holatga qaytarish |  |

**Izoh:** Ikkinchi ustunchaga qatnashchilar tomonidan fikr bildiriladi. Mazkur tushunchalar haqida qo‘shimcha ma’lumot glossariyda keltirilgan.

**Venn Diagrammasi metodi**

**Metodning maqsadi:** Bu metod grafik tasvir orqali o‘qitishni tashkil etish shakli bo‘lib, u ikkita o‘zaro kesishgan aylana tasviri orqali ifodalanadi. Mazkur metod turli tushunchalar, asoslar, tasavurlarning analiz va sintezini ikki aspekt orqali ko‘rib chiqish, ularning umumiy va farqlovchi jihatlarini aniqlash, taqqoslash imkonini beradi.

**Metodni amalga oshirish tartibi:**

* ishtirokchilar ikki kishidan iborat juftliklarga birlashtiriladilar va ularga ko‘rib chiqilayotgan tushuncha yoki asosning o‘ziga xos, farqli jihatlarini (yoki aksi) doiralar ichiga yozib chiqish taklif etiladi;
* navbatdagi bosqichda ishtirokchilar to‘rt kishidan iborat kichik guruhlarga birlashtiriladi va har bir juftlik o‘z tahlili bilan guruh a’zolarini tanishtiradilar;
* juftliklarning tahlili eshitilgach, ular birgalashib, ko‘rib chiqilayotgan muammo yohud tushunchalarning umumiy jihatlarini (yoki farqli) izlab topadilar, umumlashtiradilar va doirachalarning kesishgan qismiga yozadilar.

**Namuna:** Dasturlash tillari turlari bo‘yicha

**Python / C++**

**Python dasturlash tili**

**C++ dasturlsh tili**

**“Blis-o‘yin” metodi**

**Metodning maqsadi:** o‘quvchilarda tezlik, axborotlar tizmini tahlil qilish, rejalashtirish, prognozlash ko‘nikmalarini shakllantirishdan iborat. Mazkur metodni baholash va mustahkamlash maksadida qo‘llash samarali natijalarni beradi.

**Metodni amalga oshirish bosqichlari:**

1. Dastlab ishtirokchilarga belgilangan mavzu yuzasidan tayyorlangan topshiriq, ya’ni tarqatma materiallarni alohida-alohida beriladi va ulardan materialni sinchiklab o‘rganish talab etiladi. Shundan so‘ng, ishtirokchilarga to‘g‘ri javoblar tarqatmadagi «yakka baho» kolonkasiga belgilash kerakligi tushuntiriladi. Bu bosqichda vazifa yakka tartibda bajariladi.

2. Navbatdagi bosqichda trener-o‘qituvchi ishtirokchilarga uch kishidan iborat kichik guruhlarga birlashtiradi va guruh a’zolarini o‘z fikrlari bilan guruhdoshlarini tanishtirib, bahslashib, bir-biriga ta’sir o‘tkazib, o‘z fikrlariga ishontirish, kelishgan holda bir to‘xtamga kelib, javoblarini «guruh bahosi» bo‘limiga raqamlar bilan belgilab chiqishni topshiradi. Bu vazifa uchun 15 daqiqa vaqt beriladi.

3. Barcha kichik guruhlar o‘z ishlarini tugatgach, to‘g‘ri harakatlar ketma-ketligi trener-o‘qituvchi tomonidan o‘qib eshittiriladi, va o‘quvchilardan bu javoblarni «to‘g‘ri javob» bo‘limiga yozish so‘raladi.

4. «To‘g‘ri javob» bo‘limida berilgan raqamlardan «yakka baho» bo‘limida berilgan raqamlar taqqoslanib, farq bulsa «0», mos kelsa «1» ball quyish so‘raladi. Shundan so‘ng «yakka xato» bo‘limidagi farqlar yuqoridan pastga qarab qo‘shib chiqilib, umumiy yig‘indi hisoblanadi.

5. Xuddi shu tartibda «to‘g‘ri javob» va «guruh bahosi» o‘rtasidagi farq chiqariladi va ballar «guruh xatosi» bo‘limiga yozib, yuqoridan pastga qarab qo‘shiladi va umumiy yig‘indi keltirib chiqariladi.

6. Trener-o‘qituvchi yakka va guruh xatolarini to‘plangan umumiy yig‘indi bo‘yicha alohida-alohida sharhlab beradi.

7. Ishtirokchilarga olgan baholariga qarab, ularning mavzu bo‘yicha o‘zlashtirish darajalari aniqlanadi.

**«Dasturiy vositalarni o‘rnatish va sozlash» ketma-ketligini joylashtiring. O‘zingizni tekshirib ko‘ring!**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Harakatlar mazmuni** | **Yakka baho** | **Yakka xato** | **To‘g‘ri javob** | **Guruh bahosi** | **Guruh xatosi** |
| import math |  |  | 1 |  |  |
| print(“Hello world!”) |  |  | 2 |  |  |

**“Brifing” metodi**

“Brifing”- (ing. briefing-qisqa) biror-bir masala yoki savolning muhokamasiga bag‘ishlangan qisqa press-konferensiya.

**O‘tkazish bosqichlari:**

Taqdimot qismi.

Muhokama jarayoni (savol-javoblar asosida).

Brifinglardan trening yakunlarini tahlil qilishda foydalanish mumkin. Shuningdek, amaliy o‘yinlarning bir shakli sifatida qatnashchilar bilan birga dolzarb mavzu yoki muammo muhokamasiga bag‘ishlangan brifinglar tashkil etish mumkin bo‘ladi. Tinglovchilar yoki tinglovchilar tomonidan yaratilgan mobil ilovalarning taqdimotini o‘tkazishda ham foydalanish mumkin. **“Portfolio”** **metodi** “Portfolio” – ( ital. portfolio-portfel, ingl.hujjatlar uchun papka) ta’limiy va kasbiy faoliyat natijalarini autentik baholashga xizmat qiluvchi zamonaviy ta’lim texnologiyalaridan hisoblanadi. Portfolio mutaxassisning saralangan o‘quv-metodik ishlari, kasbiy yutuqlari yig‘indisi sifatida aks etadi. Jumladan, talaba yoki tinglovchilarning modul yuzasidan o‘zlashtirish natijasini elektron portfoliolar orqali tekshirish mumkin bo‘ladi. Oliy ta’lim muassasalarida portfolioning quyidagi turlari mavjud:

|  |  |  |
| --- | --- | --- |
| **Faoliyat turi** | **Ish shakli** | |
| **Individual** | **Guruhiy** |
| Ta’limiy  faoliyat | Tinglovchilar portfoliosi, bitiruvchi, doktorant, tinglovchi portfoliosi va boshq. | Tinglovchilar guruhi, tinglovchilar guruhi portfoliosi va boshq. |
| Pedagogik faoliyat | O‘qituvchi portfoliosi, rahbar xodim portfoliosi | Kafedra, fakultet, markaz, OTM portfoliosi va boshq. |

**O‘ZBEKISTON RESPUBLIKASI MUDOFAA VAZIRLIGI**

**AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI**

**VA ALOQA HARBIY INSTITUTI**

**KIBERXAVFSIZLIK FAKULTETI**

**AXBOROT TEXNOLOGIYALARI VA DASTURIY INJINIRING**

**kafedrasi**

****

**«PYTHON DASTURLASH TILI»**

**FANINING**

O'QUV DASTURI

**Toshkent 2025**

**O‘ZBEKISTON RESPUBLIKASI HARBIY XAVFSIZLIK VA MUDOFAA UNIVERSITETINING AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI VA HARBIY ALOQA INSTITUTI**

**“AXBOROT TEXNOLOGIYALARI VA SUN’IY INTELLEKTI”**

**kafedrasi**

PYTHON DASTURLASH TILI FANINING

OʻQUV DASTURI

|  |  |  |
| --- | --- | --- |
| **Bilim sohasi:** |  | 1000 000 - Xizmatlar |
| **Ta’lim sohasi:** |  | 1030 000 - Xavfsizlik xizmati |
| **Ta’lim yoʻnalishlari:** |  | 61030700 - Qo‘shinlarning taktik qo‘mondonlik muhandisligi (Aloqa qo‘shinlari) |

**Toshkent 2025**

**PYTHON DASTURLASH TILI**

**1. Oʻquv fanining dolzarbligi va oliy kasbiy ta’lim dasturidagi oʻrni**

“Python dasturlash tili” fani Qoʻshinlarning taktik qoʻmondonlik muhandisligi (Aloqa qo‘shinlari) mutaxassisligi boʻyicha tayyorlanadigan boʻlajak ofitserlarni yetuk mutaxassis boʻlib chiqishida muhim ahamiyatga ega. Ushbu fan bugungi zamonaviy AKT vositalari rivojlangan davrda juda ham **dolzarb** ahamiyatga ega hisoblanadi.

Ushbu oʻquv fanining **oliy kasbiy ta’lim dasturidagi oʻrni** muhim hisoblanib, turli sohalarda zamonaviy texnologiyalar va nazorat punktlaridagi AKT vositalarini konfiguratsiyasi va ekspluatatsiyasi hamda apparat-dasturiy xizmat koʻrsatish yoʻnalishida ofitser kadrlarga boʻlgan ehtiyoj juda ham yuqori.

Mazkur fan tanlov fanlar bloki tarkibiga kirib, 1 ta semestr (7 semestr) davomida oʻqitiladi.

Kursantlar ushbu fanni oʻzlashtirish uchun quyidagi koʻnikmalarga ega boʻlishi kerak:

- algoritm tushunchasi va xossalari;

- dasturlash texnologiyalari turlari;

- chiziqli, tarmoqlanuvchi va takrorlanuvchi jarayonlar;

- zamonaviy axborot texnologiyalari vositalari arxitekturasi;

- kompyuterning apparat va dasturiy ta’minoti xususiyatlari.

Ushbu fanni oʻrganishda “Informatika”, “Dasturlash texnologiyalari”, “Ma’lumotlar bazasini boshqarish tizimlari” kabi fanlar nazariy zamin boʻlib xizmat qiladi, hamda ushbu fanning oʻzi “Ma’lumotlar bazasi va web texnologiyalar” fani uchun nazariy zamin boʻlib xizmat qiladi.

**2. Oʻquv fanining maqsadi va vazifasi**

Oʻquv fanini oʻqitishdan asosiy **maqsad** turli sohalarda axborot texnologiyalari boʻyicha yuqori malakali ofitserlarini tayyorlashda zamonaviy AKT vositalari va dasturiy ta’minoti tuzilishi, ishlash jarayoni hamda yaratilish bosqichlarini nazariy va amaliy jihatdan oʻrgatishdan iborat.

«Pyton dasturlash tili» fanini **vazifasi**ga kursantlarda mutaxassislik va kasbiy koʻnikmalarni shakllantirish va quyidagi maqsadlarga erishish kiradi:

kursant hamda tinglovchilarda AKT vositalari va dasturiy ta’minoti tuzilishida zamonaviy qurilmalarning dasturiy ta’minoti, ulardan foydalanish, shuningdek, ularning zamonaviy imkoniyatlaridan foydalanishga oid tayyorgarlikni shakllantirish;

ijodiy ravishda mustaqil bilim olish koʻnikma, malakalarni hosil qilish, ularni Oʻzbekiston Respublikasi qoʻshinlarida jangovar tayyorgarlik va xavfsizlikni mustahkamlash hamda obyektga yoʻnaltirilgan dasturlash hamda AKT vositalarining dasturiy vositalaridan samarali foydalanishga yoʻnaltirish.

Amaliy koʻnikma va malakalar hosil qilish: AKT va kompyuter vositalari hamda turli sohalardagi zamonaviy xavfsizlik qurilmalarining dasturiy va texnik ta’minotidan foydalangan holda harbiy maqsadlar uchun dasturiy ishlanmalar yaratish, harbiy texnik va dasturiy tizimlarga oid bilim va malakalarni shakllantirish, zamonaviy texnik qurilmalar, dasturiy vositalar hamda ular bilan ishlash, takomillashtirish, raqamli qurilmalarda ishlash koʻnikmasi va malakalarini oʻzlashtirish.

Fanni oʻzlashtirish davomida va yakunida kursantlar quyidagi koʻnikmalarga ega boʻladi**:**

* dasturlash qismida dasturlash tillarining tuzilmasi, funksiyalari va asosiy parametrlari;
* texnik qoʻriqlash qurilmalarining dasturiy ta’minoti va obyektga yoʻnaltirilgan dasturlash tushunchalari;
* sinflar, tuzilma va birlashmalar;
* dasturiy ta‘minot konfiguratsiyasini boshqarish;
* dasturiy ta‘minotni testlash va sifatini ta‘minlash usullari toʻg‘risida; ***tasavvurga ega boʻlish;***
* qoʻyilgan masalaga mos algoritmlarni tanlash;
* dastur strukturasini ishlab chiqish;
* dasturda xatoliklarni bartaraf etish va boshqarish;
* grafik foydalanuvchi interfeysini shakllantirish va boshqarish;
* belgilangan obyekt hususiyatlari asosida interfeys ishlab chiqish;
* sinf, tuzilma va birlashmalar bilan ishlashni ***bilishi va ulardan foydalana (qoʻllay) olish;***

* zamonaviy dasturlash tillarining asoslarini va dasturiy muhitlarni tadbiq qilish;
* dasturlash tillarning sodda va murakkab tuzilmalarini qoʻllash;
* algoritmlarni baholash, qoʻyilgan masalani yechish algoritmini tanlash, tanlovni asoslash va algoritmni tadbiq etish;
* obyektga moʻljallangan dasturlash texnologiyalaridan ***foydalanish koʻnikmalariga ega boʻlish.***

**3. Oʻquv fanining mazmuni**

**3.1. Ma’ruza mashgʻulotlari:**

**4-kurs 7-semestr**

**1-Mavzu: “Python dasturlash tili” faniga kirish va asosiy tushunchalari.**

**1-mashg‘ulot. Python dasturlash tilining klassifikatsiyasi va rivojlanish tarixi. Python dasturlash tilining asosiy tushunchalari..**

Fanning mazmuni, maqsadi, vazifalari. Python dasturlash tilining avzalliklari. Pythonni o‘rnatish. PyCharm dasturini o‘rnatish. Python sintaksisi bilan tanishish. Pythonda “Hello world!” dasturini tuzish.

**2-Mavzu: PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.**

**1-mashg‘ulot. PyQt5 paketi va uning imkoniyatlari bilan tanishish.**

PyQt5 paketining imkoniyatlari. PyQt5 paketini o‘rnatish. PyQt5 paketini pip yordamida o‘rnatish. QtDesigner dasturini o‘rnatish hamda uning imkoniyatlari bilan tanishish.

**3-Mavzu: Pythonda tarmoq dasturlashga kirish.**

**1-mashg‘ulot. Tarmoqda ma’lumot almashuvchi klient-server dasturini tuzish.**

Socket modulining asosiy metodlari bilan tanishish.

**3.2. Amaliy mashgʻulotlar uchun quyidagi mavzular tavsiya etiladi:**

**4-kurs 7-semestr**

**1-Mavzu: “Python dasturlash tili” faniga kirish va asosiy tushunchalari:**

**2-mashg‘ulot. Pythonda arifmetik operatorlar.**

Arifmetik operatorlar. Sonlar bilan ishlash. O‘zgaruvchilar. Bool tipli ma’lumotlar bilan ishlash.

**3-mashg‘ulot. Satrlar bilan ishlovchi operatorlar va metodlar.**

Satrlar bilan ishlovchi operatorlar va metodlar. str.format() metodi yordamida satrlarni formatlash.

**4-mashg‘ulot. Shart operatori. Shart operatoriga doir dasturlari tuzish.**

IF, IF-ELSE va IF-ELIF-ELSE operatorlari..

**5-mashg‘ulot. Pythonda takrorlanuvchi jarayonlarni dasturlash.**

Sikl operatorlari – For va while bilan ishlash. Break, continue va else operatorlarining qo‘llanilishi.

**6-mashg‘ulot. Pythonda roʻyxatlar bilan ishlash.**

Ro‘yxatlar va ularning qo‘llanilishi. Ro‘yxatlarni yaratish usullari. Ro‘yxatlar bilan ishlovchi metodlar.

**7-mashg‘ulot. Pythonda Kortejlar (Tupllar) bilan ishlash.**

Kortejlar va ularning qo‘llanilishi. Kortejlarni yaratish usullari. Kortejlar bilan ishlovchi metodlar.

**8-mashg‘ulot. Pythonda Setlar ro‘yxati bilan ishlash**

Setlar va ularning qo‘llanilishi. Setlarni yaratish usullari. Setlar bilan ishlovchi metodlar.

**9-mashg‘ulot. Pythonda “Lug‘at” bilan ishlash.**

Lug‘atlar va ularning qo‘llanilishi. Lug‘atlarni yaratish usullari. Lug‘atlar bilan ishlovchi metodlar.

**10-mashg‘ulot. Pythonda Funksiya tushunchasi. Foydalanuvchi funksiyasi..**

Funksiyalarni aniqlash va uni chaqirish. Parametrli va parametrsiz funksiya. Anonim funksiya. Dekoratorlar. Global va lokal o‘zgaruvchi. Lambda funktsiyasi.

**11-mashg‘ulot. Fayllar va kataloglar bilan ishlash.**

Faylni ochish. Fayllar bilan ishlovchi metodlar. os modulining imkoniyatlari. Fayl va katalog yo‘lini o‘zgartirish. Katalog va fayl bilan ishlovchi funksiya va metodlar.

**12-mashg‘ulot. Pythonda OOP asoslari.**

OOP asoslari. Klasslarni e’lon qilish va nusxasini yaratish. Sinf va obyekt. Sinf konstruktori. \_\_init\_\_() va \_\_del\_\_() metodlari.

**13-mashg‘ulot. Pythonda Vorislik tushunchasi.**

Vorislik. Maxsus metodlar. Klass xususiyatlari.Dekoratorlar.

**2-Mavzu: PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish:**

**2-mashg‘ulot. PyQt5 kutubxonasi. QLabel va QLineEdit vidjetlari.**

Arduinoda ma’lumotlar butunligi tushunchasi.CRC ma’lumotlar butunligi..

**3-mashg‘ulot. Mikrokontroller o‘rtasida o‘zaro ma’lumot almashinuvi.**

QLabel vidjeti;QLabel shrift, o‘lcham va text xususiyatlari;QLineEdit vidjeti;

setStyleSheet() metodi.

**4-mashg‘ulot. PyQt5 modal dialog. QMessageBox vidgeti bilan ishlash.**

QMessageBox vidgetining vazifasi. QMessageBox vidgetining asosiy xususiyatlari. Statik funksiyalari. Piktogramma va Pixmap xususiyatlari.

**5-mashg‘ulot. PyQt da rasmlar va menyular.**

PyQt paketi yordamida rasm joylashtirish usullari. Menu yaratish. Menu vidgetining xususiyatlari.

**6-mashg‘ulot. Matn muharriri dizaynini yaratish.**

Yaratiladigan matn muharriri uchun kerakli uskunalarni tanlash. Tanlangan uskunalarni matn muharriri ekraniga joylashtirish. Matn muharririr ekrani dizaynini shakllantirish.

**7-mashg‘ulot. Matn muharriri dasturini yozish.**

PyQt5 da matn muharriri dizaynidagi elementlarning funksiyalari uchun dastur yozish. Dasturni testlash.

**8-mashg‘ulot. PyQt5 da Minesweeper o‘yini dizaynini yaratish.**

PyQt5 da Minesweeper o‘yinini yaratish. O‘yin uchun kerakli uskunalarni tanlash.

O‘lchamlarni o‘rnatish. O‘yin dizaynini yaratish.

**9-mashg‘ulot. PyQt5 da Minesweeper o‘yini dasturini yozish.**

PyQt5 da Minesweeper o‘yini dizaynidagi tugmalarning funksiyalari uchun dastur yozish. Dasturni testlash. O‘yinni ishga tushirish.

**3-Mavzu: Pythonda tarmoq dasturlashga kirish:**

**2-mashg‘ulot. Socket moduli bilan ishlash.**

Socket modulining asosiy metodlari bilan tanishish. socket(), .bind, .listen, .accept(), .connect(), .send(), recv(), .close() metodlari bilan ishlash.

**3-mashg‘ulot. Pythonda TCP klient-server dasturini tuzish.**

Socket modulining asosiy metodlari yordamida client-server dasturini tuzish.

**4-mashg‘ulot. TCP klient-server dasturini testlash.**

TCP client-server dasturi yordamida ma’lumot almashish.

**5-mashg‘ulot. PyQt5 paketidan foydalanib zamonaviy chat dasturini tuzish.**

TCP dasturini client qismini GUI ko‘rinishda ishlab chiqish.

**6-mashg‘ulot. Pythonda GUI paketidan foydalanib chat dasturini tuzishni yakunlash**

TCP client-server dasturini GUI ko‘rinishda ishlab chiqish.

**7-mashg‘ulot. PyQt5 paketidan foydalanib zamonaviy chat dasturini tuzish.**

GUI TCP klient-server dasturi yordamida ma’lumot almashinuvini testlash.

**8-mashg‘ulot. Python ilovasini kompilyatsiya qilish.**

Python ilovasini kompilyatsiya qilish.

O‘qituvchilar amaliy mashg‘ulotlarni o‘tkazishda, tinglovchilarning yakka tartibdagi sifatlariga ko‘proq javob beradigan va o‘quv materiallarini ular tomonidan yuqori darajada o‘zlashtirishni ta’minlaydigan, shuningdek, mustaqil va ijodiy fikrlashni rivojlantiradigan o‘qitish usul va vositalarini tanlaydilar.

**Mashg‘ulotlar o‘quv soati hisoboti va mavzular ro‘yxati**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mavzular** | **Fan mavzularini nomlanishi** | | **Ishlar hajmi (soat)** | | | | | | |
| **Umumiy yuklama hajmi** | **Auditoriyadagi ish** | | | | | |
| **Jami** | **Ma’ruza** | **Amaliy mashg‘ulotlar** | **Guruh mashg‘uloti** | **Seminarlar** | **Mustaqil o‘qish** |
| **O‘rnatilgan tizimlarning apparat – dasturiy ta’minoti** | | | | | | | | | |
| **4-kurs 7 semestr** | | | | | | | | | |
|  | | “Python dasturlash tili” faniga kirish va asosiy tushunchalari. Python dasturlash tilining klassifikatsiyasi va rivojlanish tarixi. Python dasturlash tilining asosiy tushunchalari | 4 | 2 | 2 |  |  |  | 2 |
|  | | Pythonda arifmetik operatorlar. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Satrlar bilan ishlovchi operatorlar va metodlar. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Shart operatori. Shart operatoriga doir dasturlari tuzish. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Pythonda takrorlanuvchi jarayonlarni dasturlash. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Pythonda roʻyxatlar bilan ishlash. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Pythonda Kortejlar (Tupllar) bilan ishlash. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Pythonda Setlar ro‘yxati bilan ishlash. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Pythonda “Lug‘at” bilan ishlash. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Pythonda Funksiya tushunchasi. Foydalanuvchi funksiyasi | 4 | 2 |  | 2 |  |  | 2 |
|  | | Fayllar va kataloglar bilan ishlash. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Pythonda OOP asoslari. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Pythonda Vorislik tushunchasi. | 4 | 2 |  | 2 |  |  | 2 |
|  | | PyQt5 paketi va uning imkoniyatlari bilan tanishish. | 4 | 2 | 2 |  |  |  | 2 |
|  | | PyQt5 kutubxonasi. QLabel va QLineEdit vidjetlari. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Mikrokontroller o‘rtasida o‘zaro ma’lumot almashinuvi. | 4 | 2 |  | 2 |  |  | 2 |
|  | | PyQt5 modal dialog. QMessageBox vidgeti bilan ishlash. | 4 | 2 |  | 2 |  |  | 2 |
|  | | PyQt da rasmlar va menyular. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Matn muharriri dizaynini yaratish. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Matn muharriri dasturini yozish. | 4 | 2 |  | 2 |  |  | 2 |
|  | | PyQt5 da Minesweeper o‘yini dizaynini yaratish | 4 | 2 |  | 2 |  |  | 2 |
|  | | PyQt5 da Minesweeper o‘yini dasturini yozish. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Tarmoqda ma’lumot almashuvchi klient-server dasturini tuzish. | 4 | 2 | 2 |  |  |  | 2 |
|  | | Socket moduli bilan ishlash. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Pythonda TCP klient-server dasturini tuzish. | 4 | 2 |  | 2 |  |  | 2 |
|  | | TCP klient-server dasturini testlash. | 4 | 2 |  | 2 |  |  | 2 |
|  | | PyQt5 paketidan foydalanib zamonaviy chat dasturini tuzish.. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Pythonda GUI paketidan foydalanib chat dasturini tuzishni yakunlash | 4 | 2 |  | 2 |  |  | 2 |
|  | | PyQt5 paketidan foydalanib zamonaviy chat dasturini tuzish. | 4 | 2 |  | 2 |  |  | 2 |
|  | | Python ilovasini kompilyatsiya qilish. | 4 | 2 |  | 2 |  |  | 2 |
|  | | **Jami fan bo‘yicha:** | **120** | **60** | **6** | **54** |  |  | **60** |

**4. Fanni oʻqitish boʻyicha tashkiliy – uslubiy koʻrsatmalar.**

“Pyton dasturlash tili” fanini oʻqitish davomida kursantlarni mustaqil va erkin fikr yuritishga, mantiqiy va algoritmik fikrlashlarini hamda, nutq mahoratini oshirishga, u yoki bu muammoga nisbatan oʻz nuqtai nazarini aniq va ravshan ifoda etishga chorlaydigan innovatsion pedagogik texnologiyalardan hamda “Bumerang”, “Zinama-zina”, “Fikrlar hujumi” (aqliy hujum), “Charxpalak”, “3 x 4”, “Muammo”, “Labirint”, “Blis-soʻrov”, “Skorobey”, “Interfaol suhbat”, “T-sxema”, “Klasster”, “FSMU”, “VEN-diagramma”, SWOT-tahlil” va boshqa interfaol metodlardan foydalaniladi.

Ma’ruza materiallari bayoni mustaqil va tugallangan hususiyatga ega boʻlib, avval bayon qilingan materiallarga mantiqiy bogʻlangan hamda boshqa fanlarda, hamda amaliyotda qoʻllanishga yoʻnaltirilgan boʻlishi kerak. Amaliy mashgʻulotlarda kursantlar olgan nazariy bilimlarini qoʻllay olishni oʻrganishlari kerak.

Har bir ma’ruza oʻz ichiga kirish, asosiy va yakuniy qismni oladi.

Kirish qismida: mavzuning nomi, ma’ruza mavzusining asosiy gʻoyasi va muhimligi; oʻquv maqsadlar; ma’ruzaning oʻquv savollari; oldingi va keyingi mashgʻulotlar bilan bogʻliqligi; OHTMda ofitserlarni tayyorlash jarayonidagi ma’ruzaning tutgan oʻrni bayon qilinadi.

Ma’ruzaning asosiy qismida oʻquv savollarining mazmuni yetkaziladi. Ma’ruzaning har bir nazariy jihati eng maqsadga muvofiq usullarni qoʻllagan holda asoslangan va isbotlangan boʻlishi kerak. Ma’ruzaning asosiy qismini bayon qilishda ta’lim oluvchilarga ilmiy gʻoyalarni rivojlanishi, jamlanishi, mavhumlikdan aniqlikka oʻtishining mantigʻini yoritib berishga imkon beruvchi dalillarga tayanish ma’ruzaga boʻlgan majburiy talab hisoblanadi. Har bir ma’ruzaning asosiy qismining mazmuni fundamental boʻlishi kerak.

Amaliy maqsadlarga yoʻnaltirilgan ma’ruzalarda kasbga oid va oʻquv vazifalarni hal etish boʻyicha amaliy tavsiyalarni koʻzda tutish maqsadga muvofiq boʻladi.

Har bir oʻquv savoli, uni, keyingi oʻquv savoliga mantiqiy olib keluvchi, rivojlanish istiqbollarining nazariyasi va amaliyoti hamda qisqacha xulosasini yoritish bilan tugatilishi kerak.

Ma’ruzaning yakuniy qismida, nazariya va amaliyotni qoʻllash soha va chegaralarini koʻrsatgan holda, asosiy qism mazmuni umumlashtiriladi

va qisqacha xulosa qilinadi, mustaqil oʻrganish hamda kelgusi seminar va boshqa turdagi mashg‘ulotlarda muhokama qilish uchun savollar va vazifalar belgilanadi.

Ma’ruzani oʻqishda kino va videofilmlar, chizmalar, plakatlar, modellar, asboblar va maketlarni namoyish qilgan holda oʻquv materiallarining og‘zaki yetkazilishi oʻqitishning yetakchi uslubi hisoblanadi.

Materialni yetkazish tempini tanlashda, oʻqituvchi, ta’lim oluvchilar (tinglovchilar, kursantlar) toifasini, ushbu mavzu (yoʻnalish) boʻyicha oʻquv, ilmiy, uslubiy adabiyotlar mavjudligi va boshqa omillarni albatta hisobga olishi kerak.

Individual va kollektiv yondashish yoʻli bilan oʻqituvchi suhbat orqali ma’ruzaning oʻz ichiga olgan muammoli savollarning yechimini topadi.

Oʻrganilayotgan oʻquv materiallarini faollashtirish uchun «nima uchun bunday qilingan», «qanchalik bu qulay (ma’qullik, maqsadga muvofiq)», bunda oʻrganuvchilar orasida seminar mashg‘ulot xususiyatga ega boʻlgan fikrlarni almashuv va metodik usullarni kiritish foydalidir.

Amaliy mashg‘ulot oʻtkazish maqsadida kursantlar zamonaviy kompyuterlarda zamonaviy dasturlash tillarida dastur yaratishadi va dasturlarni tahlilini oʻrganishadi.

Amaliy mashg‘ulotlar zamonaviy kompyuterlar va multimedia vositalari bilan jihozlangan maxsus oʻquv sinf xonalarida oʻtkaziladi. Nazariy tajribani va amaliyotni oʻtash mobaynida oʻz qobiliyatini hamda koʻnikmalarini takomillashtiradi.

Mashg‘ulotlarni individuallashtirish va oʻqitishni sifatini oshirish maqsadida vositalarning soniga qarab guruhlar bir qancha guruhlarga boʻlinadi va ular oʻquv joylariga taqsimlanadi.

Amaliy mashg‘ulotlarda kursantlar me’yorlarni bajarishda ishtirok etishi maqsadida bellashuv, musobaqa va sog‘lom raqobat elementlarini kiritish lozim.

Oʻquv-tarbiyaviy jarayonini jadallashtirishga qoʻyilgan talablar oshishini inobatga olib mashg‘ulotlarni tashkil etish va oʻtkazish uslubiyatini doimo takomillashtirish lozim.

Mustaqil ta’lim jarayonida kursantlar tavsiya etilgan adabiyotlarni oʻrganib, konspektlarini toʻldirib, olgan bilimlarini mustahkamlaydi.

**5. Mustaqil ta’lim va mustaqil ishlar.**

Mustаqil oʻzlаshtirilаdigаn mаvzulаr boʻyichа belgilangan vaqt davomida fan boʻyicha oʻtkazilgan mavzular va zarur koʻnikma va malakani shakllantirishga undaydigan qoʻshimcha mavzular hamda materiallar ustida kursantlar oʻzi mustaqil oʻrganishadi. Mustaqil ta’lim davomida kursantlar zarur adabiyotlar va elektron manbaalar bilan ta’minlanadi. Kursantlarning mustaqil ta’lim olishi fanni va mutaxassislik koʻnikmalarini yanada mustahkamroq egallashini ta’minlaydi. Mustaqil ta’lim va mustaqil ish topshiriqlarini kursantlar tomonidan bajarilishi majburiydir va u fanning joriy nazorat bahosining bir qismini tashkil etadi. Mustaqil ta’lim topshiriqlari fan oʻqituvchisi tomonidan har bir kursant uchun umumiy bir mavzuda va har biriga individual yoʻnalish va shart asosida semestr davomida berib boriladi.

Mustaqil oʻzlashtiriladigan mavzular boʻyicha kursantlar tomonidan mustaqil ish AKT vositasi yordamida amaliy ish koʻrinishida tayyorlanadi va uni taqdimoti tashkil qilinadi.

**5.1. Mustaqil ta’lim olish uchun tavsiya etiladigan mavzular:**

1. PyQt5 kutubxonasi bilan ishlash
2. Pythonda umumiy masalalarga dior dastur tuzish
3. Pythonda fayllar bilan ishlash
4. Pythonda tarmoq dasturlari yaratish

Mustaqil ta’lim va mustaqil ishning baholanishi har bir kursantning bajargan topshirigʻi sifati va taqdimotiga koʻra aniqlanadi. Mustaqil ta’lim va mustaqil ishning baholash mezonlari fanning ishchi oʻquv dasturi (sillabus) da batafsil yoritilgan.

1. **Asosiy va qoʻshimcha oʻquv adabiyotlar hamda axborot manbaalari**

**Asosiy adabiyotlar:**

1. Sh.R. Sapayev, B.K. Yusupov, A.A.Abidov. “Python dasturlash tili” Darslik. Toshkent: 2024y. В - 316.
2. Sh.R. Sapayev “Python dasturlash tili asoslari”. O‘quv qo‘llanma. Toshkent: 2023y. В – 137.
3. Sh.R. Sapayev “PyQt5 paketi va QtDesigner dasturida grafik ilovalar tuzish”. O‘quv qo‘llanma. Toshkent: 2024y. В - 150

**Qo‘shimcha adabiyotlar:**

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих.-СПб.: Питер, 2017.-288 с. : ил. ISBN 978-5-496-02541-6
2. Н.А.Прохоренок, В.А.Дронов. “Python3 и PyQT5. Разработка приложений”. СПб.: БХВ-Петербург, 2016. – 832 с.: ил.
3. Франсуа Шолле. “Глубокое обучение на Python”. — СПб.: Питер, 2018. — 400 с.: ил. — (Серия «Биб- лиотека программиста»).
4. Чан, Уэсли. “Python: создание приложений. Библиотека профессионала”, 3-е изд. [Пер. с англ. - М. : ООО "И.Д. Вильяме"], Москва: Санкт-Петербург • Киев 2015.
5. Марк Саммерфилд. “Программирование на Python 3. Подробное руководство” [Пер. с англ. – СПб]. - Москва: Санкт-Петербург–2009 год.

**Internet saytlari:**

1. https://www.python.org
2. https://python-scripts.com
3. https://webformyself.com/python

|  |
| --- |
| **Fan/modul uchun ma’sullar:**  B.K. Yusupov – OʻR MV AKT va AHI “Axborot texnologiyalari va dasturiy injiniring” kafedrasi boshlig‘i, PhD, dotsent.  Sh.R. Sapayev - OʻR MV AKT va AHI “Axborot texnologiyalari va dasturiy injiniring” kafedrasi dotsenti. |
| **Taqrizchilar:**  podpolkovnik S. Porsiyev – O‘R QK Bosh Shtabi AAT va AH BB axborot-kommunikatsiya texnologiyalarini rivojlantirish boshqarmasi boshlig‘i VVB.  podpolkovnik B. To‘rayev **-** O‘R QK Akademiyasi Qurolli Kuchlarda Axborot texnologiyalari va kiberxavfsizlik kafedrasi boshlig‘i LMVB. |

**O‘ZBEKISTON RESPUBLIKASI MUDOFAA VAZIRLIGI**

**AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI VA ALOQA HARBIY INSTITUTI**

KIBERXAVFSIZLIK FAKULTETI

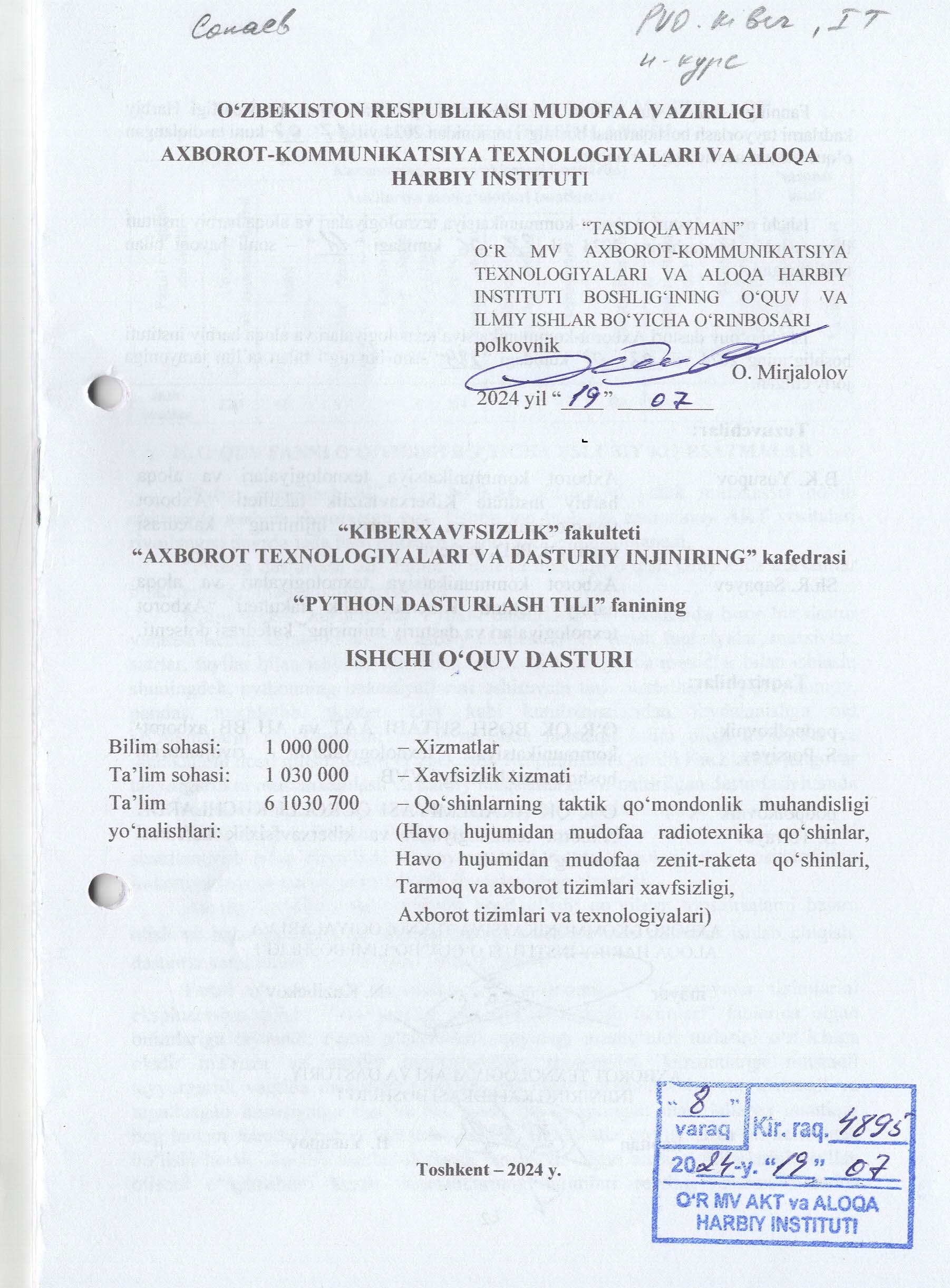
“AXBOROT TEXNOLOGIYALARI VA DASTURIY INJINIRING”

KAFEDRASI

**“PYTHON DASTURLASH TILI” fanining**

ISHCHI O‘QUV DASTURI

**Toshkent 2025 y.**



**I. O‘QUV VAQTINI SEMESTRLAR VA O‘QUV MASHG‘ULOT**

**TURLARI BO‘YICHA TAQSIMLANISHI**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Fanni o‘qitish muddati:** | **Kursantning o‘quv yuklamasi (soatlarda)** | | | | | | | | | | **Nazorat usuli** | |
| **Umumiy yuklamaning hajmi** | **Auditoriya mashg‘ulotlari (soatlarda)** | | | | | | | | **Mustaqil tayyorgarlik** |
| **Jami** | **Ma’ruzalar** | **Guruh mashg‘ulotlari (mashqlari)** | **Amaliy mashg‘ulotlar** | **Laboratoriya mashg‘ulotlari** | **Seminarlar** | **Mustaqil ta’lim** | **Kurs loyihasi (ishi)** | **Oraliq nazorati** | **Yakuniy nazorat** |
| **7** | 120 | 60 | 6 |  | 54 |  |  | 60 |  |  | + | + |
| **Jami soatlar** | **120** | **60** | **6** |  | **54** |  |  | **60** |  |  |  |  |

**II. O‘QUV FANNI O‘QITILISH BO‘YICHA USLUBIY KO‘RSATMALAR**

“Python dasturlash tili” fani boʻlajak ofitserlarni yetuk mutaxassis boʻlib chiqishida muhim ahamiyatga ega. Ushbu fan bugungi zamonaviy AKT vositalari rivojlangan davrda juda ham **dolzarb** ahamiyatga ega hisoblanadi.

“Python dasturlash tili” fanini o‘tish va mustaqil o‘qish jarayonida kursantlar bilan quyidagi maqsadlarga erishiladi:

Bilim berish: kursantlarda Python dasturlash tili yordamida biror bir dastur yaratish uchun kerak bo‘ladigan asosiy elementlari bo‘lmish funksiyalar, massivlar, satrlar, fayllar bilan ishlash, ular bilan ishlovchi funksiya va metodlar bilan ishlash; shuningdek, pythonning imkoniyatlarini oshiruvchi turli modullari - PyQt5, numpy, pandas, matplotlib, tkinter, kivy kabi kutubxonalaridan foydalanishga oid ko‘nikmalarni shakllantirish; ijodiy ravishda mustaqil bilim olish ko‘nikma va malakalarni hosil qilish; ularni O‘zbekiston Respublikasi Qurolli Kuchlarida jangovar tayyorgarlikni mustahkamlash va harbiy maqsadlarga yo‘naltirilgan dasturlash hamda AKT vositalarining dasturiy vositalaridan samarali foydalanishga yo‘naltirish.

Tarbiyalash: ofitserlar faoliyatiga xos bo‘lgan kasbiy va psixologik sifatlarni shakllantirish bilan birgalikda kompyuter va Axborot texnologiyalari vositalarining imkoniyatlaridan harbiy maqsadlarda foydalanishga o‘rgatish.

Amaliy ko‘nikma va malakalar hosil qilish: qo‘yilgan topshiriqlarni bajara olish va bajarish usullarini tanlash; berilgan talabga mos dasturlar ishlab chiqish; dasturda xatoliklarni bartaraf etish va boshqarish.

Fanni o‘zlashtirish kursantlarning “Informatika”, “Kompyuter tizimlarini ekspluatasiya qilish” “Ma’lumotlar bazasini boshqarish tizimlari” fanlarida olgan bilimlariga tayanadi. Fanni o‘zlashtirish quyidagi mashg‘ulot turlarini o‘z ichiga oladi: ma’ruza va amaliy mashg‘ulotlar, shuningdek kursantlarga mustaqil tayyorgarlik vaqtida maslahatlar berish. Ma’ruza materiallari bayoni mustaqil va tugallangan hususiyatga ega bo‘lib, avval bayon qilingan materiallarga mantiqiy bog‘langan hamda boshqa fanlarda, hamda amaliyotda qo‘llanishga yo‘naltirilgan bo‘lishi kerak. Amaliy mashg‘ulotlarda kursantlar olgan nazariy bilimlarini qo‘llay olishni o‘rganishlari kerak. Kursantlarning bilimlari reyting nazorat tizimida baholanadi. Kursantlar bilimlarini reyting nazoratida baholash quyidagi tartibda amalga oshiriladi:

- kundalik nazorat: kursantlarni mashg‘ulotlarda muntazam so‘roq qilish;

- oraliq nazorat;

- yakuniy nazorat.

Fanning harbiy yo‘nalishi aloqa qo‘shinida mavjud va mutaxassislarning keyingi professional faoliyatiga ta’luqli aniq texnika baza namunalarini tuzilishini va ekspluatatsiya qilishda amaliy bilimlar olish bilan ta’minlanadi.

Mashg‘ulotning asosiy ko‘rinishi ma’ruza mashg‘ulotlari va amaliy mashg‘ulotlar hisoblanadi.

Ma’ruza mashg‘ulotlari, odatda, bir necha o‘quv guruhini o‘z ichiga olgan,   
100 nafardan ortiq bo‘lmagan kursantlarning oqimi (potok) bilan o‘tkaziladi. Ma’ruza kafedra boshlig‘i va professor-o‘qituvchilar tomonidan o‘qiladi. Ma’ruzani o‘qishga tajribali o‘qituvchilar ham ruxsat etiladi. Ma’ruzani o‘qish uslubi ma’ruzachi tomonidan aniqlanadi, lekin bunda mashg‘ulotda o‘rganuvchilarning faolligini oshirish usullariga ko‘proq e’tibor qaratiladi:

* muammoli savollarni o‘rtaga qo‘yish;
* ma’ruzani munozara usulida muloqat formasida harbiy tajribaga va o‘rganilayotgan texnika namunalari jangovar qo‘llash hamda amaliy ekspluatatsiya qilishga tayangan holda o‘qitish.

Ma’ruzaning materiallari doimo yangilanib borish lozim. Ma’ruza o‘rganilayotgan fan bo‘yicha ilmiy bilimlar asosini berishi, o‘quv materiallarining o‘ta murakkab savoli dialektik o‘zaro bog‘liqligini, kursantlarning ijodiy fikrlashini rivojlanishiga ko‘maklashuvi, zamonaviy ilm va texnika yutuqlarini, nazariya va amaliyotning dolzarb savollarini yoritib berishi va boshqa mashg‘ulotlar turlarining hamda kursantlarning mustaqil tayyorgarligini tashkil qilish va o‘tkazishning asosi hisoblanadi.

Ma’ruza mashg‘ulotlarining faol shakllari:

* tasviriy (vizual) ma’ruza;
* muammoli ma’ruza;
* ma’ruza – matbuot konferensiyasi;
* ikki kishilik ma’ruza;
* ma’ruza – provokatsiya (chalg‘ituvchi ma’ruza);
* ma’ruza – konsultatsiya;
* ma’ruza – suhbat;
* qarshi aloqa texnikasidan foydalanuvchi ma’ruza.

Har bir ma’ruza o‘z ichiga kirish, asosiy va yakuniy qismni oladi.

Kirish qismida: mavzuning nomi, ma’ruza mavzusining asosiy g‘oyasi   
va muhimligi; o‘quv maqsadlar; ma’ruzaning o‘quv savollari; oldingi va keyingi mashg‘ulotlar bilan bog‘liqligi.

Ma’ruzaning asosiy qismida o‘quv savollarining mazmuni yetkaziladi. Ma’ruzaning har bir nazariy jihati eng maqsadga muvofiq usullarni qo‘llagan holda asoslangan va isbotlangan bo‘lishi kerak. Ma’ruzaning asosiy qismini bayon qilishda ta’lim oluvchilarga ilmiy g‘oyalarni rivojlanishi, jamlanishi, mavhumlikdan aniqlikka o‘tishining mantig‘ini yoritib berishga imkon beruvchi dalillarga tayanish ma’ruzaga bo‘lgan majburiy talab hisoblanadi. Har bir ma’ruzaning asosiy qismini mazmuni fundamental bo‘lishi kerak.

Amaliy maqsadlarga yo‘naltirilgan ma’ruzalarda kasbga oid va o‘quv vazifalarni hal etish bo‘yicha amaliy tavsiyalarni ko‘zda tutish maqsadga muvofiq bo‘ladi.

Har bir o‘quv savoli, uni, keyingi o‘quv savoliga mantiqiy olib keluvchi, rivojlanish istiqbollarining nazariyasi va amaliyoti hamda qisqacha xulosasini yoritish bilan tugatilishi kerak.

Ma’ruzaning yakuniy qismida, nazariya va amaliyotni qo‘llash soha   
va chegaralarini ko‘rsatgan holda, asosiy qism mazmuni umumlashtiriladi   
va qisqacha xulosa qilinadi, mustaqil o‘rganish hamda kelgusi seminar va boshqa turdagi mashg‘ulotlarda muhokama qilish uchun savollar va vazifalar belgilanadi.

Ma’ruzani o‘qishda kino va videofilmlar, chizmalar, plakatlar, modellar, asboblar va maketlarni namoyish qilgan holda o‘quv materiallarining og‘zaki yetkazilishi o‘qitishning yetakchi uslubi hisoblanadi.

Materialni yetkazish tempini tanlashda, o‘qituvchi, kursantlar toifasini, ushbu mavzu (yo‘nalish) bo‘yicha o‘quv, ilmiy, uslubiy adabiyotlar mavjudligi va boshqa omillarni albatta hisobga olishi kerak.

Individual va kollektiv yondashish yo‘li bilan o‘qituvchi suhbat orqali ma’ruzaning o‘z ichiga olgan muammoli savollarning yechimini topadi.

O‘rganilayotgan o‘quv materiallarini faollashtirish uchun «nima uchun bunday qilingan», «qanchalik bu qulay (ma’qullik, maqsadga muofiq)», bunda o‘rganuvchilar orasida seminar mashg‘ulot xususiyatga ega bo‘lgan fikrlarni almashuv va metodik usullarni kiritish foydalidir.

Amaliy mashg‘ulot o‘tkazish maqsadida kursantlar zamonaviy kompyuterlarda Python va C++ dasturlash tillarida dastur yaratishadi va dasturlarni tahlilini o‘rganishadi.

Amaliy mashg‘ulotlar zamonaviy kompyuterlar va multimedia vositalari bilan jihozlangan maxsus o‘quv sinf xonalarida o‘tkaziladi. Nazariy tajribani va amaliyotni o‘tash mobaynida o‘z qobiliyatini hamda ko‘nikmalarini takomillashtiradi.

Mashg‘ulotlarni individuallashtirish va o‘qitishni sifatini oshirish maqsadida vositalarning soniga qarab guruhlar bir qancha guruhlarga bo‘linadi va ular o‘quv joylariga taqsimlanadi.

Amaliy mashg‘ulotlarda kursantlar me’yorlarni bajarishda ishtirok etishi maqsadida bellashuv, musobaqa va sog‘lom raqobat elementlarini kiritish lozim.

Harbiy tajribani va amaliyotni o‘tash mobaynida o‘z qobiliyat hamda ko‘nikmalarni takomillashtiriladi.

O‘quv-tarbiyaviy jarayonini jadallashtirishga qo‘yilgan talablar oshishini inobatga olib mashg‘ulotlarni tashkil etish va o‘tkazish uslubiyatini doimo takomillashtirish lozim.

Mustaqil o‘qish soatida kursantlar tavsiya etilgan adabiyotlarni o‘rganib, konspektlarini to‘ldirib, olgan bilimlarini mustahkamlaydi.

Guruh va yakka tartibdagi konsultatsiyalar kursantlarga o‘qituvchilar tomonidan amaliy mashg‘ulotlarga va imtihonlarga yordam ko‘rsatish maqsadida o‘tkaziladi.

Kursantlarni bilimini aniqlash joriy va yakuniy nazoratlar baholari orqali amalga oshiriladi. Joriy nazorat o‘tkazish, kursantlar o‘quv materiallarni sifatli o‘zlashtirishlarini to‘liq tekshirish va ularni ishini rag‘batlantirish uchun amalga oshiriladi. U amaliy mashg‘ulotlar davomida o‘tkaziladi.

Kursantlarning bilimlarini tekshirish to‘rt balli tizimida amalga oshiriladi. Kursantlar bilim darajasining nazorati quyidagi ko‘rinishda amalga oshiriladi:

Joriy nazorat – mashg‘ulot jarayonida doimiy tizimli ravishda **savol-javob, test, amaliy ish usullari bilan olib boriladi**.

Yakuniy nazorat kursantlarni nazariy bilimlari va amaliy tayyorgarligi darajasini tekshirish maqsadida o‘tkaziladi. Sinov va imtihon orqali amalga oshiriladi.

Fan bo‘yicha kursantlarning bilim, ko‘nikma va malakalariga quyidagi talablar qo‘yiladi. Kursant:

**Kursant bilimlarga ega bo‘lishi:**

* python dasturlash tilining tuzilmasi, funksiyalari va asosiy parametrlari;
* PyQt5 paketi imkoniyatlari va vidgetlar bilan ishlash;
* tarmoqda dasturlash asoslari.

**Kursant ko‘nikma va malakalarini egallashi:**

* qo‘yilgan masalaga mos algoritmlarni tanlash;
* PyQt5 paketi vidgetlaridan turli dasturlar ishlab chiqish;
* dasturda xatoliklarni bartaraf etish va boshqarish;
* grafik foydalanuvchi interfeysini shakllantirish va boshqarish.

**Kursant kompetensiyalarni egallashi lozim:**

* python dasturlash tilining sodda va murakkab tuzilmalarini qo‘llash;
* algoritmlarni baholash, qo‘yilgan masalani yechish algoritmini tanlash, tanlovni asoslash va algoritmni tadbiq etish;
* obyektga mo‘ljallangan dasturlash texnologiyalaridan foydalanish;
* tarmoqda dasturlash asoslarini hayotga tadbiq qilish va rivojlantirish.

**III. FANNI O‘QUV MASHG‘ULOT TURLARI BO‘YICHA**

**O‘QITISH REJASI**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **T.r.** | **O‘quv mashg‘ulot raqami va turlari** | **Soatlar hajmi** | **Mashg‘ulot mavzusi va o‘quv savollari** | **Mashg‘ulotning moddiy jihatdan ta’minlanishi** |
| **7- semestr** | | | | |
| 1. | Ma’ruza | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **1-mashg‘ulot.** Python dasturlash tilining klassifikatsiyasi va rivojlanish tarixi. Python dasturlash tilining asosiy tushunchalari.  O‘quv savollari:   1. Fanning mazmuni, maqsadi, vazifalari; 2. Pythonni o‘rnatish. PyCharm dasturini o‘rnatish; 3. Pythonda “Hello world!” dasturini tuzish. 4. Python tilining asosiy operatorlari bilan tanishish | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 2. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **2-mashg‘ulot.** Pythonda arifmetik operatorlar.  O‘quv savollari:   1. Arifmetik operatorlar. 2. Sonlar bilan ishlash. O‘zgaruvchilar. 3. Bool tipli ma’lumotlar bilan ishlash. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 3. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **3-mashg‘ulot.** Satrlar bilan ishlovchi operatorlar va metodlar.  O‘quv savollari:   1. Satrlar bilan ishlovchi operatorlar va metodlar. 2. str.format() metodi yordamida satrlarni formatlash. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 4. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **4-mashg‘ulot.** Shart operatori. Shart operatoriga doir dasturlari tuzish.  O‘quv savollari:   1. IF, IF-ELSE va IF-ELIF-ELSE operatorlari. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 5. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **5-mashg‘ulot.** Pythonda takrorlanuvchi jarayonlarni dasturlash.  O‘quv savollari:   1. Sikl operatorlari – For va while bilan ishlash. 2. Break, continue va else operatorlarining qo‘llanilishi. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 6. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **6-mashg‘ulot.** Pythonda roʻyxatlar bilan ishlash.  O‘quv savollari:   1. Ro‘yxatlar va ularning qo‘llanilishi. 2. Ro‘yxatlarni yaratish usullari. 3. Ro‘yxatlar bilan ishlovchi metodlar. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 7. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **7-mashg‘ulot.** Pythonda Kortejlar (Tupllar) bilan ishlash.  O‘quv savollari:   1. Kortejlar va ularning qo‘llanilishi. 2. Kortejlarni yaratish usullari. 3. Kortejlar bilan ishlovchi metodlar. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 8. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **8-mashg‘ulot.** Pythonda Setlar ro‘yxati bilan ishlash.  O‘quv savollari:   1. Setlar va ularning qo‘llanilishi. 2. Setlarni yaratish usullari. 3. Setlar bilan ishlovchi metodlar. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 9. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **9-mashg‘ulot.** Pythonda “Lug‘at” bilan ishlash.  O‘quv savollari:   1. Lug‘atlar va ularning qo‘llanilishi. 2. Lug‘atlarni yaratish usullari. 3. Lug‘atlar bilan ishlovchi metodlar. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 10. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **10-mashg‘ulot.** Pythonda Funksiya tushunchasi. Foydalanuvchi funksiyasi..  O‘quv savollari:   1. Funksiyalarni aniqlash va uni chaqirish. 2. Parametrli va parametrsiz funksiya. 3. Anonim funksiya. Dekoratorlar. Global va lokal o‘zgaruvchi. 4. Lambda funktsiyasi. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 11. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **11-mashg‘ulot**. Fayllar va kataloglar bilan ishlash.  O‘quv savollari:   1. Faylni ochish. Fayllar bilan ishlovchi metodlar. 2. os modulining imkoniyatlari. 3. Fayl va katalog yo‘lini o‘zgartirish. 4. Katalog va fayl bilan ishlovchi funksiya va metodlar. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 12. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **12-mashg‘ulot.** Pythonda OOP asoslari.  O‘quv savollari:   1. OOP asoslari. Klasslarni e’lon qilish va nusxasini yaratish. 2. Sinf va obyekt. Sinf konstruktori. 3. \_\_init\_\_() va \_\_del\_\_() metodlari. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 13. | Amaliy | 2 | **1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.  **13-mashg‘ulot**. Pythonda Vorislik tushunchasi.  O‘quv savollari:   1. Vorislik. 2. Maxsus metodlar. 3. Klass xususiyatlari. 4. Dekoratorlar. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 14. | Ma’ruza | 2 | **2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.  **1-mashg‘ulot.** PyQt5 paketi va uning imkoniyatlari bilan tanishish.  O‘quv savollari:   1. PyQt5 paketining imkoniyatlari. PyQt5 paketini o‘rnatish. 2. PyQt5 paketini pip yordamida o‘rnatish. 3. QtDesigner dasturini o‘rnatish hamda uning imkoniyatlari bilan tanishish. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 15. | Amaliy | 2 | **2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.  **2-mashg‘ulot.** PyQt5 kutubxonasi. QLabel vidjeti.  O‘quv savollari:   1. QLabel vidjeti; 2. QLabel shrift, o‘lcham va text xususiyatlari; | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 16. | Amaliy | 2 | **2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish**.**  **3-mashg‘ulot.** PyQt5 kutubxonasi. QLineEdit vidjeti.  O‘quv savollari:   1. QLineEdit vidjeti; 2. setStyleSheet() metodi. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 17. | Amaliy | 2 | **2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.  **4-mashg‘ulot.** PyQt5 modal dialog. QMessageBox vidgeti bilan ishlash.  O‘quv savollari:   1. QMessageBox vidgetining vazifasi. 2. QMessageBox vidgetining asosiy xususiyatlari. 3. Statik funksiyalari. 4. Piktogramma va Pixmap xususiyatlari. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 18. | Amaliy | 2 | **2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.  **5-mashg‘ulot.** PyQt da rasmlar va menyular.  O‘quv savollari:   1. PyQt paketi yordamida rasm joylashtirish usullari. 2. Menu yaratish. Menu vidgetining xususiyatlari. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 19. | Amaliy | 2 | **2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.  **6-mashg‘ulot.** Matn muharriri dizaynini yaratish.  O‘quv savollari:   1. Yaratiladigan matn muharriri uchun kerakli uskunalarni tanlash. 2. Tanlangan uskunalarni matn muharriri ekraniga joylashtirish. 3. Matn muharririr ekrani dizaynini shakllantirish. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 20. | Amaliy | 2 | **2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.  **7-mashg‘ulot.** Matn muharriri dasturini yozish.  O‘quv savollari:   1. PyQt5 da matn muharriri dizaynidagi elementlarning funksiyalari uchun dastur yozish. 2. Dasturni testlash. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 21. | Amaliy | 2 | **2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.  **8-mashg‘ulot.** PyQt5 da Minesweeper o‘yini dizaynini yaratish.  O‘quv savollari:   1. PyQt5 da Minesweeper o‘yinini yaratish. 2. O‘yin uchun kerakli uskunalarni tanlash. 3. O‘lchamlarni o‘rnatish. O‘yin dizaynini yaratish. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 22. | Amaliy | 2 | **2-Mavzu:** PyQt5 paketi va QtDesigner dasturi yordamida GUI dasturlarini yaratish.  **9-mashg‘ulot.** PyQt5 da Minesweeper o‘yini dasturini yozish.  O‘quv savollari:   1. PyQt5 da Minesweeper o‘yini dizaynidagi tugmalarning funksiyalari uchun dastur yozish. 2. Dasturni testlash. 3. O‘yinni ishga tushirish. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 23. | Ma’ruza | 2 | **3-Mavzu:** Pythonda tarmoq dasturlashga kirish.  **1-mashg‘ulot.** Tarmoqda ma’lumot almashuvchi klient-server dasturini tuzish.  O‘quv savollari:   1. Socket modulining asosiy metodlari bilan tanishish. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 24. | Amaliy | 2 | **3-Mavzu:** Pythonda tarmoq dasturlashga kirish.  **2-mashg‘ulot.** Socket moduli bilan ishlash.  O‘quv savollari:   1. Socket modulining asosiy metodlari bilan tanishish. 2. .socket(), .bind, .listen, .accept(), .connect(), .send(), recv(), .close() metodlari bilan ishlash. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 25. | Amaliy | 2 | **3-Mavzu:** Pythonda tarmoq dasturlashga kirish.  **3-mashg‘ulot.** Pythonda TCP klient-server dasturini tuzish.  O‘quv savollari:   1. Socket modulining asosiy metodlari yordamida client-server dasturini tuzish. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 26. | Amaliy | 2 | **3-Mavzu:** Pythonda tarmoq dasturlashga kirish.  **4-mashg‘ulot.** TCP klient-server dasturini testlash.  O‘quv savollari:   1. TCP client-server dasturi yordamida ma’lumot almashish. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 27. | Amaliy | 2 | **3-Mavzu:** Pythonda tarmoq dasturlashga kirish.  **5-mashg‘ulot.** PyQt5 paketidan foydalanib zamonaviy chat dasturini tuzish.  O‘quv savollari:   1. TCP dasturini client qismini GUI ko‘rinishda ishlab chiqish. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 28. | Amaliy | 2 | **3-Mavzu:** Pythonda tarmoq dasturlashga kirish.  **6-mashg‘ulot.** Pythonda GUI paketidan foydalanib chat dasturini tuzishni yakunlash  O‘quv savollari:   1. TCP client-server dasturini GUI ko‘rinishda ishlab chiqish. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 29. | Amaliy | 2 | **3-Mavzu:** Pythonda tarmoq dasturlashga kirish.  **7-mashg‘ulot.** PyQt5 paketidan foydalanib zamonaviy chat dasturini tuzish.  O‘quv savollari:   1. GUI TCP klient-server dasturi yordamida ma’lumot almashinuvini testlash. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| 30. | Amaliy | 2 | **3-Mavzu:** Pythonda tarmoq dasturlashga kirish.  **8-mashg‘ulot.** **Python ilovasini kompilyatsiya qilish**.  O‘quv savollari:   1. Python ilovasini kompilyatsiya qilish. | Kompyuter. Interaktiv panel. Taqdimot materiallari. |
| **JAMI:** | | **60 soat** | | |

**IV. MUSTАQIL TА’LIM VА MUSTАQIL ISHLАR**

|  |  |  |
| --- | --- | --- |
| **T/r** | **Mustaqil tayyorgarlik mavzulari** | **Soatlar hajmi** |
| **7- semestr** | | |
| 1. | Pythonda umumiy masalalarga dior dastur tuzish (Natijani Prt.Scrn., kodni yozing, izox bering va word faylga saqlang). | 15 |
| 2. | Pythonda fayllar bilan ishlash (Natijani Prt.Scrn., kodni yozing, izox bering va word faylga saqlang). | 15 |
| 3. | PyQt5 kutubxonasi bilan ishlash (Natijani Prt.Scrn., kodni yozing, izox bering va word faylga saqlang). | 15 |
| 4. | Pythonda tarmoq dasturlari yaratish (Natijani Prt.Scrn., kodni yozing, izox bering va word faylga saqlang). | 15 |
|  | **JAMI:** | **60 soat** |

Mustaqil o‘zlashtiriladigan mavzular bo‘yicha kursantlar tomonidan (referat, prezentatsiya, esse, mustaqil (ijodiy) ish, muammoli ma’ruza va boshqalar) tayyorlanadi va uni taqdimoti tashkil qilinadi.

**V. FAN BO‘YICHA KURSANTLAR BILIMINI BAHOLASH  
VA NAZORAT QILISH ME’ZONLARI**

**Baholash tartibi**

Kursantlarning bilim saviyasi, ko‘nikma va malakalarini nazorat qilishning reyting tizimi asosida kursantning har bir fan bo‘yicha o‘zlashtirish darajasi ballar orqali ifodalanadi.

Har bir fan bo‘yicha kursantning semestr davomidagi o‘zlashtirish ko‘rsatkichi **100 ballik tizimda** butun sonlar bilan baholanadi.

Baholash usullari:

ekspress testlar;

yozma ishlar;

og‘zaki so‘rov;

me’yor va mashqlarni amaliy bajarish;

taqdimotlar.

Fanning xususiyatidan kelib chiqqan holda joriy nazorat uchun ajratilgan maksimal ball kursantning bilim va ko‘nikmalarini, mashg‘ulotlardagi faolligini, bajarilgan ishlarini kundalik mashg‘ulotlar davomida joriy baholash hamda ular tomonidan bajarilgan mustaqil ta’lim topshiriqlarini baholashga quyidagicha bo‘linadi:

Joriy, oraliq va yakuniy nazorat ballari quyidagicha taqsimlanadi:

|  |  |
| --- | --- |
| Joriy nazorat | 40 ball |
| Oraliq nazorat | 20 ball |
| Yakuniy nazorat | 40 ball |
| Fan bo‘yicha jami: | 100 ball |

**Joriy nazoratga maksimal 40 ball ajratilganda:** kundalik mashg‘ulotlar davomida joriy baholashga - 30 ball; mustaqil ta’lim topshiriqlarini baholashga - 10 ball;

Kursantlarning bilim va ko‘nikmalari, mashg‘ulotlardagi faolligi kundalik mashg‘ulotlar davomida joriy baholash 5 ballik (0-5 ball) tizimda butun sonlar bilan baholanadi.

**5 ball -** kursant mavzuga oid materiallarga doir bilimlarini chuqur namoyon etib, ularni bilimi bilan va mantiqan to‘g‘ri bayon etsa, mustaqil xulosa va to‘g‘ri qaror qabul qilsa, ijodiy fikrlab mustaqil mushohada yurita olsa, mavzuning mohiyatini chuqur tushunib bilsa va ifodalay olganda;

**4 ball -** kursant mavzuga oid materiallari puxta bilib, ularni mantiqan to‘g‘ri bayon etsa, bergan javoblarida sezilarli noaniqliklarga yo‘l qo‘ymagan bo‘lsa, mustaqil mushohada yuritsa, mavzuning mohiyatini tushunib bilsa va ifodalay olganda;

**3 ball -** kursant mavzuga oid materialning asosiy qismini bilib, tafsilotlarini o‘zlashtirib olmagan, lekin bergan javoblarida qo‘pol xatoliklarga yo‘l qo‘ymagan bo‘lsa, to‘g‘ri qaror qabul qilishi uchun ayrim hollarda unga yordamchi (esga soluvchi) savollar berilishi zarur bo‘lsa, mavzuning mohiyatini tushunsa va ifodalay olganda;

**2 ball -** kursant mavzuga materialning asosiy qismini bilmasa yoki bilib, tafsilotlarini o‘zlashtirib olmagan, bergan javoblarida qo‘pol xatoliklarga yo‘l qo‘ygan bo‘lsa, olgan bilimini amalda qo‘llay olishni mukammal bilmaganda.

**0-1 ball -** kursant mavzuga materialning asosiy qismini bilmasa yoki bilib, tafsilotlarini o‘zlashtirib olmagan, bergan javoblarida pala-partishlik bo‘lsa, qo‘pol xatoliklarga yo‘l qo‘yganda;

Semestr yakunida kursantning kundalik mashg‘ulotlar davomida joriy baholash bo‘yicha yig‘gan balini hisoblashda uning mashg‘ulotlar davomida olgan ballar yig‘indisi kursant baholangan mashg‘ulotlar soni yig‘indisiga bo‘linadi va mazkur nazorat turiga ajratilgan maksimal balldan kelib chiqqan holda belgilanadigan koeffitsiyentga ko‘paytiriladi:

KJ = \* Q

jumladan:

KJ - kursantlarning kundalik mashgʻulotlar davomida joriy baholash bo‘yicha to‘plagan bali;

J - kursantning mashgʻulotlar davomida olgan ballari yigʻindisi;

M - kursant baholangan mashgʻulotlar soni (faqat kursant baholangan mashgʻulotlar soni ko‘rsatiladi);

Q - ajratilgan maksimal baldan kelib chiqqan holda belgilanadigan koeffitsiyent (joriy nazoratning mazkur turiga ajratilgan maksimal ball 30 ball bo‘lganda koeffitsiyent - 6, maksimal ball 40 ball bo‘lganda koeffitsiyent - 8).

Kursantlar tomonidan fanning **mustaqil ta’lim** mavzulari bo‘yicha bajarilgan mustaqil ta’lim topshiriqlarini baholash 5 ballik (0-5 ball) tizimda butun sonlar bilan baholanadi.

Kursantlar tomonidan mustaqil ta’lim mavzulari bo‘yicha bajarilgan mustaqil ta’lim topshiriqlarini baholash 5 ballik tizimda butun sonlar bilan quyidagicha baholanadi:

**5 ball** – topshiriqqa doir bilimlar to‘liq bayon etilgan, amalda qo‘llay olishini to‘gʻri va ishonch bilan ifodalangan;

**4 ball** – topshiriqqa doir bilimlar bayon etilgan, amalda qo‘llay olishida ayrim noaniqliklarga yo‘l qo‘ygan holda ifodalangan;

**3 ball** – topshiriqqa doir bilimlar bayon etilgan, amalda qo‘llay olishida sezilarli noaniqliklarga yo‘l qo‘ygan holda ifodalangan;

**2 ball** – topshiriqqa doir bilimlar juda kam darajada bayon qilingan, amalda qo‘llay olishida xatolarga yo‘l qo‘ygan holda ifodalangan;

**1 ball** – topshiriqqa doir bilimlar xatoliklar bilan bayon qilingan, amalda qo‘llay olishini ifodalay olmagan.

**0 ball** – topshiriqqa doir bilimlar bayon qilinmagan, topshiriq bajarilmagan (0-ball jurnalga qo‘yilmaydi, lekin kursantga yetkaziladi).

Kursantlar har bir mustaqil ta’lim mavzusi bo‘yicha navbatdagi mustaqil ta’lim mavzusi topshiriqlari berilgunga qadar semestrga rejalashtirilgan so‘nggi mustaqil ta’lim mavzusi bo‘yicha esa attestatsiya sessiyasi boshlangunga qadar baholanishlari lozim.

Semestr yakunida kursantning mustaqil ta’lim mavzulari bo‘yicha yigʻgan balini hisoblashda, uning mustaqil ta’lim topshiriqlari bo‘yicha olgan ballari yigʻindisi ishchi о‘quv dasturiga ko‘ra semestrga rejalashtirilgan mustaqil ta’lim mavzulari soniga bo‘linadi va mazkur nazorat turiga ajratilgan maksimal baldan kelib chiqqan holda belgilanadigan koeffitsiyentga ko‘paytiriladi:

bu yerda:

MJ - kursantning mustaqil ta’lim mavzulari bo‘yicha to‘plagan balli;

MI - kursantning mustaqil ta’lim topshiriqlari bo‘yicha olgan ballar yigʻindisi;

MT - mustaqil ta’lim mavzulari soni (ishchi o‘quv dasturiga ko‘ra semestrga rejalashtirilgan barcha mustaqil ta’lim mavzulari soni ko‘rsatiladi);

Q - ajratilgan maksimal balldan kelib chiqqan holda belgilanadigan koeffitsiyent (joriy nazoratning mazkur turiga ajratilgan maksimal ball 10 ball bo‘lganda koeffitsiyent - 2, maksimal ball 20 ball bo‘lganda koeffitsiyent - 4).

Semestr yakunida kursantning joriy baholash bo‘yicha to‘plagan umumiy bali, uning kundalik mashgʻulotlar davomida joriy baholash bo‘yicha va mustaqil ta’lim mavzulari bo‘yicha to‘plagan ballari yigʻindisi bo‘yicha chiqariladi:

*JB =KJ + MJ*

bu yerda:

JB - semestr yakunida kursantning joriy baholash bo‘yicha to‘plagan umumiy balli;

KJ - kursantning kundalik mashgʻulotlar davomida joriy baholash bo‘yicha to‘plagan balli;

MJ - kursantning mustaqil ta’lim mavzulari bo‘yicha to‘plagan balli.

Kursantlarning bilim va amaliy ko‘nikmalari darajasining oraliq nazoratini o‘tkazishda **oraliq nazorat** test shaklida qabul qilinadigan oraliq nazoratlardagi test savollariga berilgan har bir to‘g‘ri javob uchun - 0,5 ball beriladi.

Test shaklida qabul qilinadigan oraliq nazoratlarda kursant tomonidan to‘plangan butun bo‘lmagan ballar yuqoriga yaxlitlanadi.

Semestr yakunida kursantning mustaqil ta’lim topshiriqlari bo‘yicha to‘plagan umumiy ball, har bir mustaqil ta’lim topshirig‘i natijasilarining yig‘indisi bo‘yicha hisoblanadi.

*MJ=M1+M2 ...+Mn,*

bu yerda:

*MJ* - kursant mustaqil ta’lim topshiriqlari bo‘yicha umumiy ball;

*M1, M2, Mn,-* kursant *1-* chi*, 2-* chi, *n-*chi mustaqil ta’lim topshiriqlari bo‘yicha alohida olgan ballari;

Semestr yakunida kursantning joriy baholash bo‘yicha to‘plagan umumiy balli, uning kundalik mashg‘ulotlar davomida joriy baholash bo‘yicha va mustaqil ta’lim mavzulari bo‘yicha to‘plagan ballari yig‘indisi bo‘yicha chiqariladi:

*JB =KJ + MJ*

bu yerda:

JB - semestr yakunida kursantning joriy baholash bo‘yicha to‘plagan umumiy balli;

KJ - kursantning kundalik mashg‘ulotlar davomida joriy baholash bo‘yicha to‘plagan balli.

**Yakuniy nazorat** topshirishda har bir savolga berilgan javobni baholash mezonlari:

**9-10 ball** - kursant dasturiy materiallarga doir bilimlarini chuqur namoyon etib, ularni bilimdonlik bilan va mantiqan to‘g‘ri bayon etsa, ijodiy fikrlab mustaqil mushohada yurita olsa, savol mohiyatini chuqur tushunib bilsa va unga javobni to‘g‘ri ifodalay olsa, hamda yetarli darajada tasavvurga ega bo‘lsa;

**7-8 ball** – agar kursant dasturiy materiallarni puxta bilib, ularni mantiqan to‘g‘ri bayon etsa, bergan javoblarida sezilarli noaniqliklarga yo‘l qo‘ymagan bo‘lsa, mustaqil mushohada yuritsa, olgan bilimini amalda qo‘llay olishni namoyon qilsa, fanning mohiyatini tushunib bilsa va ifodalay olsa, hamda tasavvurga ega bo‘lsa;

**5-6 ball** – kursant dasturiy materialning asosiy qismini bilib, tafsilotlarini o‘zlashtirib olmagan, lekin bergan javoblarida qo‘pol xatoliklarga yo‘l qo‘ymagan bo‘lsa, to‘g‘ri qaror qabul qilishi uchun ayrim hollarda unga yordamchi (esga soluvchi) savollar berilishi zarur bo‘lsa, olgan bilimini amalda qo‘llay olishni bilsa, fanning mohiyatini tushunsa va ifodalay olsa hamda fan bo‘yicha tasavvurga ega bo‘lsa;

**0-4 ball** – kursant dasturiy materialning asosiy qismini bilmasa yoki bilib, tafsilotlarini o‘zlashtirib olmagan, bergan javoblarida qo‘pol xatoliklarga yo‘l qo‘ygan bo‘lsa, olgan bilimini amalda qo‘llay olishni mukammal bilmasa.

Yakuniy nazoratlarda kursantlarning bilimiga belgilanadigan umumiy ball har bir savolga berilgan javoblar uchun qo‘yilgan alohida ballar yig‘indisiga asosan chiqariladi.

Kursantning semestr davomida fan bo‘yicha to‘plagan umumiy bali har bir nazorat turidan belgilangan qoidalarga muvofiq to‘plagan ballari yig‘indisiga teng.

Kursantlar tegishli fan bo‘yicha yakuniy nazorat o‘tkaziladigan muddatga qadar joriy va oraliq nazoratlari topshirgan bo‘lishlari shart.

Kursantlar fan bo‘yicha yakuniy nazorat o‘tkaziladigan muddatga qadar joriy nazoratlarni topshirgan bo‘lishlari shart.

Joriy nazorat turlari bo‘yicha 55 va undan yuqori ballni to‘plagan kursant fanni o‘zlashtirgan deb hisoblanadi va ushbu fan bo‘yicha yakuniy nazoratga kirmasligiga yo‘l qo‘yiladi.

Fan bo‘yicha joriy va oraliq nazoratlarga ajratilgan umumiy balning **55 foizi  
(33 ball)** saralash ball hisoblanib, ushbu foizdan kam ball to‘plagan kursantlar yakuniy nazoratga **kiritilmaydi**.

Fan bo‘yicha o‘tkazilgan joriy va yakuniy nazorat turlari bo‘yicha to‘plangan ballar yig‘indisi 55 balldan kam bo‘lsa, kursant akademik qarzdor hisoblanadi.

Fan bo‘yicha kursantning semestr davomidagi o‘zlashtirish ko‘rsatkichi 100 ballik tizimda butun sonlar bilan baholanadi.

**86-100 ball (a’lo)**, agar kursant dasturiy materiallarga doir bilimlarini chuqur namoyon etib, ularni bilimdonlik bilan va mantiqan to‘g‘ri bayon etsa, mustaqil xulosa va to‘g‘ri qaror qabul qilsa, ijodiy fikrlab mustaqil mushohada yurita olsa, olgan bilimini amalda qo‘llay olishni namoyon qilsa, fanning mohiyatini chuqur tushunib bilsa va ifodalay olsa hamda fan bo‘yicha yetarli darajada tasavurga ega deb topilganda;

**71-85 ball (yaxshi)**, agar kursant dasturiy materiallarni puxta bilib, ularni mantiqan to‘g‘ri bayon etsa, bergan javoblarida sezilarli noaniqlikarga yo‘l qo‘yilmagan bo‘lsa, mustaqil mushohada yuritsa, olgan bilimini amalda qo‘llay olishni namoyon qilsa, fanning mohiyatini tushunib bilsa va ifodalay olsa hamda fan bo‘yicha tasavvurga ega deb topilganda;

**55-70 ball (qoniqarli)**, agar kursant dasturiy materialning asosiy qismini bilib, tafsilotlarini o‘zlashtirib olmagan, lekin bergan javoblarida qo‘pol xatoliklarga yo‘l qo‘yilmagan bo‘lsa, to‘g‘ri qaror qabul qilish uchun ayrim hollarda unga yordamchi (esga soluvchi) savollar berilishi zarur bo‘lsa, olgan bilimini amalda qo‘llay olishni bilsa, fanning mohiyatini tushunsa va ifodalay olsa hamda fan bo‘yicha tasavvurga ega deb topilsa;

**0-54 ball (qoniqarsiz),** agar kursant dasturiy materialning asosiy qismini bilmasa, yoki bilib, tafsilotlarini o‘zlashtirib olmagan, bergan javoblarida qo‘pol xatoliklarga yo‘l qo‘ygan bo‘lsa, olgan bilimini amalda qo‘llay olishni mukammal bilmasa, fanning mohiyatini tushunmasa, yoqi tushunsada, ammo ifodalay olmasa hamda fan bo‘yicha tasavvurga ega emas deb topilganda.

**VI. АSOSIY VА QO‘SHIMCHА O‘QUV АDАBIYOTLАR HАMDА**

**АXBOROT MАNBАLАRI**

**Asosiy adabiyotlar:**

1. Sh.R. Sapayev, B.K. Yusupov, A.A.Abidov. “Python dasturlash tili” Darslik. Toshkent: 2024y. В - 316.
2. Sh.R. Sapayev “Python dasturlash tili asoslari”. O‘quv qo‘llanma. Toshkent: 2023y. В – 137.
3. Sh.R. Sapayev “PyQt5 paketi va QtDesigner dasturida grafik ilovalar tuzish”. O‘quv qo‘llanma. Toshkent: 2024y. В - 150

**Qo‘shimcha adabiyotlar:**

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих.-СПб.: Питер, 2017.-288 с. : ил. ISBN 978-5-496-02541-6
2. Н.А.Прохоренок, В.А.Дронов. “Python3 и PyQT5. Разработка приложений”. СПб.: БХВ-Петербург, 2016. – 832 с.: ил.
3. Франсуа Шолле. “Глубокое обучение на Python”. — СПб.: Питер, 2018. — 400 с.: ил. — (Серия «Биб- лиотека программиста»).
4. Чан, Уэсли. “Python: создание приложений. Библиотека профессионала”, 3-е изд. [Пер. с англ. - М. : ООО "И.Д. Вильяме"], Москва: Санкт-Петербург • Киев 2015.
5. Марк Саммерфилд. “Программирование на Python 3. Подробное руководство” [Пер. с англ. – СПб]. - Москва: Санкт-Петербург–2009 год.

**Internet saytlari:**

1. https://www.python.org
2. https://python-scripts.com
3. https://webformyself.com/python

**O‘ZBEKISTON RESPUBLIKASI HARBIY XAVFSIZLIK VA MUDOFAA UNIVERSITETINING AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI VA HARBIY ALOQA INSTITUTI**

**“AXBOROT TEXNOLOGIYALARI VA SUN’IY INTELLEKT”**

**kafedrasi**

****

**« PYTHON DASTURLASH TILI» FANIDAN**

Fan bo'yicha asosiy va

qo'shimcha adabiyotlar hamda

axborot manbaalari

(5 ilova)

**Toshkent – 2025ASOSIY VA QO‘SHIMCHA O‘QUV ADABIYOTLAR HAMDA AXBOROT MANBALARI**

**Asosiy adabiyotlar:**

1. Sh.R. Sapayev, B.K. Yusupov, A.A.Abidov. “Python dasturlash tili” Darslik. Toshkent: 2024y. В - 316.
2. Sh.R. Sapayev “Python dasturlash tili asoslari”. O‘quv qo‘llanma. Toshkent: 2023y. В – 137.
3. Sh.R. Sapayev “PyQt5 paketi va QtDesigner dasturida grafik ilovalar tuzish”. O‘quv qo‘llanma. Toshkent: 2024y. В - 150

**Qo‘shimcha adabiyotlar:**

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих.-СПб.: Питер, 2017.-288 с. : ил. ISBN 978-5-496-02541-6
2. Н.А.Прохоренок, В.А.Дронов. “Python3 и PyQT5. Разработка приложений”. СПб.: БХВ-Петербург, 2016. – 832 с.: ил.
3. Франсуа Шолле. “Глубокое обучение на Python”. — СПб.: Питер, 2018. — 400 с.: ил. — (Серия «Биб- лиотека программиста»).
4. Чан, Уэсли. “Python: создание приложений. Библиотека профессионала”, 3-е изд. [Пер. с англ. - М. : ООО "И.Д. Вильяме"], Москва: Санкт-Петербург • Киев 2015.
5. Марк Саммерфилд. “Программирование на Python 3. Подробное руководство” [Пер. с англ. – СПб]. - Москва: Санкт-Петербург–2009 год.

**Internet saytlari:**

1. https://www.python.org
2. https://python-scripts.com
3. https://webformyself.com/python