“ Dunyoda ilmdan boshqa
najot yo`q va bo`lmagay. ”
Imom Al-Buxoriy

O’zbekistonda o’qitish texnologiyalarini zamonaviylashtirish, jadallashtirish rivojlangan iqtisodiyotli mamlakatlarga qaraganda yanada dolzarb ahamiyatga ega. Chunki hozirgi kunda milliy ta’lim tizimining salohiyatli tizimli rivojlanishi yanada yuqori pog’onaga ko’tarildi.

Bitiruv malakaviy ish mavzusining dolzarbligi: mavzuning dolzarbligi shundan iboratki, Python bu umumiy maqsadli dasturlash uchun keng tarzda foydalaniladigan yuqori darajali dasturlash tili, chunki o'rganish oson va qulay sintaksisga ega. Undan tashqari skriptli dasturlash tillariga kiradi. Python dinamik tipizatsiyaga ega, obyektga yo'naltirilgan dasturlash, funksional dasturlash, strukturali, avtomatik xotirani boshqarish va albatta ko'p patokli dasturlash tillaridan biri. Python har xil platformalar uchun yozilgan masalan Windows, Linux, Mac OSX, Palm OS, Mac OS va hokazo. Python Microsoft.NET platformasi uchun yozilgan realizatsiyasi ham bor uni nomi IronPython. Bugungi kunda dunyoga mashhur ko`plab kompaniyalar NASA, Google, Yandex, CERN, Apple computer, Dream Works, kosmik teleskop institutlari Pythonni ishlatishadi. Dunyoning rivojlangan mamlakatlari AQSH (Koliforniya Universiteti, Florida Universiteti, Lova Universiteti, Massachustva Texnologiya Universiteti), Kanada (Toronto Universiteti, Alberto Universiteti), Buyuk Britaniya (Oksford Universiteti), Fransiya, Rossiya, Avstraliya, Ispaniyaning universitet va kollejlarida o`qitishda Python dasturlash tili qo`llaniladi. Pythonning o`ziga xos jihatlaridan biri bu dasturni yozish davomida quyi darajadagi detallarni, misol uchun xotirani boshqarishni hisobga olishga hojat qolmaydi. Shuningdek dasturni yozish davomida

ortiqcha kod yozishdan xalos bo`linadi. Masalan: massiv elementlarini tartiblash misolini Paskal va Python dasturidagi talqinini taqqoslab solishtirib ko`raylik.

```
For i:=0 to N-1 do
For j:=N-2 downto i do
If A[j] > A[j+1] then begin
c:=A[j]; A[j]:=A[j+1];
A[j+1]:=c
end;
```

```
A.sort()
```

Shu bilan birga Pythonda dastur yozayotganda *begin-end, {}* yoki satr tugagani bildirish uchun *nuqtali vergul (;)* qo`yilmaydi.

Tadqiqot ob'yekti va predmeti: Python dasturlash tili. Dasturiy ta’minotlar bilan jihozlangan kompyuter xonalari. Python dasturlash tilida dastur tuzish uchun WingWare, Paycharm, Paydev, Notepad++ va boshqa muhitlari. Dasturlarni tuzishda va muhitlar bilan ishlashda kompyuter texnologiyalari, shu bilan birgalikda eng zamonaviy usullardan foydalanish. Biz tadqiqot obyekti sifatida Wingware muhitida ishlab dastur natijalarini olishga harakat qildik.

Ishning maqsadi va vazifalari: Python dastulash tili haqida uslubiy qo`llanma yaratish. Python dasturini o`rnatish, uning sintaksisini, operator va funksiyalarini o`rganish, arifmetik amallarni bajarishni, sonlar, satrlar bilan ishlashni hamda ulardan foydalanib masalalar yechishni o`rganish. Masalalarr yechimini to`griligini kompyuterga WingWare muhitiga kiritib tekshirib ko`rish. Python tilining imkoniyatlarini ko`rish, uni bugungi kundagi qanday loyihalarni tuzishda ishlatilishini o`rganish. Talabalar mustaqil shug’ullanishlari uchun uslubiy qo’llanma ko’rinishida tayyorlash. Talabalarga kompyuterda mustaqil masala yechish usullarini ko’rsatish va talabalarni dasturlashga bo’lgan qiziqishlarini oshirish.

Tadqiqot usuli va uslubiyoti: Wingwareda operatorlarni ishlatilishini, sintaksisini o`rganib dasturlash tilida kiritib natijalarni olib xulosalab har bir operator va funksiyalar haqida to`liq ma’lumot berishga harakat qilish.

Olingan asosiy natijalar: Qo'llanmamizda Pyhton dasturlash tili uning imkoniyatlari sintaksisi kutubxonalari va Python dasturi haqida boshlang'ich va mukammal tushunchalar amaliy misollar bilan birgalikda yoritildi. Boshqa dasturlash tillariga nisbatan Python dasturini operator, funksiya, standart kutubxonalarini o`rganib dastur tuzishda qo`llash va undan olingan natijalarning afzallik tomonlari ko'rsatib berildi. Shular asosida Python dasturlash tili va uning imkoniyatlari bo`yicha o`zbek tilida uslubiy qo`llanma yaratildi.

Natijalarning ilmiy yangiligi va amaliy ahamiyati: Mazkur Python dasturlash tilini o`rganish talabalarga dasturlarni tuzish va masalalarni yanada mukammal ishlay olish, samaradorligi va ahamiyatlilik darajasini oshirish kabi imkoniyatlar yaratib bera oladi. Python dasturlash tilidan zamonaviy dasturlash tillari fanini o`qitishda foydalanish ilmiy yangilikdir va bu borada uslubiy qo`llanma juda katta ahamiyat kasb etadi.

Tadbiq etish darajasi va qo'llanish sohasi. Ushbu qo`llanma Python dasturlash tili haqidagi boshlangich tushunchalarni akdemik litsey, kasb-hunar kolleji va oliy ta'limning talabalariga o`rgatishga mo`ljallangan. Bundan tashqari olimpiada masalalarini yechishda va yaxshi natijalarga erishishda ushbu dasturlash tilidan foydalanishni amalga oshirishda mazkur qo`llanmadan foydalanish maqsadga muofiqdir.

Ishning hajmi va tuzilishi: Kirish, 2 ta bob, har bir bobning qisqacha xulosasi, foydalanilgan adabiyotlar ro'yxati va xotimalardan iborat holda bayon qilingan. I bobda Python dasturlash tili va uning sintaksisi haqida malumot keltirilgan bo`lib 40 betdan iborat, II bobda Pythonda ma'lumotlar tuzilmasi va tilning standart modullari keltirilgan. Bu bob 21 betdan iborat. Jami 11 ta chizma dan tashkil topgan.

I-BOB. PYTHON DASTURLASH TILI VA UNING SINTAKSISI

1.1 Python dasturlash tili yaratilishi tarixi, imkoniyatlari va uni o`rnatish

Python dasturlash tilini yaratilishi 1980-yil oxiri 1990-yil boshlaridan boshlangan. O`sha paytlarda uncha taniqli bo`lmagan Gollandiyaning CWI instituti xodimi Gvido van Rossum ABC tilini yaratilish proektida ishtirok etgan edi. ABC tili Basic tili o`rniga talabalarga asosiy dasturlash konsepsiyalarini o`rgatish uchun mo`ljallangan til edi. Bir kun Gvido bu ishlardan charchadi va 2 hafta davomida o`zining Macintoshida boshqa oddiy tilning interpretatorini yozdi, bunda u albatta ABC tilining ba'zi bir g`oyalarini o`zlashtirdi. Shuningdek, Python 1980-1990-yillarda keng foydalanilgan Algol-68, C, C++, Modul3 ABC, SmallTalk tillarining ko`plab xususiyatlarini o`ziga olgandi. Gvido van Rossum bu tilni internet orqali tarqata boshladi. Bu paytda o`zining "Dasturlash tillarining qiyosiy taqrizi" veb sahifasi bilan internetda to 1996-yilgacha Stiv Mayevskiy ismli kishi taniqli edi. U ham Macintoshni yoqtirardi va bu narsa uni Gvido bilan yaqinlashtirdi. O`sha paytlarda Gvido BBC ning "Monti Paytonning havo sirki" komediyasining muxlisi edi va o`zi yaratgan tilni Monti Payton nomiga Python deb atadi (ilon nomiga emas).

Til tezda ommalashdi. Bu dasturlash tiliga qiziqqan va tushunadigan foydalanuvchilar soni ko`paydi. Boshida bu juda oddiy til edi. Shunchaki kichik interpretator bir nechta funksiyalarga ega edi. 1991-yil birinchi OYD(Obyektga Yo`naltirilgan Dasturlash) vositalari paydo bo`ldi.

Bir qancha vaqt o`tib Gvido Gollandiyadan Amerikaga ko`chib o`tdi. Uni CNRI korparatsiyasiga ishlashga taklif etishdi. U o`sha yerda ishladi va korparatsiya shug`ullanayotgan proektlarni Python tilida yozdi va bo`sh ish vaqtlarida tilni interpretatorini rivojlantirib bordi. Bu 1990-yil Python 1.5.2 versiyasi paydo bo`lguncha davom etdi. Gvidoning asosiy vaqti korparatsiyani proektlarini yaratishga ketardi bu esa unga yoqmasdi. Chunki uning Python dasturlash tilini rivojlantirishga vaqti qolmayotgandi. Shunda u o`ziga tilni rivojlantirishga imkoniyat yaratib bera oladigan homiy izladi va uni o`sha paytlarda endi tashkil etilgan BeOpen firmasi qo`llab quvvatladi. U CNRI dan ketdi, lekin shartnomaga binoan u Python 1.6 versiyasini chiqarib berishga majbur edi. BeOpen da esa u Python 2.0 versiyani chiqardi. 2.0 versiyasi bu oldinga qo`yilgan katta qadamlardan edi. Bu versiyada eng asosiysi til va interpretatorni rivojlanish jarayoni ochiq ravishda bo`ldi.

Shunday qilib 1.0 versiyasi 1994-yil chiqarilgan bo`lsa, 2.0 versiyasi 2000-yil, 3.0 versiyasi esa 2008-yil ishlab chiqarildi. Hozirgi vaqtda uchinchi versiyasi keng qo`llaniladi.

Python dasturlash tili imkoniyatlari

Python – bu o'rganishga oson va shu bilan birga imkoniyatlari yuqori bo'lgan oz sonlik zamonaviy dasturlash tillari qatoriga kiradi. Python yuqori darajadagi ma'lumotlar strukturasi va oddiy lekin samarador obyektga yo'naltirilgan dasturlash uslublarini taqdim etadi.

Pythonning o'ziga xosligi

- Oddiy, o'rganishga oson, sodda sintaksisga ega, dasturlashni boshlash uchun qulay, erkin va ochiq kodlik dasturiy ta'minot.
- Dasturni yozish davomida quyi darajadagi detallarni, misol uchun xotirani boshqarishni hisobga olish shart emas.
- Ko'plab platformalarda hech qanday o'zgartirishlarsiz ishlay oladi.
- Interpretatsiya(Интерпретируемый) qilinadigan til.
- Kengayishga (Расширяемый) moyil til. Agar dasturni biror joyini tezroq ishlashini xoxlasak shu qismni C yoki C++ dasturlash tillarida yozib keyin shu qismni python kodi orqali ishga tushirsa(chaqirsa) bo'ladi.
- Juda ham ko'p xilma-xil kutubxonalarga ega.
- xml/html fayllar bilan ishlash
- http so`rovlari bilan ishlash
- GUI(grafik interfeys)
- Web ssenariy tuzish
- FTP bilan ishlash
- Rasmli audio video fayllar bilan ishlash
- Robot texnikada
- Matematik va ilmiy hisoblashlarni programmalash

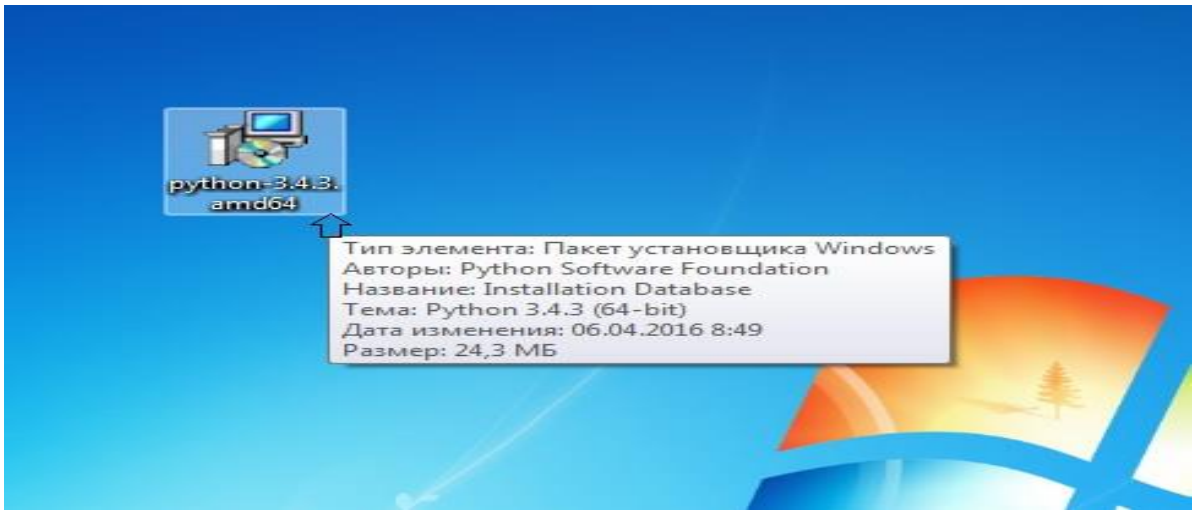
Pythonni katta proyektlarda ishlatish mumkin. Chunki, uni chegarasi yo`q, imkoniyati yuqori. Shuningdek, u sodda va universalligi bilan programmalash tillari orasida eng yaxshisidir.

Python dasturlash tilini o`rnatish.

Agar siz biror GNU/Linux distributivini ishlatayotgan bo'lsangiz ko'p xollarda sizning tizimingizda **python** o'rnatilgan bo'ladi. Buni tekshirib ko'rish uchun terminalingizdan quyidagi buyruqni ishga tushirib ko'ring. **python -V**

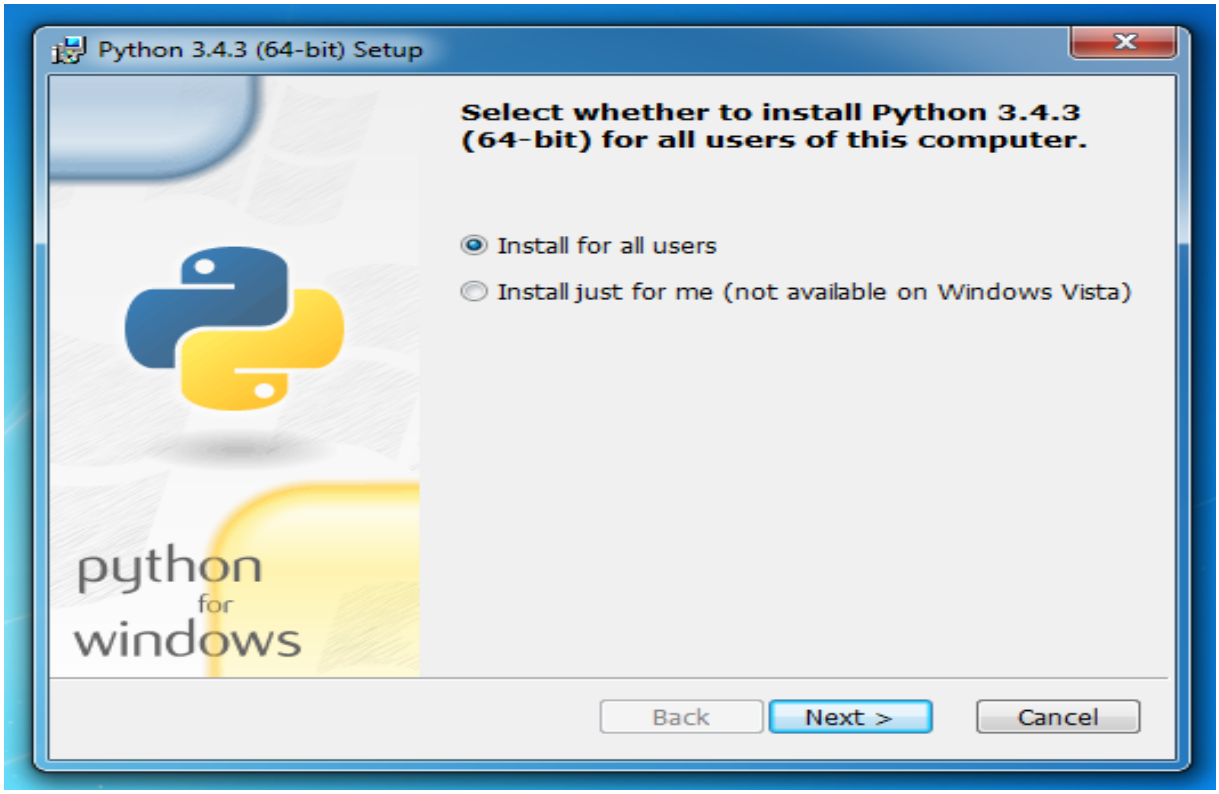
Agar sizda **Python 3.4.3** yozuvi yoki shunga o'xshash yozuv hosil bo'lsa unda hammasi joyida.

Windows operatsion tizimiga o'rnatish uchun www.python.org/downloads web sahifasiga o'tamiz va u yerdan oxirgi python versiyasini yuklab olamiz. Pythonni o'rnatish odatiy dasturlarni o'rnatish kabi kechadi. Hech qanday qiyin joyi yo'q.



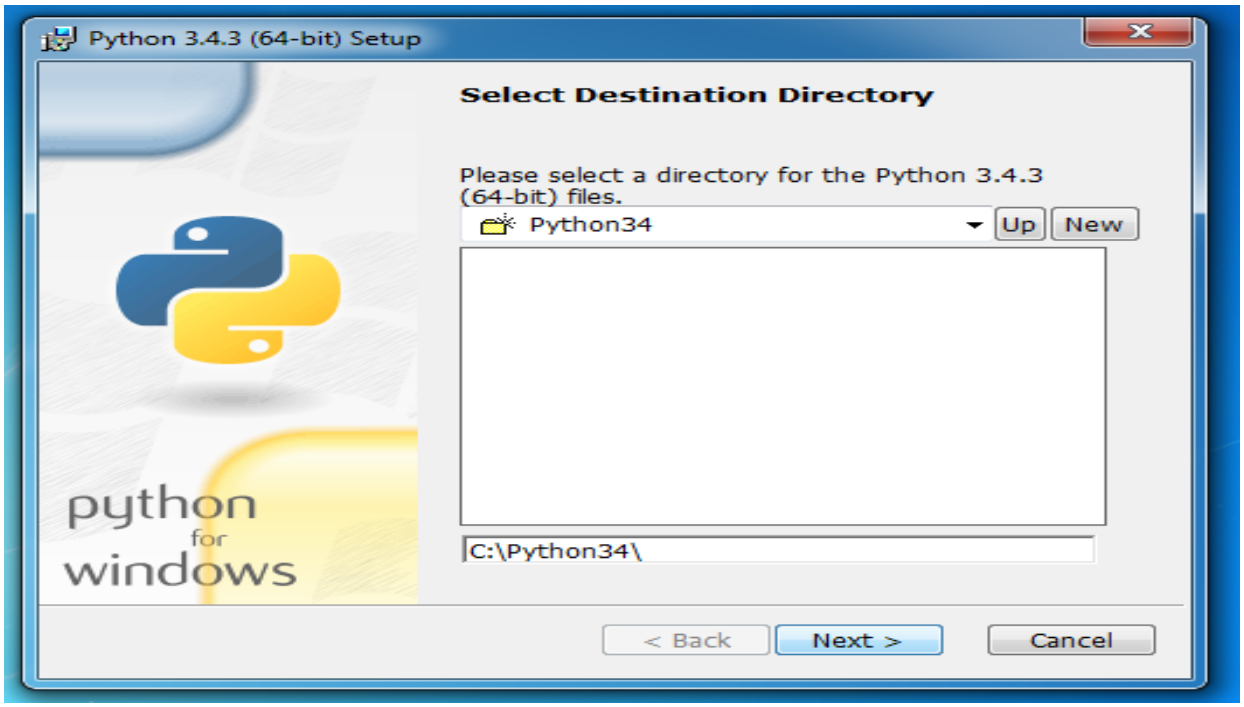
1.1.1-chizma. Python dasturining o`rnatiluvchi fayli.

Python dasturlash tilining o`rnatuvchi paketini ustiga sichqoncha ko`ratkichini 2 marta bosamiz va bizga quyidagi oyna hosil bo`ladi.



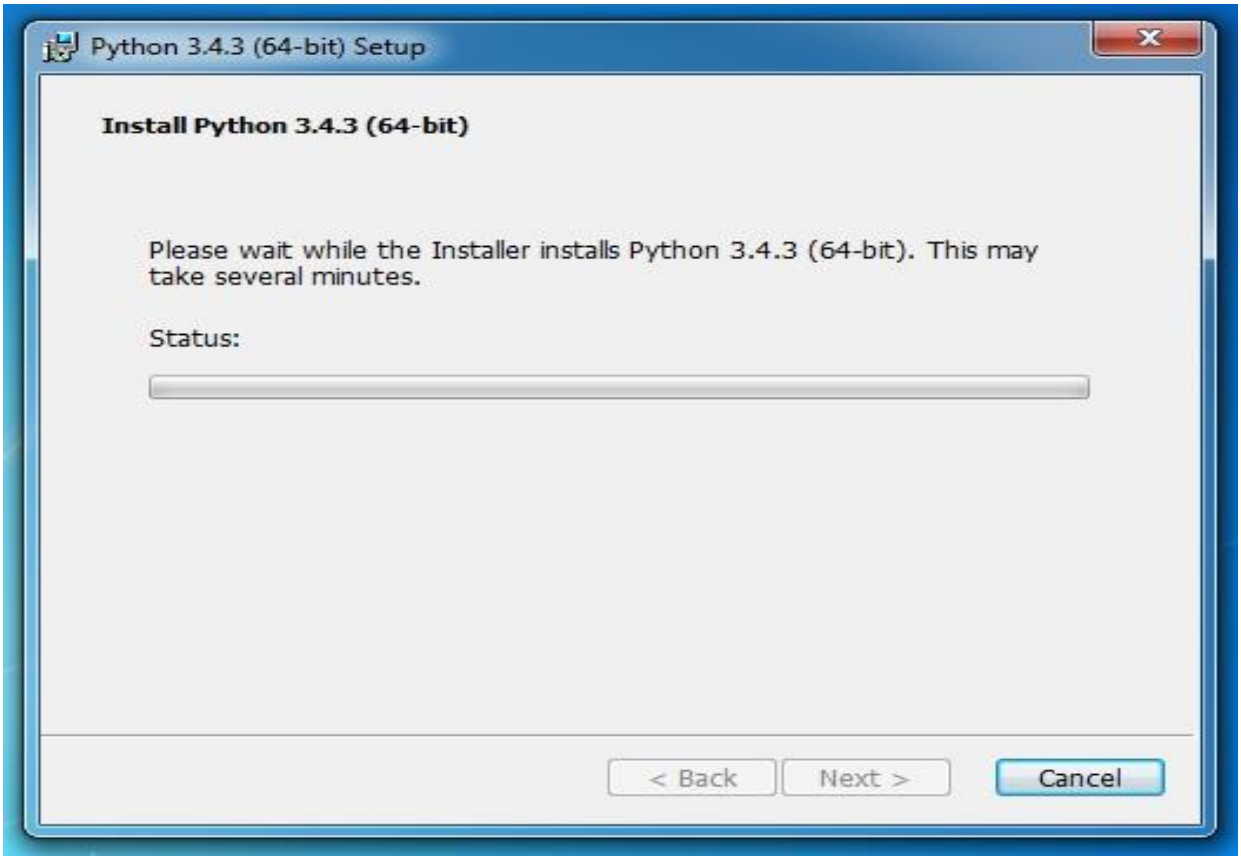
1.1.2-chizma. Python dasturini o`rnatishni boshlashni ko`rsatuvchi oyna.

Bu yerda **Install for all users**-barcha foydalanuvchilar uchun. **Install just for me**-faqat siz uchun, agar buni tanlab istalyatsiya qilsak ya'ni o`rnatsak Windows Vista operatsion sistemasida xatolik yuz beradi va dastur ishlamaydi. Shuning uchun **Install for all users** ni tanlaganimiz maqul. Keyin next tugmasi bosamiz.



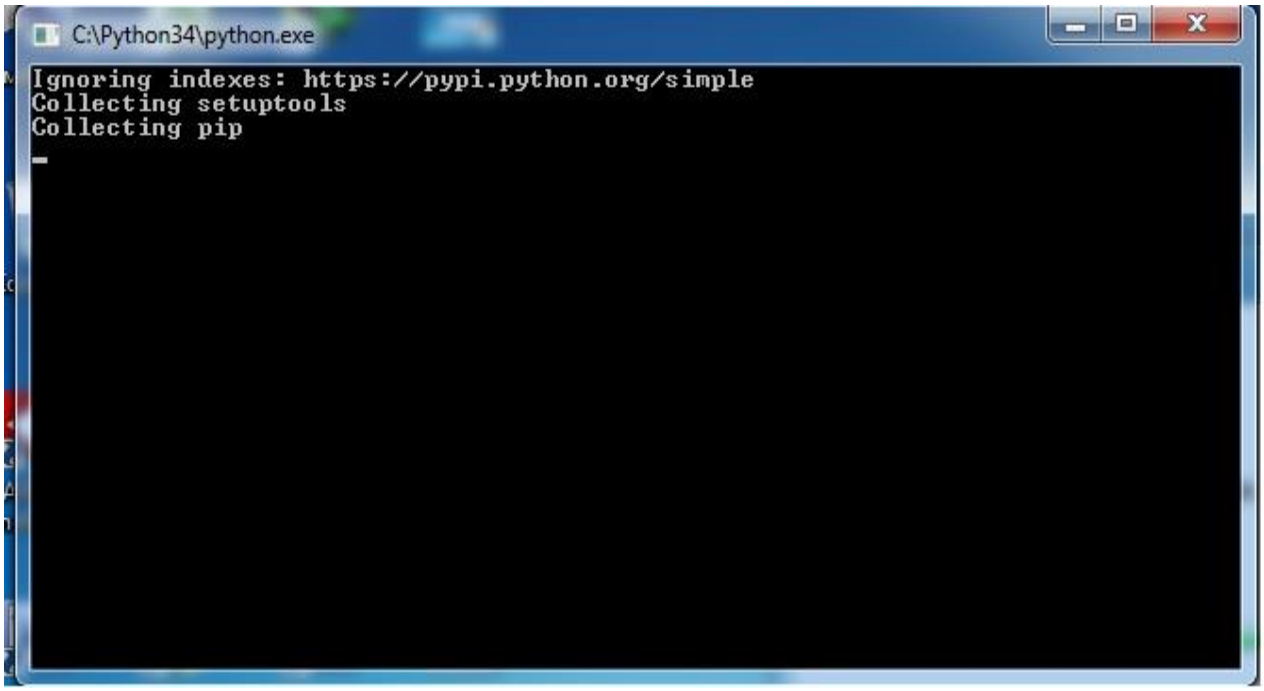
1.1.3-chizma. Python dasturini o`rnatilish joyini ko`rsatish oynasi.

Bu yerda esa Python dasturlash tilini qayerda o`rnatilishi ko`rsatilayapti.



1.1.4-chizma. Python dasturini o`rnatilish jarayoni.

Python o`rnatilyapti va bir necha sekunddan so`ng quyidagi oyna namoyon bo`ladi:



1.1.5-chizma. PIP kutubxonasini qo`shish jarayonida hosil boladigan oyna.

Bunda Console rejimida dastur ishga tushib **pip** kutubxonasini qo`shadi.

1.1.6-chizma. Python dasturini o`rnatish tugallanganligi haqidagi muloqot oynasi.

Va dasturni o`rnatish muofaqqiyatli tugallandi.

1.2.Python tili sintaksisi, asosiy operatorlari.

Python tili sintaksisi o`zi kabi sodda

- Satr oxiri instruksiyaning oxiri hisoblanadi (nuqta vergul shart emas)
- Har bir qator boshidagi bo`sh joy(отступ) muhim ahamiyatga ega. Kiritilgan amallar bo`sh joylarning kattaligiga qarab **blok**larga birlashadi. Bo`sh joy istalgancha bo`lishi mumkin asosiysi bitta kiritilgan blok chegarasida bo`sh joy bir xil bo`lishi kerak. Noto`g`ri qo`yilgan bo`sh joylar xatolik yuz berishiga olib kelishi mumkin. Bitta probel bilan bo`sh joy hosil qilish yaxshi qaror emas uni o`rniga to`rtta probel yoki *Tab* belgisini ishlatish kerak.
- Pythonga kiritilgan amallar bir xil shablonda yoziladi. Bunda asosiy amal ikki nuqta bilan tugatiladi va uning orqasidan kiritilgan blok kodi ham joylashadi. Odatda, asosiy amalning ostidagi satr bo`sh joy bilan ajratiladi.

Bir nechta maxsus holatlar

- Bazan bir nechta amalni bitta satrga nuqtali vergul bilan ajratgan holda yozish mumkin.

```
a = 1; b = 2; print(a, b)
```

Buni ko`p ham qo`llamang! Yaxshisi bunday qilmang, o`qishga noqulay.

- Bitta amalni bir nechta satrga yozish mumkin faqat aylana, to`rtburchak va figurali qavslardan foydanish kerak.

```
if (a == 1 and b == 2 and
c == 3 and d == 4):
print('spam'*3)
```

Kalit so`zlar

False – yolg`on.

True - rost.

None - "bo`sh" obyekt.

and – mantiqiy VA amali.

with / as – konteks menejeri.

break –tsikldan chiqish.

class – metod va atributlarda iborat.

continue – tsikldan keyingi iteratsiyaga o`tish.

def – funksiyani aniqlash.

del – obyektni yo`qotish.

elif – aks holda, agar.

else – for/else yoki if/elsega qarang.

for – for tsikli.

from – moduldan bir nechta funksiyani import qilish.

if - agar.

import – moduldan import.

is –xotirani bitta joyida 2 ta obyektni jo`natsa bo`ladimi.

lambda –yashirin funksiyani aniqlash.

not –mantiqiy inkor amali.

or –mantiqiy Yoki amali.

while – while tsikli.

Komentariy

Kommentariy. *Kommentariy* # simvolidan keyin yoziladi va dastur kodini o'qiyotgan dasturchi uchun eslatma bo'lib xizmat qiladi. Misol uchun:

print('salom dunyo!') # print — bu funksiya

yoki:

 # print — bu funksiya

print('salom dunyo! ')

Komentariy dastur kodini o'qiyotganlar uchun foydali bo'ladi va dastur nima qilishini oson tushunishga yordam beradi. Unga yechimdagi muhim joylarni, muhim bo'lgan qismlarni yozish mumkin.

O`zgaruvchilar

Biror ma'lumotni saqlash va uning ustida turli amallarni bajarish uchun bizga o'zgaruvchilar yordam beradi. O'zgaruvchining qiymati, o'z nomi bilan aytib turibdiki, o'zgarishi mumkin. Unda xohlagan qiymatni saqlash mumkin. O'zgaruvchilar kompyuter xotirasidagi joy bo'lib, u yerda siz biror ma'lumotni saqlaysiz. O'zgaruvchining konstantadan farqi, o'zgaruvchiga dastur ishlashi davomida (run time) murojaat qilib, uning qiymatini o'zgartira olamiz. Konstantaga esa oldindan ma'lum bir qiymat beriladi va bu qiymatni o'zgartirib bo'lmaydi.

O'zgaruvchilarni nomlashda quyidagi qoidalarga amal qilish kerak:

- O'zgaruvchining birinchi belgisi alifbo harfi (ASCII simvollari katta va kichik registrda) yoki “_” (ostki chiziq) simvoli bo'lishi mumkin.
- O'zgaruvchilarning qolgan qismi harflardan (ASCII simvollari katta va kichik registrda), “_" (ostki chiziq) simvoli va raqamlardan(0-9) tashkil topishi mumkin.
- O'zgaruvchilar nomlashda katta va kichik registrlar farqlanadi. Masalan, *myname va myName – bular boshqa-boshqa o'zgaruvchi hisoblanadi.*
- O'zgaruvchilarni to'g'ri nomlashga misollar*: i, _my_name, name_23, a1b2_c3*
- O'zgaruvchilarni noto'g'ri nomlashga misollar: 2things, ' ', my-name, >a1b2_c3 va “o'zgaruvchi qo'shtirnoqda"

O`zgaruvchi va konstantalarni qo`llanishiga misol:

```
i = 5
print(i)
```

```
i = i + 1
print(i)
s = "'Bu ko'p qatorlik satr.
Bu uning ikkinchi qatori."'
print(s)
```

Natija:

```
5
6
Bu ko'p qatorlik satr.
Bu uning ikkinchi qatori.
```

Yuqoridagi misolda dastlab biz 5 konstanta qiymatini '=' operatori yordamida *i* o'zgaruvchiga o'zlashtirib olamiz.

i = 5

so'ng *i* o'zgaruvchi qiymatini print funksiyasi orqali ekranga chop etamiz.

print(i)

i o'zgaruvchining qiymatiga 1 qo'shamiz va o'zgaruvchining o'ziga saqlaymiz. So'ng *i* o'zgaruvchining qiymatini chop etamiz.

i = i + 1

print(i)

Yuqoridagi kabi satr konstanta qiymatini *s* o'zgaruvchiga biriktiramiz va shundan so'ng uni chop etamiz.

s = '''Bu ko'p qatorlik satr.

Bu uning ikkinchi qatori.'''

print(s)

Eslatma: O'zgaruvchilar oddiy qiymat biriktirish bilan ishlatiladi. Hech qanday oldindan e'lon qilib qo'yish talab etilmaydi.

Operatorlar va ifodalar

Dasturdagi ko'p amallar (mantiqiy qatorlar) ifodalardan tashkil topgan. Bunga oddiy misol: 2 + 3. Ifodani operatorlar va operandlarga ajratish mumkin. **Operator** – bu biror amalni bajaruvchi va simvol yordamida yoki zaxiraga olingan so'zlar yordamida ifodalanadigan funksional. Operatorlar qiymatlar ustida biror amalni

bajaradi va bu qiymatlar **operandlar** deyiladi. Bizning xolatda 2 va 3 – bu operandlar.

Operator	Nomlanishi	Ta'rifi	Misol
+	Qo'shish	Ikkita ob'yektning yig'indisini hisoblaydi	3 + 5 ifoda 8 ni beradi; 'a' + 'b' ifoda 'ab' ni beradi.
–	Ayirish	Ikkata sonning farqini beradi. Agar birinchi operand mavjud bo'lmasa, uning qiymati 0 ga teng deb olib ketiladi.	-5.2 manfiy qiymat beradi, 50 – 24 ning qiymati esa 26 ga teng.
*	Ko'paytirish	Ikkita son ko'paytmasini beradi yoki satrni ko'rsatilgan miqdorda takrorlangan yangi satrni qaytaradi.	2 * 3 ifoda 6 beradi. 'xa' * 3 ifoda 'xaxaxa' ni qaytaradi.
**	Darajaga ko'tarish	x sonini y darajaga ko'tarilganda	3**4 ifoda 81 ni qaytaradi (ya'ni 3*3*3*3)

		hosil bo'lgan qiymatni qaytaradi.	
/	Bo'lish	'x' ni 'y' ga bo'lganda hosil bo'lgan bo'linmani qaytaradi.	4 / 3 ifoda 1.3333333333333333 ni beradi.
//	Qoldiqsiz bo'lish	Bo'lishdan hosil bo'lgan bo'linmaning qoldiqsiz butun qismini qaytaradi.	4 // 3 ifoda 1 ni qaytaradi.
%	Qoldiqlik bo'lish	Bo'lishdan hosil bo'lgan qoldiqni qaytaradi.	8 % 3 ifoda 2 ni beradi. -25.5 % 2.25 ifoda 1.5 ni beradi.
<<	Chapga surish	Bit sonni chapga ko'rsatilgan miqdorda suradi.	2 << 2 ifoda 8 ni beradi. Ikkilik sanoq tizimida 2 soni 10 ko'rinishiga ega bo'ladi. Chapga 2 bit miqdorida surish 1000 beradi, bu o'nlik sanoq tizimida 8 ni beradi.

>>	O'ngga surish	Bit sonni o'ngga ko'rsatilgan miqdorda suradi.	11 >> 1 ifoda 5 ni beradi. Ikkilik sanoq sistemasida 11 soni 1011 ko'rinishida bo'ladi uni 1 bit o'ngga sursak 101 hosil bo'ladi va bu onlik sanoq tizimida 5 ni beradi.
&	'Va' bit operatori (Побитовое И)	Sonlar ustida 'va' bit operatsiyasini bajaradi.	5 & 3 ifoda 1 ni beradi
|	'Yoki' bit operatori (Побитовое ИЛИ)	Sonlar ustida 'yoki' bit operatsiyasini bajaradi.	5 | 3 ifoda 7 ni beradi
^	'shartlik yoki' bit operatori (Побитовое ИСКЛЮЧИТЕЛЬНО ИЛИ)	Sonlar ustida 'shartlik yoki' bit operatsiyasini bajaradi.	5 ^ 3 ifoda 6 ni beradi
~	'Emas' bit operatori Побитовое НЕ	'Emas' bit operatsiyasi x soni uchun -(x+1) ga to'g'ri keladi.	~5 ifoda 6 ni beradi.

<	Kichik	X qiymat y qiymatdan kichikligini aniqlaydi. Hamma qiyoslash operatorlari True yoki False qaytaradi. Bu so'zlardagi katta xarflarga e'tibor bering.	5 < 3 False qaytaradi 3 < 5 ifoda esa True qaytaradi. Ixtiyoriy bir – biri bilan bog'langan ifodalar tuzish ham mumkin: 3 < 5 < 7 ifoda True ni qaytaradi
>	Katta	X qiymat y qiymatdan katta ekanligini aniqlaydi.	5 > 3 ifoda True ni qaytaradi.
<=	Kichik yoki teng	x qiymat y qiymatdan kichik yoki teng ekanligini aniqlaydi.	x = 3; y = 6; x <= y ifoda True qaytaradi.
>=	Katta yoki teng	x qiymat y qiymatdan katta yoki teng ekanligini anqlaydi.	x = 4; y = 3; x >= 3 ifoda True qaytaradi.

==	Teng	Ob'yektlarning tengligini tekshiradi	x = 2; y = 2; x == y ifoda True qaytaradi. x = 'str'; y = 'stR'; x == y ifoda False qaytaradi. x = 'str'; y = 'str'; x == y ifoda True qaytaradi.
!=	Teng emas	Ob'yektlar teng emasligi to'g'riligini tekshiradi.	x = 2; y = 3; x != y ifoda True qaytaradi.
Not	Mantiqiy 'emas' (Логическое НЕ)	Agar x True bo'lsa, operator False qaytaradi. Agar x False bo'lsa operator True qaytaradi.	x = True; not x ifoda False qaytaradi.
And	Mantiqiy 'va' (Логическое И)	x and y ifoda False qaytaradi agar x False bo'lsa. Aks holda y ning qiymatini qaytaradi.	x = False; y = True; x and y ifoda False qaytaradi, sababi x равно False. Bu holda Python y ning qiymatini tekshirib o'tirmaydi sababi 'and' operatoridan

			chapdagi ifoda qismi False ga teng va butun ifoda qiymati boshqa oparatorlar qiymatlariga bog'liqsiz ham False bo'ladi.
Or	Mantiqiy 'yoki'	x or y agar x True ga teng bo'lsa True qaytaradi aks xolda y ning qiymatini qaytaradi.	x = True; y = False; x or y ifoda True qaytaradi.

1.2.1-chizma. Operatorlar va ularning qo'llanilishi

Operatorlar va ularning qo'llanilishini qisqacha ko'rib chiqamiz. Misol uchun, arifmetik ifodalarini tekshirib ko'rish uchun interaktiv interpretatordan foydalanishimiz mumkin. Python interpretatori xuddi kalkulator kabi ishlaydi.

```
>>> 2+3
5
>>> 10.5-2.0
8.5
>>> 15-3
12
>>> 8*9
72
>>> 25/3
8.333333333333334
>>> 8**150
290735489718242756219729523155201813741456544274927224112596079672255715245359169330476420285505426224305008642506471173413840651445862 4
>>> 36%5
1
>>> 36//5
7
```

Matematik amallar va o'zlashtirishlarni qisqacha yozish

Ko'pincha bir o'zgaruvchi ustida biror matematik amal bajarib, natijani o'sha o'zgaruvchining o'ziga o'zlashtirish zaruriyati tug'iladi. Bu holda amallarni qisqacha yozish mumkin. Siz

a = 2; a = a * 3

ni quyidagicha yozishingiz mumkin:

a = 2; a *= 3

Amallar bajarilish ketma-ketligi

2 + 3 * 4 ifodada qaysi amal birinchi bajariladi: qo'shishmi yoki ko'paytirish?

Matematika fanida ko'paytirish birinchi bajarilishi ko'rsatilgan. Demak, ko'paytirish operatori qo'shish operatoriga qaraganda katta prioritetga(muhimlik darajasiga) ega. Quyidagi jadvalda Python operatorlari prioriteti ko'rsatilgan. Bunda yuqoridan pastga qarab Python operatorlari prioriteti oshib boradi. Bu shuni anglatadiki, ixtiyoriy ifodada Python oldin eng quyidagi operatorlarni hisoblaydi va keyin esa yuqoridagilarini. Amaliyotda esa amallarni qavslar bilan aniq ajratish tavsiya etiladi. Bu dastur kodini oson o'qishga yordam beradi.

Operator	Izoh
Lambda	lambda ifoda
Or	Mantiqiy 'yoki'
And	Mantiqiy 'va'
Not x	Mantiqiy 'emas'
in, not in	Tegishlilikni tekshirish
is, is not	Bir xillikni tekshirish
<, <=, >, >=, !=, ==	Taqqoslash

|	'yoki' bit operatori
^	'shartlik yoki' bit operatori
&	'va' bit operatori
<<, >>	Surilishlar
+, –	Qo'shish va ayirish
*, /, //, %	Ko'paytirish, bo'lish, qoldiqsiz bo'lish va qoldiqlik bo'lish
+x, -x	Musbat va manfiy
~x	'emas' bit operatori
**	Darajaga ko'tarish
x.attribute	Atributga link
x[index]	Indeks bo'yicha murojat
x[index1:index2]	Kesib olish
f(argumentlar …)	Funksiyani chaqirish
(ifoda, …)	Kortej (Связка или кортеж)
[ifoda, …]	Ro'yxat (Список)
{kalit:qiymat, …}	Lug'at (Словарь)

1.2.2-chizma. Python operatorlari prioriteti.

Bu jadvalda bir xil prioritetga ega bo'lgan operatorlar bir qatorda joylashgan. Misol uchun '+' va '-'.

Hisoblash tartibini o'zgartirish

Ifodalarni o'qishni osonlashtirish uchun qavslarni ishlatish mumkin. Misol uchun, 2 + (3 * 4) ni tushunish oson operatorlar prioriteni bilish lozim bo'lgan 2 + 3

* 4 ifodadan ko'ra. Qavslarni o'ylab ishlatish kerak. Ortiqcha qavslarni ishlatishdan saqlaning. Misol uchun: (2 + (3 * 4)).

Qavslarni ishlatishni ya'na bir afzalligi hisoblash tartibini o'zgartirish imkonini beradi. Misol uchun, qo'shish amalini ko'paytirish amalidan birinchi bajarish kerak bo'lsa, quyidagicha yozish mumkin:

(2 + 3) * 4.

If- shart operatori.

If operatori shartni tekshirish uchun ishlatiladi. Pythonda shart operatorini bir necha xil ko`rinishi mavjud:

1. if (mantiqiy ifoda):- shart operatorining bu ko`rinishi mantiqiy ifoda rost bo`lgan holda qandaydir kod bajarilishi uchun ishlatiladi.

2. if (mantiqiy ifoda):…**else-**shart operatorining bu ko`rinishida mantiqiy ifoda rost bo`lsa, birinchi ifodalar bloki bajariladi(bu blok "**if-blok**" deb nomlanadi), **aks holda** keyingi ifodalar bloki bajariladi(bu blok "**else-blok**" deb nomlanadi).

3. if (mantiqiy ifoda):…**elif(mantiqiy ifoda):**…**else-** shart operatorining bu ko`rinishida oldingi shart yolg`on bo`lganda keyingi shart tekshiriladi. Bu ifoda o`zida ikkita bir-biriga bog`liq bo`lgan **if else-if else** ifodani bir ifodada **if elif else** saqlaydi. Bu dasturni o`qishni osonlashtiradi.

If operatoriga misol:

```
baho=5
if baho==5:
    print(baho,"a'lo baho")
```

Natija

```
>>>
5 a'lo baho
>>>
```

Kiritilgan sonning juft toqligini aniqlash:

```
a=int(input("sonni kirit="))
if (a%2==0):
    print("juft son")
else:
        print("toq son")
```

Natija:

```
sonni kirit=5
toq son
```

Kiritilgan sonni musbat manfiyligini aniqlash:

```
a=int(input("sonni kirit="))
if (a>0):
    print("musbat son")
elif(a<0):
        print("manfiy son")
else:
        print("nolga teng")
```

Natija:

```
sonni kirit=-125
manfiy son
```

If opreratorini pythonda nafaqat shart operatori sifatida balki tanlash opretori (switch) sifatida ham qo`llash mumkin. Masalan:

A va B haqiqiy sonlarni hamda arifmetik amal belgisini kiritgandan keyin ifodani hisoblash dasturi:

```
a=int(input('1-sonni kiriting='))
b=int(input('2-sonni kiriting='))
k=input('amal belgisini kiriting=')
if k=='^': print(a**b)
elif k=='-': print(a-b)
elif k=='*': print(a*b)
elif k=='/': print (a/b)
else: print(a+b)
```

Natija:

```
1-sonni kiriting=25
2-sonni kiriting=35
amal belgisini kiriting=^
847032947254300339068322500679641962051391601562 5
# 25 ni 35 chi darajasini hisoblab chiqardi
```

Yil oylarining raqami kiritilgach oy necha kundan iborat ekanligini topish dasturi:

```
m = int(  input("yil oyining raqamini kiriting: ") )
if m == 2: d = 28
elif m in [1,3,5,7,8,10,12]: d = 31
else: d = 30

print( m,"-oy", d ,'kundan iborat')
```

Natija:

```
yil oyining raqamini kiriting: 2
2 -oy 28 kundan iborat
```

Pythonda rostlikka tekshirish

- Har qanday nolga teng bo`lmagan son yoki bo`sh bo`lmagan obyekt-rost
- Nol yoki bo`sh obyekt-yolg`on
- Taqqoslash amallari True yoki False qiymat qaytaradi
- Mantiqiy operatorlar and va or rost yoki yolg`on obyekt-operandni qaytadi

Mantiqiy operatorlar:

X and Y

Rost, agar x va y ham rost bo`lsa

X or Y

Rost, agar x yoki y dan bittasi rost bo’lsa

Not X

Rost, agar x yolg`on bo`lsa

While sikl operatori

While operatori quyidagi umumiy ko`rinishga ega:

While (shart) {
sikl_tanasi}

While sikl operatorining ishlash tartibi

Agar (shart) rost (**true**) qiymatga ega bo`lsa, **sikl_tanasi** bajariladi. Qachonki shart yolg`on (**false**) qiymatga teng bo`lsa sikl tugatiladi.

Agar (shart) true qiymatga ega bo`lmasa sikl tanasi biror marta ham bajarilmaydi.

Toq sonlarni ekranga chiqarish: **Natija:**

```
k=1
while k<=10:
    print(k)
    k+=2
```

```
1
3
5
7
9
```

For operatori

Python dasturlash tilida **for** operatori C va Paskal dasturlash tillarida qo`llanishidan farq qiladi. Python da **for** operatori biroz murakkabroq, lekin **while** sikliga qaraganda ancha tezroq bajariladi. **For…in** operatori obyektlar ketma-ketligida iteratsiyani amalga oshiradi, ya’ni bu sikl har qanday iteratsiya qilinadigan obyekt bo`ylab o`tadi(satr yoki ro`yxat bo`ylab) va har bir o`tish vaqtida sikl tanasini bajaradi.

For operatoriga misol:

```
for i in 'salom dunyo':
    print(i*2, end='')
```

```
ssaalloomm  dduunnyyoo
```

Range() va xrange funksiyasi

Agar dasturda sonlarni ketma-ket chiqarish kerak bo`lsa **range()** funksiyasidan foydalaniladi. U arifmetik progressiyaga asoslangan ro`yxat tuzadi.

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Range(10) funksiyasi 10 ta elementdan iborat bo`lgan ro`yxat tuzadi. Bunda yuqori chegara sifatida 10 beriladi, lekin u yaratilgan ro`yxat ketma-ketligiga kirmaydi. Shuningdek funksiyaga quyi chegara va qadamni ham berish mumkin.

```
>>> range(5,10)
[5, 6, 7, 8, 9]
```

```
>>> range(0,10,3)
[0, 3, 6, 9]
>>> range(-10,-100,-30)
[-10, -40, -70]
```

Ketma-ketlik indekslarini tanlash uchun range() va len() funksiyalarini birgalikda ishlating.

```
a=['Mariyaning','kichkina','qo`zichog`i','bor']
for i in range(len(a)):
    print (i, a[i])
```

```
0 Mariyaning
1 kichkina
2 qo`zichog`i
3 bor
```

Katta diapazondagi raqamlardan foydalanib ro`yxatni yaratish **range()** funksiyasi o`zini oqlamaydi yoki ba'zi hollarda xotira yetishmaydi.

```
>>> l=range(10000000)
Traceback (innermost last):
  File "<stdin>", line 1, in ?
MemoryError
```

Shunday hollarda Python da **xrange()** funksiyasidan foydalaniladi.

Break operatori

Break operatori agar siklning bajarilish sharti hali **False** qiymat olmagan bo'lsa ham yoki ketma-ketlik elementlari hali tugamagan bo'lsa ham siklni to'xtatish, ya'ni buyruqlar bajarilishini to'xtatish uchun xizmat qiladi. Shuni aytib o'tish kerakki, **for** yoki **while** sikllarini **break** operatori bilan to'xtatilsa, ularga tegishli bo'lgan **else** bloki bajarilmaydi.

Misol:

```
for i in "hello world":
    if i=='o':
        break
    print(i*2, end='')
```

Natija:

```
hhee
```

Continue operatori

Continue operatori joriy blokdagi o'zidan keyingi qolgan barcha buyruqlarni bajarmay siklning keyingi iteratsiyasidan davom ettirish uchun ishlatiladi.

Misol:

```
for i in "hello world":
    if i=='o':
        continue
    print(i*2, end='')
```

Natija:

```
hheellll  wwrrlldd
```

1.3. Funsiya, fayllar bilan ishlash, sanoq sistemasi va son.

Funksiyani aniqlash. Def kalit so`zi funksiyani aniqlashni taqdim etadi. **Def** so`zidan so`ng **funksiya nomi** va qavs ichida **formal parametrlar ro`yxati** ko`rsatiladi. Funksiya tanasini hosil qiluvchi instruksiyalar keyingi qatordan boshlab bo`sh joy(отступь) bilan yoziladi. Misol sifatida Fibonachi sonlar ro`yxatini chiqaradigan funksiyani yaratishimiz mumkin:

```
def fib(n):
    a, b = 0, 1
    while b < n:
        print( b),
        a, b = b, a+b  # funksiyaga tegishli blok

fib(2000) # funksiyani chaqirish
```

Natija:

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Yana bir misol sifatida kiritilgan sonning raqamlar yig`indisini hisoblovchi dasturni funksiya yordamida yechishni ko`rsak:

```
""" Fuksiyadan foydalanish  Kiritilishi:
    12345
 Natijani chiqarish:
    Raqamlar yigindisini:15
"""
def sumDigits(n):
  sum = 0
  while n != 0:
    sum += n % 10
    n = n // 10
  return sum

N = int ( input("son kiriting: ") )
print ( "sonning raqamlar yig`indisi=", sumDigits(N))
```

Natija:

```
son kiriting: 123
sonning raqamlar yig`indisi= 6
```

Python dasturiga kiritilgan funksiyalar. Tiplarni o`zgartiruvchi funksiyalar.

bool(x)- rostlikka tekshirishni standart usulidan foydalanuvchi bool tipiga o`zgartirish. Agar x yolg`on bo`lsa yoki tushirib qoldirilgan bo`lsa, False qiymatini qaytaradi, aksincha esa True qaytaradi.

bytearray([manba, [kodlash[xatolar]]])- bytearray ga o`zgartirish. Bytearray- 0<=x<256 diapazondagi butun sonlarni o`zgarmas ketma-ketligi. Konstruktor argumentlari bytearray() ga mos ko`rinishga ega bo`ladi.

complex([real],[image])- kompleks songa o`zgartirish.

dict(object)- lug`atga o`zg artirish.

float([x])-haqiqiy songa o`zgartirish. Agar argument ko`rsatilmagan bo`lsa, 0.0 qaytaradi.

frozenset([ketma-ketlik])

int([object],[asosiy sanoq sistemasi])- butun sonni berilgan sanoq sistemasidan o`nlik sanoq sistemasiga o`tkazish.

list([object])-ro`yxat tuzadi.

memoryview(object)- memoryview obyektini tuzadi.

object()-hamma obyektlarga asos bo`lgan bosh obyektni qaytaradi.

range([start=0], stop,[step=1])- step qadamli start dan stop gacha bo`lgan arifmetik progressiya.

set(object)-to`plamni yaratadi.

slice([start=0], stop, [step=1])-step qadamga ega startdan stopgachaga bo`lgan kesma obekti.

tuple(obj)- kortejga o`zgartiradi

Qo`shimcha funksiyalar

abs(x)- absolyut raqamni (sonni modulini) qaytaradi.

all(ketmaketlik)- agarda hamma elementlar haqiqiy bo`lsa (yoki ketmaketlik bo`sh bo`lsa) True ni qaytaradi.

any(ketmaketlik)-agarda elementlardan hech bo`lmaganda bittasi haqiqiy bo`lsa True ni qaytaradi. Bo`sh ketmaketlik uchun False qaytaradi.

ascii(object)- repr ga o`xshab obyekt ko`rinishiga mos qatorni ekranga xuddi shunday qaytaradi.

```
>>> ascii('A')
"'A'"
>>> ascii(64)
'64'
>>> repr('A')
"'A'"
>>> repr(64)
'64'
```

bin(x)- butun sonni ikkilik sanoq sistemasiga o`tkazadi

chr(x)- x ning Unicode ga mos belgini qaytaradi.

classmethod(x)- sinf metodi ko`rsatgan funksiyani taqdim etadi.

compile(source, filename, mode, flags=0, don't_inherit=False)- ketmaketlik eval yoki exec funksiyalari bilan bajariladigan dastur kodiga komplyatsiya qilinishi. Qator karetkani qaytaruvchi belgilar yoki nolga teng baytlarga ega bo`lmasligi kerak.

delattr(object, name)- "name" nomidan atributni qaytaradi.

dir([object])- obyekt nomlarining ro`yxati, agar obyekt ko`rsatilmagan bo`lsa, local maydondagi nomlar ro`yxati.

divmod(a,b) – a ni b ga bo`lganda hosil bo`lgan bo`linmaning butun va qoldiq qismi.

enumerate(iterable, start=0)- nomer va unga mos ketmaketlik a'zosidan tarkib topgan kortejni har bir o`tishda taqdim etuvchi iteratorni qaytaradi.

eval(expression, globals=None, locals=None)- dastur kodi qatorini bajaradi.

filter(function, iterable)- function yordamida rost qiymatni elementlarga qaytaruvchi iteratorni qaytaradi.

format(value [,format_spec])- formatlash (qatorni formatlash).

getattr(object, name,[default])- obyekt atributini yoki default.globals()-global nomlar lugatini chiqaradi.

hasattr(object, name)- "name" nomidagi atribut obyektga ega ekanligini tekshiradi.

hash(x)- ko`rsatilgan obyektning heshini qaytaradi.

help([object])- dasturni yordam qismiga kiritilgan ma'lumotnoma tizimini chaqirish.

hex(x)- butun sonni o`n oltilik sanoq sistemasiga o`tkazish.

id(object)-obyekt manzilini qaytaradi**.**

input([promt])- foydalanuvchi tomonidan kiritilgan qatorni qaytaradi. Promt-foydalanuvchiga yordam.

isinstance(object, ClassInfo)-agarda obyekt classinfo yoki uning sinfosti ekzemplari bo`lsa rost qiymat qaytaradi. Agarda ekzemplar berilgan tipdagi obyekt bo`lmasa, funksiya yolg`on qiymat qaytaradi.

issubclass(sinf, ClassInfo)-agarda sinf ClasssInfo sinfostisi bo`lsa rost qiymat qaytaradi. Sinf o`z-o`ziga sinfosti bo`ladi.

iter(x)- iterator obyektini qaytaradi.

len(x)-ko`rsatilgan obektni elementlar sonini qaytaradi.

locals()-lokal nomlar lug`ati.

map(function, iterator)-ketmaketlikning har bir elementiga function funksiyasini qo`llash orqali yaratiladigan iterator.

max(iter,[args…]*[, key])-ketma-ketlikning maksimal elementi.

min(iter,[args…]*[, key])-ketmaketlikning minimal elementi.

next(x)-iteratorning keyingi elementini qaytaradi.

oct(x)- butun sonni sakkizlik sanoq sistemasiga o`tkazadi.

open(file, mode='r', buffering=None, encoding=None, errors=None, newline=None, closefd=True)- faylni ochadi va kerakli oqimni qaytaradi.

ord(x)- belgi kodi.

pow(x, y[,r])-(x**y)%r.

reversed(object)-yoyilgan obyektning iteratori.

print([object,…],*,sep=” ”, end=’/n’, file=sys.stdout)- ma’lumotlarni ekranga chop etish.

round(X,[N])- verguldan keyin N- belgilargacha to`g`rilash.

setattr(obekt, nom, qiymat)- obyekt atributini belgilash.

sorted(iterable[, key][, reverse])- tartiblangan ro`yxat.

staticmethod(function)- funksiya uchun statistik metod.

sum(iter, start=0)-ketmaketlik elementlarini yig`indisi.

type(object)- obyekt tipini qaytaradi.

type(name, bases, dict)- name sinfidagi yangi ekzemplarni qaytaradi.

vars([object])- obyekt atributlarining ro`yxati. Jimlik holatida- local nomlar lug`ati.

Fayllar bilan ishlash

Fayllar bilan ishlash **file** klassi obyektini hosil qilish hamda uning **read**, **readline** va **write** metodlari yordamida amalga oshiriladi. Faylni o’qish yoki faylga yozish faylni ochish vaqtida ko’rsatilgan rejimga bog’liq. Fayl bilan ishlab bo’lgandan keyin **close** metodini chaqirish kerak bo’ladi.

```
poem = '''\
Dasturlash qiziqarli.
Agar ish zerikarli bo'lsa,
Unga quvnoq tus berish uchun - Pythonni ishlating!

'''
f = open('poem.txt', 'w') # faylni yozish uchun ochamiz (writing)

f.write(poem) # matnni faylga yozamiz

f.close() # faylni yopamiz

f = open('poem.txt')
# agar rejim ko'rsatilmagan bo'lsa,
# u xolda o'qish rejimi tushunilib ketiladi ('r'eading)

while True:
    line = f.readline()

if len(line) == 0: # Nol uzunlik faylning oxirini bildiradi (EOF)
    break
print(line, end='')

f.close() # faylni yopamiz
```

Natija:

```
Dasturlash qiziqarli.
Agar ish zerikarli bo’lsa,
Unga quvnoq tus berish uchun – Pythonni ishlating!
```

Bu misolda biz birinchi navbatda faylni rejim ko’rsatgan holda **open** funksiyasi bilan ochyapmiz. Rejim o’qish uchun («r»), yozish uchun («w») yoki fayl oxiriga yozuvni qo’shish uchun («a») bo’lishi mumkin. Faylni yana qanday holda o’qish, yozish yoki matn qo’shish holatini ham ko’rsatish mumkin: («t») tekst ko’rinishida yoki («b») binar ko’rinishida.

Bizning holatda faylni yozish («w») rejimida ochyapmiz va **write** metodi yordamida matnni faylga yozyapmiz. Shundan so’ng faylni **close** metodi yordamida yopyapmiz. So’ng xuddi shu faylni o’qish rejimida ochamiz. Bu holda rejimni ko’rsatishga hojat yo’q. Sababi agar rejim ko’rsatilmasa, fayl o’qish rejimida ochiladi. Faylni qatorma-qator **readline** metodi yordamida, sikl ichida o’qib olamiz. Qachonki bu metod bo’sh qator qaytarsa, u holda bu biz faylning oxiriga yetib borganimizni anglatadi va **break** yordamida siklni to’xtatamiz.

Shundan so’ng **print** funksiyasi yordamida o’qib olinayotgan satrlarni ekranga chop qilamiz. Oxirida **close** metodi yordamida faylni yopamiz. Haqiqatda dastur matnni faylga yozganligini tekshirish uchun **poem.txt** faylini tekshirib ko’ring.

Pickle

Pythonning **pickle** moduli yordamida har qanday obyektni faylga saqlash va keyinchalik fayldan o’qib olish mumkin. Bunday imkoniyat ob’yektlarni uzoq muddat saqlashda qo’l keladi.

```
import pickle
# obyektni saqlash fayli

shoplistfile = 'shoplist.data'

# xaridlar ro'yxati

shoplist = ['olma', 'mango', 'sabzi']

# faylga yozish

f = open(shoplistfile, 'wb')

pickle.dump(shoplist, f) # obyektni faylga yozamiz

f.close()

del shoplist # shoplist o'zgaruvchisini o'chirib tashlaymiz

# fayldan o'qish

f = open(shoplistfile, 'rb')

storedlist = pickle.load(f) # ob'yektni fayldan yuklab olish

print(storedlist)
```

Natija:

```
['olma', 'mango', 'sabzi']
```

Bu misolda obyektni faylga yozish uchun birinchi galda faylni binar yozish **(“wb”)** rejimida ochilyapti, so’ng pickle modulining dump funksiyasi chaqirilyapti. Bu jarayon “konservatsiya” (“pickling”) deyiladi. Shundan so’ng obyektni fayldan o’qib olish uchun pickle modulining **load** funksiyasidan foydalanilyapti.

Sanoq sistemasining ishlatilishi

Maktab kursidagi informatika faninidan bizga ma’lumki, sonlar nafaqat o`nlik sanoq sistemasida balki boshqa sanoq sistemalarida ham bo`lishi mumkin. Masalan: kompyuter ikkilik sanoq sistemasidan foydalanadi ya’ni 19-soni ikkilik sanoq sistemasida (kompyuterda) 10011 ko`rinishida ifodalanadi. Bundan tashqari sonlarni bir sanoq sistemasidan ikkinchi sanoq sistemasiga o`tkazish kerak. Python bu uchun bir qancha funksiyalarni taqdim etadi:

int([object],[sanoq sistemasi asosi])- butun sonni berilgan sanoq sistemasidan o`nlik sanoq sistemasiga o`tkazadi.

bin(x)- butun sonni ikkilik sanoq sistemasiga o`tkazadi

hex(x)- butun sonni o`n oltilik sanoq sistemasiga o`tkazadi

oct(x)- butun sonni sakkizlik sanoq sistemasiga o`tkazadi.

```
>>> bin(19)
'0b10011'
>>> oct(19)
'0o23'
>>> hex(19)
'0x13'
>>> int('10011',2)
19
>>> int('0b10011',2)
19
>>> a=int('19')# satrni songa o`tkazadi
>>> b=int(19.5)# haqiqiy sonni butun qismini qaytaradi
>>> print(a,b)
19 19
```

Son

Sonlar Python dasturlash tilida 3 turda bo'ladi:

1. butun sonlar,
2. haqiqiy sonlar
3. kompleks sonlar

- Butun songa misol *2*,5, …
- Haqiqiy sonlarga misol 3.23 va 52.3e-4.
- Kompleks sonlarga misol (-5+4i) va (2.3-4.6i)

Butun sonlar

Python interpretatorida yuqorida operator va ifodalar mavzusida ko`rib chiqqan barcha operatorlarni oddiy matemetika kursida ishlatilganidek bajarilishini ko`rdik. Ya’ni ko`paytirish, qo`shish, ayirish, bo’lish, darajaga ko`tarish va hokazo. Endi esa butun sonlar ustida bajarish mumkin bo`lgan qo`shimcha metodlarni ko`ramiz.

int.bit_length()- sonni oldidagi ishora va nollarni hisobga olmasdan uni ikkilik sanoq sistemasida taqdim etish uchun kerakli bo`lgan bitlar soni.

```
>>> n=-37
>>> bin(n)
'-0b100101'
>>> n.bit_length()
6
```

int.to_bytes(length, byteorder, *, signed=False)-shu sonni taqdim etuvchi baytlar qatorini qaytaradi.

```
>>> (1024).to_bytes(2, byteorder='big')
b'\x04\x00'
>>> (1024).to_bytes(10, byteorder='big')
b'\x00\x00\x00\x00\x00\x00\x00\x00\x04\x00'
>>> (-1024).to_bytes(10, byteorder='big', signed=True)
b'\xff\xff\xff\xff\xff\xff\xff\xff\xfc\x00'
>>> x=1000
>>> x.to_bytes((x.bit_length() // 8) + 1, byteorder='little')
b'\xe8\x03'
```

classmethod **int.from_bytes(bytes, byteorder, *, signed=False)**-berilgan baytlar qatoriga mos sonni qaytaradi.

```
>>> int.from_bytes(b'\x00\x10', byteorder='big')
16
>>> int.from_bytes(b'\x00\x10', byteorder='little')
4096
>>> int.from_bytes(b'\xfc\x00', byteorder='big', signed=True)
-1024
>>> int.from_bytes(b'\xfc\x00', byteorder='big', signed=False)
64512
>>> int.from_bytes([255, 0, 0], byteorder='big')
16711680
```

Haqiqiy sonlar

Haqiqiy sonlar ham butun sonlar qo`llab quvvatlovchi operatsiyalarni qo`llab quvvatlaydi. Haqiqiy sonlarni ishlatilishiga oddiy misol:

```
>>> c=150
>>> d=12.9
>>> c+d
162.9
>>> k=abs(d-c)# sonning moduli
>>> print(k)
137.1
>>> round(k)# yaxlitlash
137
```

Haqiqiy sonlar ustida amal bajarishda foydalanish mumkin bo`lgan qo`shimcha metodlar:

float.as_integer_ratio- shu haqiqiy son bilan juftlik munosabatida bo`lgan butun son.

float.is_integer()- ko`rsatgich butun son bo`lish bo`lmasligini tekshiradi.

float.hex()-float ni hex ga (o`n oltilik sanoq sistemasiga) o`tkazadi.

classmethod **float.fromhex(s)**- o`n oltilik sanoq sistemasisdan floatga otkazadi. Ya'ni float.hex() ni teskarisi.

```
>>> (12.9).is_integer()
False
>>> (13.0).is_integer()
True
>>> (13.0).as_integer_ratio()
(13, 1)
>>> (10.5).hex()
'0x1.5000000000000p+3'
>>> float.fromhex('0x1.5000000000000p+3')
10.5
```

Pythonda sonlar bilan ishlaydigan standart metodlardan tashqari bir qancha modullar ham bor. **Math** moduli- murakkab matematik funksiyalarni taqdim etadi:

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sqrt(81)
9.0
```

Random moduli tasodifiy sonlar generatorini tasodifiy tanlov funksiyasini amalga oshiradi:

```
>>> import random
>>> random.random()
0.3974041754203669
```

Kompleks son

Pythonda komleks sonlar ustida arifmetik amallarni butun va haqiqiy sonlar ustida bajarilgani kabi oddiy bajarish mumkin yani matematika kursida komleks sonlar ustida arifmetik amallar qanday bajarilsa xuddi shunga o`xshab bajariladi.

```
>>> x=complex(1,2)
>>> print(x)
(1+2j)
>>> y=complex(3,4)
>>> print(y)
(3+4j)
>>> z=x+y
>>> print(z)
(4+6j)
>>> z=x*y
>>> print(z)
(-5+10j)
>>> z=x/y
>>> print(z)
(0.44+0.08j)
>>> print(x.imag)# Mavhum qismi chiqaradi
2.0
>>> print(x.real)# Haqiqiy qismni chiqaradi
1.0
```

```
>>> pow(3+4j,2)# Darajasini hisoblaydi
(-7+24j)
>>> abs(3+4j)# Kompleks sonni moduli
5.0
>>>
```

1.4.Satrlar bilan ishlash

Satrlar – bu belgilar ketma-ketligi. Ko'p hollarda satrlar so'zlar jamlanmasidan tashkil topadi. Pythonda satrlar bilan ishlash juda qulay. Bir qancha satr literallari mavjud. Ularni ko`rib chiqamiz

Apostrof va qo`shtirnoqdagi satrlar

```
>>> s='spam"s'
>>> print(s)
spam"s
>>> s="spam's"
>>> print(s)
spam's
```

Apostrof va qo`shtirnoqdagi satrlar bir narsa. Uni ikki xil variantda keltirilishiga sabab literallarga apostrof va qo`shtirnoq belgilarini maxsus xizmatchi belgilardan foydalanmasdan kiritish mumkinligi deb hisoblanadi.

Ekran bilan ishlash ketma-ketliklari-xizmatchi belgilar

Ekran bilan ishlash ketma-ketliklari- klaviatura yodamida kiritish murakkab bo`lgan belgilarni yozishga imkon beradi.

Xizmatchi belgilar	Vazifasi
\n	Keyingi qatorga o`tish
\a	Qo`ng`iroq
\f	Keyingi betga o`tish
\r	Koretkani qaytarish
\t	Gorizontal tabulatsiya
\v	Vertical tabulatsiya
\N{id}	Unicode ma’lumotlar bazasining ID identifikatori
\uhhhh	Unicode ning 16 lik ko`rinishidagi 16 bitli belgisi
\Uhhhh. . .	Unicode ning 32 lik ko`rinishidagi 32 bitli belgisi
\xhh	Belgining 16 lik kodi
\ooo	Belgining 8 lik kodi
\0	Null belgisi (satr oxiri belgisi emas)

1.4.1-chizma. Ekran bilan ishlash ketma-ketliklari.

Ko`p qatorli satrlar

Pythonda satrlarni apostrof(‘) va qo`shtirnoqdan foydalanib hosil qilish mumkin. Apostrof (bir tirnoq(')) yoki qo'sh tirnoqni(") 3marta takrorlash orqali esa ko'p qatorlik satrlarni xosil qilish mumkin. Milsol uchun:

```
>>> "'Bu ko'p qatorlik satr.Bu uning birinchi qatori.
Bu uning ikkinchi qatori.
“Isming kim?", - so'radim men.
U javob berdi: “Bond, James Bond."
"'
```

Satr konstantalarini birlashtirish uchun ularni yonma-yon joylashtirishning o'zi kifoya. Python avtomat ularni birlashtiradi. Misol uchun: "Ismingiz" "kim?" avtomat "Ismingiz kim?" ga aylanadi.

Eslatma: Bir tirnoq va qo'sh tirnoqdagi satrlar bir-biridan hech ham farq qilmaydi.

Satrlarning funksiya va metodlari

Shunday qilib satrlar bilan ishlash haqida gapirdik, endi satrlarning funksiyalari va metodlari haqida gapiramiz. Quyida satrlarning barcha funksiya va metodlari keltirilgan.

Asosiy operatsiyalar

Konkatenatsiyalash (qo`shish)

```
>>> s1='spam'
>>> s2='eggs'
>>> s1+s2
'spameggs'
```

Satrni takrorlash (dublikat qilish)

```
>>> print('dunyo'*3)
dunyodunyodunyo
```

Satr uzunligi (len() funksiyasi)

```
>>> gap='bu satrning uzunligi qancha'
>>> len(gap)
27
```

Indeks bo`yicha chiqarish

```
>>> s='spam'
>>> s[0]
's'
>>> s[2]
'a'
>>> s[-2]
'a'
```

Misoldan ko`rinib turibidiki Python manfiy indeks bo`yicha chiqarishga ruxsat etadi, lekin hisoblash qator oxiridan boshlanadi.

Kesmani ajratib olish. Kesmani ajratib olish operatori:[X:Y]. X- kesmaning boshi, Y esa –oxiri. Y raqamli belgi kesmaga kirmaydi. Jimlik holatida birinchi indeks 0 ga teng, ikkinchi indeks esa qator uzunligiga teng bo`ladi.

```
>>> s='spameggs'
>>> s[3:5]
'me'
>>> s[2:-2]
'ameg'
>>> s[:6]
'spameg'
>>> s[1:]
'pameggs'
>>> s[:]
'spameggs'
```

Bundan tashqari kesmani ajratib olishda qadamni belgilash mumkin

```
>>> s[::-1]
'sggemaps'
>>> s[3:5:-1]
''
>>> s[2::2]
'aeg'
```

Satrlarning qo`shimcha funksiya va metodlari

Metodlarni chaqirganga Pythondagi satrlar o`zgarmaydigan ketma-ketliklar darajasiga kirishini inobatga olishimiz kerak. Bu degani hamma funksiyalar va metodlar faqat yangi satrni tuzishi mumkin.

```
>>> s='spam'
>>> s[1]='b'
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    s[1]='b'
TypeError: 'str' object does not support item assignment
>>> s=s[0]+'b'+s[2:]
>>> s
'sbam'
```

Shuning uchun hamma metodlar yangi satrni qaytaradilar, va u keyin boshqa nomga ega bo`ladi.

S = ‘str’; S = “str”; S = ‘’‘str’‘’; S = “”“str”“”- Satrlarni literallari

S = “s\np\ta\nbbb”- ekran bilan ishlash ketma-ketliklari

S = r”C:\temp\new”- Formatlashtirilmagan satrlar

S = b”byte”- Baytlar qatori

S1+S2- Konkatenatsiya (qo`shish)

S1*3- Satrni takrorlash

S[i]- Indeks bo`yicha murojaat

S[i:j:step]- Step qadamli i elementdan boshlab j elementgacha bo`lgan kesmani ajratib olish.

```
>>> qator='menga pyhton dasturida ishlash yoqadi'
>>> qator[1:5]
'enga'
>>> qator[2:15:3]
'n hna'
>>> qator[0:15:2]
'mnapho a'
```

Len(s)- Satr uzunligi

```
>>> gap='bu satrning uzunligi qancha'
>>> len(gap)
27
```

S.find(str,[start],[end])- Satrdan satr ostini izlash. Satr ostining birinchi belgisini o`rinini qaytaradi, agar satrda satr osti bo`lmasa -1ni qaytaradi.

```
>>> s='salom bu Python dasturi'
>>> s.find('Python',1, 5)
-1
>>>
>>> s.find('Python',1, 50)
9
```

S.rfind(str,[start],[end])- Satrdan satr ostini axtarish. Oxirgi kirish raqamini yoki 1 ni qaytaradi

S.index(str,[start],[end])- Satrdan satr ostini axtarish. Birinchi kirish raqamini qaytaradi yoki ValueError istisnosini chaqiradi

S.rindex(str,[start],[end])- Satrdan satr ostini axtarish. Oxirgi kirish raqamini qaytaradi yoki ValueError istisnosini chaqiradi

S.replace(shablon,almashuv)- Shablonni almashtirish

S.split(belgi)- Satrni har bir so`zini alohida alohida ajratib chiqish.

```
matn='bu python dasturlash tili hozir matndagi so`zlarni ajratamiz'
matn.split()
['bu', 'python', 'dasturlash', 'tili', 'hozir', 'matndagi', 'so`zlarni', 'ajratamiz']
```

S.isdigit()-Satrda raqamlar ishtirok etganligini tekshirish.

S.isalpha()-Satr faqat harflardan iboratligini tekshirish

S.isalnum()-Satr harf yoki raqamlardan iboratligini tekshiradi

S.islower()-Satr quyi registrdagi belgilardan iboratligini tekshiradi

```
>>> satr='bu satrdagi so`zlar faqat quyi registrda yozilgan'
>>> satr.islower()
True
>>> satr='Bu satrdagi so`zlar faqat quyi registrda yozilgan Emas'
>>> satr.islower()
False
```

S.isupper()-Satr yuqori registrdagi belgilardan iboratligini tekshiradi

```
>>> k='SATRDA SO`ZLAR YUQORI REGISTRDA YOZILGAN'
>>> k.isupper()
True
>>> k='SATRDA So`zlar YUQORI REGISTRDA YOZILGAN'
>>> k.isupper()
False
```

S.isspace()-Satrda ko`rinmaydidan belgilar borligini tekshirish (probel, sahifani o`tkazish belgisi(‘\p’), yangi satrga o`tish(‘\n’), koretkani qaytarish(‘\r’), gorizontal tabulyatsiya(‘\t’) va vertikal tabulyatsiya)

```
satr=('salom\n mening ismim Guli\r bu Python dasturi')
print(satr)
print(satr.isspace())

salom
 mening ismim Guli
 bu Python dasturi
False
```

S.istitle()-Satrda so`zlar bosh harf bilan boshlanishini tekshirish

```
>>> s='Salom Dunyo Bu Python'
>>> s.istitle()
True
>>> s='salom dunyo Bu Python'
>>> s.istitle()
False
```

S.upper()-Satrni yuqori registrga o`zgartirish

```
>>> satr='bu satrda kichik harfda yozildi'
>>> satr.upper()
'BU SATRDA KICHIK HARFDA YOZILDI'
```

S.lower()-Satrni quyi registrga o`zgartirish

```
>>> s='BU UPPER FUNKSIYASIGA TESKARI FUNKSIYA'
>>> s.lower()
'bu upper funksiyasiga teskari funksiya'
```

S.starswith(str)- S satr str shablonidan boshlanishini tekshirish

S.endswith(str)- S satr str shabloni bilan tugashini tekshirish

**S.join(ro`yxat)-** S ajratuvchiga ega ro`yxatdan qatorni yig`ish

Ord(belgi)- Belgiga mos ASCII kodni qaytaradi

```
>>> ord('a')
97
>>> ord('A')
65
```

Chr(son)- ASCII kodga mos belgini qaytaradi

```
>>> chr(65)
'A'
>>> chr(97)
'a'
```

S.capitalize()-Satrning birinchi belgisi yuqori registrda qolganlarini quyi registrga o`tkazadi.

```
>>> satr='bu satrda birinchi belgi yuqori registrda bo`ladi'
>>> satr.capitalize()
'Bu satrda birinchi belgi yuqori registrda bo`ladi'
```

S.center(width,[fill])- Chegaralari bo`yicha fill (jimlik holatida probel) belgisi turuvchi markazlashtirilgan satrni qaytaradi.

S.expandtabs(tabsize)- Joriy ustungacha bir yoki bir qancha probellar bilan tabulyatsiyaning hamma belgilari almashtirilgan satr nusxasini qaytaradi. Agarda TabSize ko`rsatilmagan bo`lsa tabulyatsiya hajmi 8 probelga teng bo`ladi

S.lstrip([chars])- Satr boshidagi probel belgilarini olib tashlash

S.rstrip([chars])- Satr oxiridan probel belgilarini olib tashlash

S.strip([chars]) Satr boshidan va oxiridan probel belgilarini olib tashlash

S.partition(shablon)- Birinchi shablon oldida turuvchi qismni keyin shablonni o`zini va shablondan keyin turuvchi qismga ega kortejni qaytaradi. Agarda shablon topilmasa satrga ega bo`lgan kortejni qaytaradi, avval ikki bo`sh satr keyin satrni o`zini.

S.rpartition(sep)- Oxirgi shablon oldida turuvchi qismni keyin shablonni o`zini va shablondan keyin turuvchi qismni qaytaradi. Kortej qator o`zidan va undan keyin ikkita bo`sh qatordan iborat bo`ladi.

S.swapcase()-Quyi registrdagi belgilarni yuqori registrga, yuqorilarni esa quyiga o`tkazadi

```
>>> s='salom bu PYTHON DASTURI'
>>> s.swapcase()
'SALOM BU python dasturi'
```

S.title()-Har bitta so`zning birinchi harfini yuqori registrga qolganlarini esa quyi registrga o`tkazadi

```
>>> satr='bu satrda kichik harfda yozildi'
>>> satr.title()
'Bu Satrda Kichik Harfda Yozildi'
```

S.zfill(width)- Qator uzunligini Widthdan kam qilmaydi agar kerak bo`lsa birinchi belgilarni nollar bilan to`ldiradi.

```
>>> s='python'
>>> s.zfill(20)
'00000000000000python'
...
```

Xulosa.

Birinchi bobimiz “Python dasturlash tili va uning sintaksisi” deb nomlanadi. Bunda Python dasturlash tilining yaratilish tarixi, imkoniyatlari va Python dasturlash tilini Windows operatsion tizimida o`rnatish haqida yozilgan. Python dasturida ishlaydigan foydalanuvchilar uchun uning sintaksisi, asosiy operatorlari, fayllar, funksiyalar bilan ishlash sanoq sistemalari va satrlar bilan ishlash haqida muhim zaruriy ma’lumotlar keltirilgan.

Bu bobni o`qib, o`rgangan har bir qiziquvchi Python dasturini o`rnatish boshqa odatiy dasturlarni o`rnatish kabi hech qanday qiyinchiliksiz o`rnatilishini, tilning sintaksisi o`zi kabi sodda va oson ekanligini, o`zgaruvchilarning tipini e’lon qilinmasligini, shuningdek sonlar bilan ishlaganda nafaqat butun va haqiqiy sonlar ustida balki kompleks sonlar ustida ham amallar bajarishni ko`rsatilgan misollar yordamida o`rgana oladi va uni amaliyotda bajara oladi.

Satrlar bilan ishlash va ular ustida amallar bajarish haqida ham yetarlicha ma’lumotlar keltirilgan.

II BOB. PYTHONDA MA'LUMOTLAR TUZILMASI VA TILNING STANDART MODULLARI.

2.1. Ro`yxat va kortej

Ro`yxat- Pythonda erkin turdagi obyektlarning o`zgaruvchan qatorlashgan kolleksiyasi hisoblanadi (massivga o`xshash, lekin tiplar har xil bo`lishi mumkin). Ro`yxatlardan foydalanish uchun ularni tuzish kerak. Ro`yxatni har xil yondashuvlar yordamida yaratish mumkin. Masalan har bir iteratsiya qilinadigan obyektni (masalan satrni) Pythonni o`ziga kiritilgan **list** funksiyasi yordamida kiritish mumkin.

```
>>> list('ro`yxat')
['r', 'o', '`', 'y', 'x', 'a', 't']
```

Ro`yxatni yana literallar yordamida tuzish mumkin.

```
>>> s=[] #bo`sh ro`yxat
>>> l=['r','o','`','y',['xat'],2]
>>> s
[]
>>> l
['r', 'o', '`', 'y', ['xat'], 2]
```

Misoldan ko`rinadiki ro`yxat istalgancha obyektdan yoki hech narsadan (bo`sh) tashkil topishi mumkin.

Ro`yxat yaratishning yana bir usuli- ro`yxatlarning generatorlari. Ro`yxat generatori bu- ketma-ketlikni har bir elementiga arifmetik amalni qo`llab yangi ro`yxat tuzish usuli. Generatorlar for sikliga juda o`xshash bo`ladi.

```
>>> list('ro`yxat')
['r', 'o', '`', 'y', 'x', 'a', 't']
```

Ro`yxatlar generatorining juda murakkab konstruksiyalari bor.

```
>>> c = [c * 3 for c in 'list' if c != 'i']
>>> c
['lll', 'sss', 'ttt']
>>> c = [c + d for c in 'list' if c != 'i' for d in 'spam' if d != 'a']
>>> c
['ls', 'lp', 'lm', 'ss', 'sp', 'sm', 'ts', 'tp', 'tm']
```

Ro`yxatning funksiya va metodlari

Ro`yxatni yaratgandan so`ng uning ustida turli amallarni bajarish kerak bo`ladi, albatta, buning uchun esa Pythonni o`ziga kiritilgan bir qancha funksiya va metodlar bor.

Metod	Vazifasi
List.append(x)	Ro`yxat oxiridan element qo`shish
List.extend(L)	Oxiriga hamma elementlarni qo`shib list ro`yxatini kengaytiradi.
List.insert(i,x)	i-elementga x qiymatini kiritadi
List.remove(x)	Ro`yxatdan x qiymatga ega elementni o`chiradi
List.pop([i])	Ro`yxatning i-elementini o`chiradi va qaytaradi. Agarda indeks ko`rsatilmagan bo`lsa oxirgi element o`chiriladi
List.index(x,[start],[end])	X qiymatga teng start dan end gacha birinchi elementni qaytaradi
List.count(x)	X qiymatga teng elementlar sonini qaytaradi
List.sort([key=funksiya])	Funksiya asosida ro`yxatni saralaydi
List.reverse()	Ro`yxatni ochadi
List.copy()	Ro`txatning nusxasi
List.clear()	Ro`yxatni tozalaydi

2.1.1-chizma. Ro`yxat metodlari tasnifi

Ro`yxatda metodlarni qo`llanilishini misollar yordamida ko`rib chiqsak

```
>>> a=[] # bo`sh ro`yxat
>>> a
[]
>>> a.append(3)# ro`yxatga element qo`shish
>>> a
[3]
>>> a=a+[2]
>>> a
[3, 2]
>>> a+=[4]
>>> a
[3, 2, 4]
>>> b=[1,6]
>>> b
[1, 6]
>>> a.extend(b)#ikkita ro`yxatni qo`shish
>>> a
[3, 2, 4, 1, 6]
>>> a+=b
>>> a
[3, 2, 4, 1, 6, 1, 6]
>>> k=['u','f','d','f']
>>> k.count('f')# ro`yxxatda f elementi soni
2
>>> k.index('d')# ro`yxatdagi d elementining indeksi
2
>>> k.insert(2,'g')#ro`yxatni 2-indeksiga g elementini qo`shish
>>> k
['u', 'f', 'g', 'd', 'f']
>>> k.pop(2)#ro`yxatdan 2-indeksdagi elementni o`chirib qaytarish
'g'
>>> k
['u', 'f', 'd', 'f']
>>> k.remove('d')#ro`yxatdan d qiymatli elementni o`chirish
>>> k
['u', 'f', 'f']
>>> k.clear()#ro`yxatni tozalash
>>> k
[]
```

Ro`yxatning ishlatilishiga misol: bunda ro`yxatning maksimal elementini topish 2 xil usulda max() standart funksiyasi yordamida va har bir elementni birma-bir solishtirib chiqish orqali yechilgan.

```
""" massivdan maxsimal elementni topish
 kiritish:
   1 2 8 4 5
 natija:
   maxsimum: A[3]=8
   maxsimum: A[3]=8
"""
print ( "massiv elementlarini kiriting:" )
A = list( map(int, input().split()) )
N = len(A)

nMax = 0
for i in range(1,N):
  if A[i] > A[nMax]:
    nMax = i
print ( "Maxsimum: A[", nMax, "]=", A[nMax], sep="" )

M = max(A)
nMax = A.index(M)
print ( "Maxsimum: A[", nMax, "]=", M, sep = "" );
```

Natija:

```
massiv elementlarini kiriting:
45 16 12 8 9 63
Maxsimum: A[5]=63
Maxsimum: A[5]=63
```

Kortejlar(tuple)

Kortejlar bir nechta ob’yektlarni birgalikda saqlashga xizmat qiladi. Ularni ro’yxatlarga o’xshatish mumkin. Lekin ular ro’yxatlar kabi boy funksionallikka ega emas. Ularning asosiy jihati qatorlarga o’xshab o’zgarmasliklaridir. Kortej-elementlar orasini vergul bilan ajratish orqali hosil qilinadi. Kortejga ma’no jihatdan o`zgarmas ro`yxat deb ta’rif berdik. Shu o`rinda savol tug`iladi. Ro`yxat bo`lsa kortej nimaga kerak:

1. Turli holatlardan himoyalanish. Bu degani kortej o`zgartirishlardan himoyalangan bo`ladi, rejali (bu yomon) va tasodifiy (bu yaxshi) o`zgarishlardan xalos bo`ladi.

2. Kichik hajm. So`zlar bilan ifodalamasdan.

```
>>> a = (1, 2, 3, 4, 5, 6)
>>> b = [1, 2, 3, 4, 5, 6]
>>> a.__sizeof__()
72
>>>
>>> b.__sizeof__()
88
```

3. Kortejdan lug`at kaliti sifatida foydalanish mumkin:

```
>>> d = {(1, 1, 1) : 1}
>>> d
{(1, 1, 1): 1}
>>> d = {[1, 1, 1] : 1}
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    d = {[1, 1, 1] : 1}
TypeError: unhashable type: 'list'
```

Kortej afzalliklari haqida bilib oldik. Endi kortej bilan qanday ishlashni ko`ramiz. Bu xuddi ro`yxatlar bilan ishlashga o`xshaydi. Bo`sh kortejni yaratamiz:

```
>>> a=tuple()#Pythonning standart Tuple funksiyasi yordamida
>>> a
()
>>> a=()# kortej literali yordamida
>>> a
()
```

Bir elementli kortejni yaratamiz:

```
>>> a=('s')
>>> a
's'
```

Stop. Satr paydo bo`ldi. Bizga esa kortej kerak. Qanday qilib kortejga ega bo`lamiz?

```
>>> a=('s',)
>>> a
('s',)
```

Ura! Ishladi! Hamma gap vergulda. Qavuslar bu yerda hech nimani anglatmaydi. Kortejni yana quyidagicha yatish mumkin.

```
>>> a='s',
>>> a
('s',)
```

Lekin qavus qo`yish lozim, ba’zi bir holatlar borki qavuslar juda kerak. Tuple() funksiyasi yordamida iteratsiyalangan obyektdan kortej yaratish mumkin:

```
>>> a=tuple('salom, dunyo')
>>> a
('s', 'a', 'l', 'o', 'm', ',', ' ', 'd', 'u', 'n', 'y', 'o')
```

Kortejning elmentlariga ham ro`yxatlardagi kabi indeks bo`yicha murojat qilish mumkin (masalan: element nechanchi indeksdaligini bilish uchun). Bunda ham indekslash 0 (nol) dan boshlanadi ya’ni 6 ta elementdan iborat kortejning

indeksi 0..5 gacha bo`ladi. Kortej indeksi esa [] operatori orasiga beriladi. Agar bizga kortejning elementilarini teskaridan chiqarish kerak bo`lsa uning indeksini manfiy (masalan: kortej[-5]) beramiz. Quyidagi misolda shularni ko`rib chiqamiz.

```
k=tuple('Salom')
print(k)
print('kortejni birinchi elementi:', k[0])
print('kortejni oxirgi elementi:',k[4])
print()
#kortej elementtiga teskari tartibda murojaat

print('kortejni birinchi elementi:', k[-5])
print('kortejni oxirgi elementi:',k[-1])
```

Natija:

```
('S', 'a', 'l', 'o', 'm')
kortejni birinchi elementi: S
kortejni oxirgi elementi: m

kortejni birinchi elementi: S
kortejni oxirgi elementi: m
```

Kortej kesmasi

Kortejning biror qismini bo`lagini ya'ni kesmasini ajratish uchun(masalan: 2 elementidan to 5-chi elementigacha) uning indekslari orasiga-ikki nuqta(:) qo'yamiz.

```
kortej = ('p','r','o','g','r','a','m','i','s','t')

# 2-chidan 4-chi elemngacha chiqarish:('r', 'o', 'g')
print(kortej[1:4])

#boshidan 2 ta elementni chiqarish:('p', 'r')
print(kortej[:-8])

# 8-chi elementdan to oxirigacha chiqarish:('i', 's','t')
print(kortej[7:])

# boshidan oxirigacha barcha elementlarni
#chiqarish: ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
print(kortej[:])
```

Natija:

```
('r', 'o', 'g')
('p', 'r')
('i', 's', 't')
('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 's', 't')
```

Kortejni o`zgartirish

Kortejlar ustida amal bajarayotganda "+" operatori orqali ikkita kortejni qo`shishimiz mumkin, va bu konkatenatsiya deb ham ataladi.

Kortej elementlarini takrorlash uchun “*” operatoridan foydalanamiz. “+” va “*” operatorlarini kortejlarda qo`llaganimizdan so`ng yangi ko`rinishdagi kortejga ega bo`lamiz

```
# konkatenatsiyani chiqarish:(1, 2, 3, 4, 5, 6)
print((1, 2, 3) + (4, 5, 6))

# takrorlash:('takror', 'takror', 'takror')
print(("takror",) * 3)
```

Kortejning funksiya va metodlari

Count(x)-kortejdagi x elementi sonini qaytaradi.

Index(x)-kortejdagi x elementining indeksini qaytaradi.

Any()-agar kortej elementi mavjud bo`lsa True qiymat qaytaradi, aks holda (kortej bo`sh bo`lsa) False qiymat qaytaradi.

Max()-kortejning maksimal elementini qaytaradi.

Min()- kortejning minimal elementini qaytaradi.

Len()-kortejning uzunligini qaytaradi.

Sorted()-kortej elementlaridan iborat yangi tartiblangan ro`yxatni qaytaradi.

Sum()-kortej elementlari yig`indisini qaytaradi.

```
k=('2','12','9','78','15','12',)
j=tuple()
print(k)
print('kortejdagi x element soni',k.count('12'))
print('kortejdagi x element indeksi',k.index('78'))
print('kortejni tekshirish',any(k))
print('kortejni tekshirish',any(j))
print('max element:',max(k))
print('min element:',min(k))
print('kortej uzunligi:',len(k))

('2', '12', '9', '78', '15', '12')
kortejdagi x element soni 2
kortejdagi x element indeksi 3
kortejni tekshirish True
kortejni tekshirish False
max element: 9
min element: 12
kortej uzunligi: 6
['12', '12', '15', '2', '78', '9']
```

Elementni kortejga tegishli ekanligini tekshirish uchun ***in*** kalit so`zidan foydalaniladi:

```
kortej=('s','a','l','o','m',)
print('o' in kortej)
```

Natija:

```
True
```

2.2. Lug`atlar va to`plamlar bilan ishlash

Pythondagi lug`atlar kalit bo`yicha kirishga ruxsat etuvchi erkin obyektlarning tartiblangan jamlanmasi. Ularni yana assotsiativli massivlar yoki hesh jadvallar deb nomlaydilar. Soddaroq qilib aytadigan bo`lsak lug'at xuddi manzillar kitobiga o'xshaydi, ya'ni biror insonning ismini bilgan holda uning manzili yoki u bilan bo'g'lanish ma'lumotlarini olish mumkin. Lug`atlar bilan ishlash uchun ham albatta uni yaratish kerak. Lug`atni yaratishni esa bir necha usullari bor. **Birinchidan** literal yordamida:

```
>>> d={}
>>> d
{}
>>> d={'dict':1,'lug`at':2}
>>> d
{'dict': 1, 'lug`at': 2}
```

Ikkinchidan dict() funksiyasi yordamida:

```
>>> d=dict(qisqa='dict', uzun='dictionary')
>>> d
{'uzun': 'dictionary', 'qisqa': 'dict'}
>>> d=dict([(1,1),(2,4)])
>>> d
{1: 1, 2: 4}
```

Uchinchidan fromkeys metodi orqali:

```
>>> d=dict.fromkeys(['a','b'])
>>> d
{'a': None, 'b': None}
>>> d=dict.fromkeys(['a','b'],100)
>>> d
{'a': 100, 'b': 100}
```

**To`rtinchidan** lug`at generatori yoradamida ular ro`yxat generatoriga juda o`xshash:

```
>>> d={a: a**2 for a in range(7)}
>>> d
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}
```

Lug`at metodlari

Dict.clear()- lug`atni tozalaydi.

Dict.copy()-lug`at nusxasini qaytaradi.

Classmethod **dict.fromkeys(seq[, value])-** Seq dan kalitni va Value qiymatlariga ega bo`lgan lug`atni yaratadi.

Dict.get(key[, default])-kalit qiymatini qaytaradi, lekin u bo`lmasa xatolik beradi, default (jimlikda None) qaytaradi.

Dict.items()-juftliklarni qaytaradi(kalit, qiymat)

Dict.keys()- lug`atdagi kalitlarni qaytaradi

Dict.pop(key[default])-kalitni yo`qotib qiymatni qaytaradi. Agarda kalit bo`lmasa defaultni qaytaradi.

Dict.popitem()- juftlikni o`chirib qaytaradi (kalit, qiymat). Agarda lug`at bo`sh bo`lsa KeyError istisnoni chaqiradi. Esingizda tursin lug`atlar tartibli emas.

Dict.setdefault(key [, default])-kalit qiymatni qaytaradi, lekin u bo`lmasa xatolik bermaydi, default qiymatga ega kalitni yaratadi (jimlikda None).

Dict.update([other])- other dan juftliklarni (kalit, qiymat) kiritib lug`atni to`ldiradi. Mavjud bo`lgan kalitlar qaytadan yoziladilar. None (eski lug`at) qaytaradi.

Dict.values()-lug`atdagi qiymatni qaytaradi.

```
d=dict(ismi='Gulnoza', yoshi='8', maktabi='1')
print()
print('lug`atning qiymati:',dict.values(d))
print()
print('lugatdagi juftliklar yani kalit va uning qiymatlari:',dict.items(d))
print()
print('lugatning kalitlari:',dict.keys(d))
print()
print('lugatning nusxasi:',dict.copy(d))
```

natija:

lug`atning qiymati: dict_values(['Gulnoza', '8', '1'])

lugatdagi juftliklar yani kalit va uning qiymatlari: dict_items([('ismi', 'Gulnoza'), ('yoshi', '8'), ('maktabi', '1')])

lugatning kalitlari: dict_keys(['ismi', 'yoshi', 'maktabi'])

lugatning nusxasi: {'ismi': 'Gulnoza', 'yoshi': '8', 'maktabi': '1'}

To`plam (set & frozenset)

Pythondagi to`plam- tasodifiy tartibda va takrorlanmaydigan elementlardan tashkil topgan “konteyner” deyiladi. To`plamni yaratamiz:

```
>>> a=set()
>>> a
set()
>>> a=set('hello')
>>> a
{'e', 'l', 'h', 'o'}
>>> a={'a', 'b', 'c', 'd'}
>>> a
{'c', 'a', 'b', 'd'}
>>> a={i**2 for i in range(10)} # ko`plik generatori
>>> a
{0, 1, 64, 4, 36, 9, 16, 49, 81, 25}
>>> a={}# bunday qilmang!
>>> type(a)
<class 'dict'>
```

Misoldan ko`rinadiki to`plam lug`atdagi literalga o`xshash literalga egalik qiladi, lekin leteral yordamida bo`sh to`plamni yaratib bo`lmaydi.

To`plamdan takrorlanadigan elementlardan qutulish uchun foydalanish qulay bo`ladi:

```
>>> words=['salom', 'aka', 'salom', 'uka']
>>> set(words)
{'aka', 'uka', 'salom'}
```

To`plamlar bilan bir qancha operatsiyalarni bajarish mumkin: birlashmalalarni topish, kesimlarni topish…

Len(s)- to`plamdagi elementlar soni(to`plam hajmi).

X in s- ‘x’ ‘s’ to`plamga tegishli bo`ladimi yo`qmi shuni tekshiradi

Set.isdisjoint(other)-agarda set va other umumiy elementlarga ega bo`lmasalar rost qiymat qaytaradi.

Set==other- set ning hamma elementlari otherga tegishli bo`ladilar otherni hamma elementlari setga tegishli bo`ladilar.

Set.issubset(other) yoki **set<=other-**set ning hamma elementlari other ga tegishli bo`ladilar.

Set.issuperset(other) yoki **set>=other-**analogik holat.

Set.union(other, …) yoki **|other|…-**bir qancha to`plamlar birlashmasi.

Set.intersection(other, …) yoki **&other&…-** kesib olish.

Set.difference(other, …) yoki **-other-…-**other ga tegishli bo`lmagan set ning hamma elementlar to`plami.

Set.symmetric_difference(other); set^other- birinchi to`plamda uchraydigan, lekin ularning ikkala to`plamning kesishmasida uchramaydigan elementlar.

Set.copy-to`plam nusxasi

To`plamni to`g`ridan-to`g`ri o`zgartiradigan operatsiyalar

Set.update(other, …); set|=other| …- to`plam birlashmasi

Set.intersection_update(other, …); set&=other&…- to`plam kesishmasi

Set.difference_update(other, ...); set -= other | ...-to`plam ayirmasi

Set.symmetric_difference_update(other); set ^= other- birinchi to`plamda uchraydigan, lekin ularning ikkala to`plamning kesishmasida uchramaydigan elementlar tashkil topgan to`plam.

Set.add(elem)- to`plamga element qo`shadi.

Set.remove(elem)- to`plamdagi elementni o`chiradi. Agarda ko`rsatilgan element to`plamda mavjud bo`lmasa KeyError ni qaytaradi.

Set.discard(elem)- gar to`plamda ko`rsatilgan element bo`lsa uni o`chiradi.

Set.pop()- to`plamdagi birinchi elementni o`chiradi, lekin top`lam elementlari tartib bilan joylashmagani uchun birinchi element qaysiligini aniq ko`rsatib bo`lmaydi.

Set.clear()- to`plamni tozaydi.

2.3.Modul tushunchasi, standart kutubxonalar, sys va copy moduli.

Standart kutubxona modullarini o`rganishdan oldin Pythonda modul tushunchasiga aniqlik kiritib olish lozim.

Python tilida bir xil vazifani bajaruvchi modullar yig`indisini bitta paketga joylashtirish mumkin. Shunday paketlardan biri sifatida XML paketini misol qilib keltirish mumkin. Ushbu paket XML ning har xil aspektlarini qayta ishlashga mo`ljallangan modullardan tashkil topgan.

Python tilda dastur tuzishda modul atributlari modulda aniqlangan nomlar bo`lgan obyekt modul sifatida taqdim etiladi.

```
>>> import datetime
>>> d1=datetime.date(2017,11,20)
```

Bu misolda datetime moduli import qilinayapti. Import operatorining ishi natijasida mazkur nomlar kengligida datetime nomi bilan obyekt paydo bo`lyapti.

Python tilida modullar oddiy (Pythonda yozilgan) va kengaytiriladigan ya’ni boshqa tilda yoziladigan masalan Python interpretatori yozilgan C dasturlash tilida yozilgan modullarga bo`linadi. Foydalanuvchi nuqtai nazarida ular ishlash tezligi bilan farq qiladi. Satandart kutubxonada modul 2 xil variantda bo`ladi: Pytonda yozigan yoki C da. Bunga misol sifatida *pickle* va *cpickle* modullarini keltirish mumkin. Odatda Pythonda oddiy modullar kengaytirilgan modullarga nisbatan ishlatishda qulay hisoblanadi.

Modul funksiyalaridan foydalanish uchun uni boshqa dasturdan yuklash (импортировать) mumkin. Dastlab standart kutubxonalar modullarini qanday ishlatishni ko’raylik.

```
1  import sys
2
3  print("Buyruqlar qatori argumentlari:")
4
5  for i in sys.argv:
6
7          print(i)
8
9  print("\n\n O'zgaruvchi PythonPATH qiymati", sys.path, "\n")
```

Natija:

```
$ Python3 using_sys.py biz argumentlarmiz
Buyruqlar qatori argumentlari:
using_sys.py
biz
argumentlarmiz
O`zgaruvchi PYTHONPATH qiymati ['/home/user/python darslari/7-dars', '/usr/lib/python3.4', '/usr/lib/python3.4/plat-x86_64-linux-gnu', '/usr/lib/python3.4/lib-dynload', '/usr/local/lib/python3.4/dist-packages', '/usr/lib/python3/dist-packages']
```

Bu misolda dastlab **sys** moduli import buyrug`i yordamida yuklanyapti. **sys** moduli Python interpretatoriga va uning muxitiga ya'ni tizimiga (system) tegishli funksiyalardan tashkil topgan. Python **import sys** buyrug`ini bajarayotganda sys modulini qidiradi. Bu holatda sys standart modullardan biri bo`lganligi uchun, Python uni qayerdan izlash kerakligini biladi. Agar bu oddiy modul, ya'ni Pythonda yozilgan modul bo`lganida edi, u holda Python uni **sys.path** ko`rsatilgan kataloglardan izlagan bo`lar edi. Agar modul topilsa, undagi buyruqlar bajariladi va bu modul foydalanishga (доступным) shay holatga keladi. sys modulidagi argv o'zgaruvchisiga murojat qilish nuqta orqali amalga oshiriladi ya'ni sys.argv. Bunday ifodalashning afzalligi dasturda ishlatilishi mumkin bo`lgan argv o`zgaruvchisi bilan xatoliklar yuz bermaydi. sys.argv qatorlar ro`yxati hisoblanadi. U buyruqlar qatori argumentlaridan ya'ni buyruqlar qatoridan dasturga uzatilgan argumentlardan tashkil topgan.

Python dasturida modullarni ulash *import* operatori orqali amalga oshirilishi yuqoridagi misolda ko`rdik. Modullarni ulashini ham 2 xil shakli mavjud: birinchisi *import* operatori orqali bo`lsa, ikkinchisi *from-import* operatori orqalidir. **From… import … operatori-** argv o`zgaruvchisini dasturga to`g`ridan-to`g`ri yuklash uchun hamda har doim sys.argv deb yozmaslik uchun, from sys import argv ifodasidan foydalanish mumkin. sys modulida ishlatiladigan hamma nomlarni yuklash uchun "from sys import *" buyrug`ini bajarish mumkin.

```
from math import *

n = int(input('Biror son kiriting: '))

print(sqrt(n))
```

Standart kutubxonalar

Python tili standart kutubxonasining modullarini shartli ravishda mavzular bo`yicha quyidagi guruhlarga ajratish mumkin:

1. Bajarish davri servislari. Modular: sys, atexit, copy, traceback, math, cmath, random, time, calendar, datetime, sets, array, struct, intertools, locale, gettext.

2. Siklni qayta ishlashni qo`llab-quvvatlovchi. Modullar: pdb, hotshot, profile, unittest, pydoc. Paketlar: docutils, distutils.

3. OS (fayllar, protseslar) bilan ishlash. Modullar: os, os.path, getopt, glob, popen2, shutil, select, signal, stat, tempfile.

4. Matnlarni qayta ishlovchi. Modullar: string, re, StringIO, codecs, difflib, mmap, sgmllib, htmllib, htmlentitydefs. Paket: xml.

5. Ko`p oqimli hisoblashlar. Modullar: threading, thread, Queque.

6. Ma’lumotlarni saqlash. Arxivlash. Modullar: pickle, shelve, anydbm, gdbm, gzip, zlib, zipfile, bz2, csv, tarfile.

7. Platformaga tobe modullar. UNIX uchun: commands, pwd, grp, fcntl, resource, termios, readline, rlcompleter. Windows uchun: msvcrt, _winreg, winsound.

8. Tarmoqni qo`llab-quvvatlash. Internet protokollari. Modullar: cgi, Cookie, urllib, urlparse, htttplib, smtplib, poplib, telnetlib, socket, asyncore. Serverlarga misollar: SocketServer, BaseHTTPServer, xmlrpclib, asynchat.

9. Internetni qo`llab-quvvatlash. Ma’lumotlar formatlari. Modullar: quopri, uu, base64, binhex, binascii, rfc822, mimetools, MimeWriter, multifile, mailbox. Paket: email.

10. Python uchun. Modullar: parser, symbol, token, keyword, incpect, tokenize, pyclbr, py_compile, compileall, dis, compiler.

11. Grafik interfeys. Modul: Tkinter.

Ko`pincha modullar o`zida bir yoki bir nechta sinflarni saqlaydilar. Bu sinflar yordamida kerakli tipdagi obyekt yaratiladi, lekin gap moduldagi nomlar haqida emas, aksincha shu obtekt atributi haqida boradi. Bir nechta modullar faqat erkin obyetlar ustida ishlash uchun umumiy bo`lgan funksiyalardan iborat bo`ladilar.

Sys moduli

Sys moduli Python interpretatorida dasturni bajaruvchi muhitdir. Quyida bu modulni eng ko`p qo`llaniladigan obyektlari keltilgan:

Exit([c])- dasturdan chiqish. Tugatishning raqamli kodini yuborish mumkin: agarda dasturni tugatish muvafaqqiyatli amalga oshsa 0 ni yuboradi, aksincha bo`lsa ya'ni xatolik yuz bersa boshqa raqamlarni yuboradi.

Argv- buyruqlar qatori argumentlari ro`yxati. Oddiy holatda sys.argv[0] buyruqlar qatoriga ishga tushirilgan dastur nomini va boshqa parametrlar yuboriladi.

Platform- interpretator ishlaydigan platforma.

Stdin, stdout, stderr- standart kiritish, chiqarish, xalolarni chiqarish. Ochiq faylli obyektlar.

Version- interpretator versiyasi.

Serecursionlimit(limit)- rekursiv chaqirishlarni maksimal kiritish darajasini o`rnatadi.

Exc_info()-kiritish-chiqarish istisnosi haqida ma'lumot.

Copy moduli

Bu modul obyektlarni nusxalashga mo`ljallangan funksiyalarga ega. Boshida Pyhtonda sal sarosimaga solish uchun "paradoks" ni ko`rib chiqish tavsiya etiladi.

```
lst1 = [0, 0, 0]
lst = [lst1] * 3
print(lst)
lst[0][1] = 1
print (lst)
```

Va biz kutmagan natija paydo bo`ladi:

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
[[0, 1, 0], [0, 1, 0], [0, 1, 0]]
```

Gap shundaki bu yerda lst ro`yxati shu ro`yxatnining izohiga ega. Agarda rostdan ham ro`yxatni ko`paytirmoqchi bo`lsak, copy modulidagi copy() funksiyasini qo`llash kerak.

```
from copy import copy
lst1 = [0, 0, 0]
lst = [copy(lst1) for i in range(3)]
```

```
print (lst)
lst[0][1] = 1
print (lst)
```

Endi kutilgan natija paydo bo`ladi:

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
[[0, 1, 0], [0, 0, 0], [0, 0, 0]]
```

Copy modulida yuqori aniqalikda nusxalash uchun deepcopy() funksiyasi bor bu funksiya yordamida obektlar butun imkoniyati bilan rekursiv nusxalanadi.

2.4.Math, cmath, random va os moduli.

Math va cmath modullarida haqiqiy va compleksli argumentlar uchun matematik funksiyalar to`plangan. Bu C tilida foydalaniladigan funksiyalar. Quyida math modulining funksiyalari keltirilgan. Qayerda z harfi bilan argumentga belgilash kiritilgan bo`lsa, u cmath modulidagi analogik funksiya ham shunday belgilanishini bildiradi.

Acos(z)-arkkosinus z.

Asin(z)- arksinus z.

Atan(z)- arktangens z.

Atan2(y, x)- atan(y/x).

Ceil(x)- x ga teng yoki katta eng kichik butun son.

Cos(z)- kosinus z.

Cosh(x)- giperbolik x kosinusi.

e- e konstantasi.

Exp(z)- eksponenta (bu degani e**z)

Fabs(x)-x absolute raqami.

Floor(x)- xga teng yoki kichik eng katta butun son

Fmod(x,y)- x ni y ga bo`lgandagi qoldiq qism.

Frexp(x)- mantisa va tartibni (m, i) juftligi kabi qaytaradi, m- o`zgaruvchan nuqtali son, i esa- x=m*2**i ga teng butun son bo`ladi. Agarda 0-(0,0) qaytarsa boshqa paytda 0.5<=abs(m)<1.0 bo`ladi.

Factorial(x)- x ning faktoriali. N!=1*2*3*…*n

Hypot(x,y)- sqrt(x*x+y*y)

Ldexp(m,i)- m*(2**i).

Log(z)- natural logarifm z.

Log10(z)- o`nlik logarifm z.

Log2(z)-logarifm ikki asosga ko`ra z.

Modf(x)- (y,q) juftlikda x ning butun va kasr qismini qaytaradi.

p-pi konstantasi.

Pow(x,y)- x**y.

Sin(z)- z ning sinusi.

Sinh(z)- z ning giperbolik sinusi.

Sqrt(z)- z ning kvadrat ildizi.

Tan(z)- z ning tangensi.

Tanh(z)- z ning giperbolik tangensi.

Trunc(x)- x haqiqiy sonning butun qismini qaytaradi.

degrees(x)-x ni radiandan gradusga o`tkazish.

radians(x)- x ni gradusdan radianga o`tkazish.

```
>>> import math
>>> math.factorial(10)
3628800
>>> math.hypot(3,4)
5.0
>>> math.ceil(10.8)
11
>>> math.ceil(10.2)
11
>>> math.log10(100)
2.0
>>> math.log2(64)
6.0
>>> math.trunc(10.6)
10
```

Random moduli

Bu modul har xil taqsimotlar uchun tasodifiy raqamlarni generatsiya qiladi. Eng ko`p qo`llaniladigan funksiyalari:

Random()-[0.0, 1.0) yarim ochiq diapozondagi tasodifiy sonlarni generatsiya qiladi.

Choice(s)- s ketma- ketlikdan tasodifiy elementni tanlab oladi.

Shuffle(s)- s o`zgaruvchan ketma-ketlik elementlarini joyiga joylashtiradi.

Randrange([start], stop, [step])- renge(start, stop, step) diapozondagi tasodifiy butun raqamni chiqaradi. Choice(range(start, stop, step)) ga analogik holatda.

Normalvariate(mu, sigma)- normal holatda taqsimlangan ketma-ketlikdan raqamni chiqaradi. Bu yerda mu- o`rtacha, sigma-o`rta kvadratli (sigma>0) sonlar.

Boshqa funksiyalar va uning parametrlarini hujjatlashdan aniqlab olish mumkin. Modulda qandaydir holatga tasodifiy raqamlar generatorini joylashtirishga imkon beruvchi seed(n) funksiyasi ham mavjud. Masalan: agarda bitta tasodifiy raqamlar ketma-ketligidan ko`p marta foydalanishga ehtiyoj sezilsa.

Time va Sets moduli

Time moduli joriy vaqtni olish uchun va vaqt formatlarini o`zgartirish uchun fuksiyalarni taqdim etadi.

Sets moduli to`plamlar uchun ko`rsaatgichlar tipini amalga oshiradi. Quyidagi misol bu moduldan qanday foydalanish mumkinligini ko`rsatadi. Su o`rinda bilishimiz kerakki, Python 2.4 va undan yuqori versiyalarda set tipi sets o`rniga kiritilgan.

```
import sets
A = sets.Set([1, 2, 3])
B = sets.Set([2, 3, 4])
print (A | B, A & B, A - B, A ^ B)
for i in A:
    if I in B: print (i,)
```

Natijada:

```
Set([1, 2, 3, 4]) Set([2, 3]) Set([1]) Set([1, 4])
2 3
```

Array va struct modullari

Bu modullar past darajali massiv va korsatgichlar tuzilmasini amalgam oshiradi. Ularning asosiy vazifasi- ko`rsatgichlarning ikkilamchi formatlarini ko`rib chiqish.

Os moduli

Os moduli-har xil operatsion sistemalarning o`ziga xos xususiyatlari bilan ishlovchi kategoriyadagi asosiy modul hisoblanadi. Bu modul funksiyalari ko`plab operatsion sistemalarda ishlaydilar. Kataloglarni bo`luvchi os moduli va u bilan bog`liq bo`lgan ifodalar konstanta ko`rinishida berilgan.

Konstanta	**Vazifasi**
Os.curdir	Joriy katalog

Os.pardir	Bosh katalog
Os.sep	Yo`lning elementlarini taqsimlovchi
Os.altsep	Boshqa yo`lning elementlarini taqsimlovchi
Os.pathsep	Yo`llar ro`yxatidagi yo`llarni taqsimlovchi
Os.defpath	Yashirin yo`llar ro`yxati
Os.linesep	Satrni yakunlovchi belgi

2.4.1-chizma. Kataloglarni bo`luvchi os moduli ifodalari konstanta ko`rinishida

Pythondagi dastur operatsion tizimda alohida jarayon ko`rinishida ishlaydi. Os modulining funksiyalari protsesda, muhitda bajariladigan turli xildagi ahamiyatga ega bo`lgan kirishlarga ruxsat etadilar. Os modulining eng muhim ruxsat etuvchi obyektlaridan biri deb environ o`rab oluvchi muhiti o`zgaruvchilarning lug`ati hisoblanadi. Masalan o`rab oluvchi muhit o`zgaruvchilar yordamida web server CGI-ssenariyga bir qancha parametrlarni o`tkazadi. Quyidagi misolda PATH o`rab oluvchi muhiti o`zgaruvchini olish mumkin:

```
import os
PATH=os.environ['PATH']
```

Funksiyalarning katta qismi fayllar va kataloglar bilan ishlashga mo`ljallangan. Quyida UNIX va Windows OT lar uchun ruxsat etilgan funksiyalar taqdim etilgan:

Access(path, flags)- path nomli fayl yoki catalog ruxsat etish(доступь) ni tekshiradi. Buyurma qilishga rucsatning tartibi flags raqami bilan belgilanadi. U esa yaratilgan kombinatsiyalar os.F_OK (fayl mavjud), os.R_OK (fayldan o`qish mumkin), os.W_OK (faylga yozish mumkin) va os.X_OK (fayllarni bajarishni, katalogni ko`rib chiqish mumkin) bayroqlari bilan belgilash mumkin.

Chdir(path)- path ni joriy ishchi katalog qiladi.

Getcwd()- joriy ishchi catalog.

Chmod(path, mode)- mode ga path bo`lgan ruxsat etish rejimini belgilaydi. Ruxsat etish tartibi bayroqlarni kombinatsiya qilib belgilashi mumkin. Bu ishda chmod()

harakatda bo`lgan tartibni to`ldirmaydi, uni yangidan belgilamaydi, uni yangidan belgilaydi.

Listdir(dir)- dir katalogidagi fayllar ro`yxatini qaytaradi. Ro`yxatga maxsus belgilar “.” va “..” kirmaydi.

Mkdir(path [, mode])- path katalogini tuzadi. Jimlik holatida mode tartibi 0777 ga teng bo`ladi, bu degani S_IRWXU|S_IRWXG|S_IRWXO agarda stat moduli konstantalari bilan foydalansak.

```
import os
os.mkdir('C:\Users\Guljakhon\Desktop\Новая папка\katalog\dir2')
#ko`rsatilgan manzilda dir2 nomli yangi katalog yaratadi.
import os
os.mkdir('./dir2')
#joriy manzilda dir2 nomli yangi catalog yaratadi.
```

Makedirs(path [,mode])- hamma kataloglarni yaratuvchi, agarda ular mavjud bo`lmasalar mkdir() analogi oxirgi katalog mavjud bo`lgandan so`ng mustasnoni ishga tushiradi.

Remove(path), unlink(path)- path katalogini yo`qotadi. Kataloglarni yo`qotish uchun rmdir() va removedirs() dan foydalanadi.

Rmdir(path)- path nomli bo`sh katalogni yo`qotadi.

Removedirs(path)- birinchi bo`sh bo`lgan kataloggacha pathni yo`q qiladi. Agarda yo`lda eng oxirgi kiritilgan katalog osti bo`sh bo`lmasa OSError mustasnosini ishga tushiradi.

Rename(src, dst)- src fayli yoki katallogini dst deb qayta nomlaydi.

Renames(src, dst)- rename() analogi dst yo`li uchun kerakli kataloglarni yaratadi va src yo`lining bo`sh kataloglarini yo`qotadi.

Stat(path)- path haqidagi malumotni o`nta elementlik kortej shaklida qaytaradi. Kortej elementlariga kirish uchun stat moduli konstantalaridan foydalanish mumkin. Masalan stat.ST_MTIME (faylning oxirgi modifikatsiyasi vaqti).

Utime(path, times)- oxirgi modifikatsiya (mtime) va faylga kirishga ruxsat(atime) larini belgilaydi. Boshqa holatlarda times ikki elementli kortej (atime, mtime) sifatida ko`rib chiqiladi. Qaysidir faylni atime va mtime ni olish uchun stat() va stat modulining konstantalarini barobar ishga tushirib olish mumkin.

Os moduli protsesslar bilan ishlash uchun quyidagi funksiyalarni taqdim etadi (ular ham UNIX hamda windowsda ishlaydilar).

System(cmd)- alohida oynada cmd buyruqlar satrini bajaradi. U C tilining system kutubxonasi chqirig`iga analogik bo`ladi. Qaytarilgan qiymat foydalanadigan platformadan tobe bo`ladi.

Times()- beshta elementdan iborat bo`lgan kortejni qaytaradi. U ish jarayoni vaqtini lahzalarda ko`rsatadi, qo`shimcha protsesslar vaqtini, qo`shimcha protsesslarning axborot tizimlari vaqtini, va o`tgan zamonda qotib qolgan vaqtni ko`rsatadi (masalan tizim ishga tushgan paytdan).

Getloadavg()- coo, uchta qiymatlik kortejni qaytaradi.

II bob bo'yicha xulosa.

Bitiruv malakaviy ishining 2-bobi "Pythonda ma'lumotlar tuzilmasi va tilning standart modullari"deb atalib bunda ro`yxat, kortej, lug`at va to`plam tushunchalari va ularni qanday yaratish mumkinligi ular ustida amallar bajarish haqida ma'lumotlar keltirilib misollar yordamida tushuntirilgan. Shuningdek, bu bobda modul tushunchasi, Python dasturining juda boy kutubxonaga ega ekanligi haqidagi ma'lumotlar berilgan bo`lib, ko`plab modullar shu jumladan sys, copy, time, math, cmath, random, os modullari ularning funksiyalari va qo`llanilishi haqida yozilib, misollar keltirish yordamida tushuntirilgan.

Ushbu bobda talabalar o'z ustlarida mustaqil ishlashlari va Python dasturida ishlash bo`yicha bilim saviyalarini oshirish uchun standart modullardan foydalanishlari taklif etilgan.

XOTIMA

Mazkur metodik qo`llanmada Python dasturlash tilining yaratilish tarixi, imkoniyatlari va Python dasturlash tilini Windows operatsion tizimida o`rnatish haqida yozilgan. Python dasturida ishlaydigan foydalanuvchilar uchun uning sintaksisi, asosiy operatorlari, fayllar, funksiyalar bilan ishlash sanoq sistemalari va satrlar bilan ishlash haqida muhim zaruriy ma'lumotlar keltirilgan. Qo`llanmani o`qigan har bir qiziquvchi Python dasturini o`rnatish boshqa odatiy dasturlarni o`rnatish kabi hech qanday qiyinchiliksiz o`rnatilishini, tilning sintaksisi o`zi kabi

sodda va oson ekanligini, o`zgaruvchilarning tipini e’lon qilinmasligini, shuningdek sonlar bilan ishlaganda nafaqat butun va haqiqiy sonlar ustida balki kompleks sonlar ustida ham amallar bajarishni ko`rsatilgan misollar yordamida o`rgana oladi va uni amaliyotda bajara oladi. Satrlar bilan ishlash va ular ustida amallar bajarish haqida ham yetarlicha ma’lumotlar keltirilgan.

Ushbuda yana ro`yxat, kortej, lug`at va to`plam tushunchalari va ularni qanday yaratish mumkinligi ular ustida amallar bajarish haqida ma’lumotlar keltirilib misollar yordamida tushuntirilgan. Shuningdek, modul tushunchasi, Python dasturining juda boy kutubxonaga ega ekanligi haqidagi ma’lumotlar berilgan bo`lib, ko`plab modullar shu jumladan sys, copy, time, math, cmath, random, os modullari ularning funksiyalari va qo`llanilishi haqida yozilib, misollar keltirish yordamida amaliyotda qo`llab tushuntirilgan. O`quvchilar o’z ustilarida mustaqil ishlashlari va Python dasturida ishlash bo`yicha bilim saviyalarini oshirish uchun standart modullardan foydalanishlari taklif etilgan. Python dasturlash tilining afzallik tomonlari tushuntirib berildi va shular asosida o’zbek tilida uslubiy qo’llanma yaratildi.