H.SH.Rustamov

# ALGORITMIK TILLAR VA DASTURLASH

O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA MAXSUS
TA’LIM VAZIRLIGI
BUXORO DAVLAT UNIVERSITETI

**Rustamov Hakim Sharipovich**

# ALGORITMIK TILLAR VA DASTURLASH

*60540200 – Amaliy matematika ta’lim yo’nalishi talabalari uchun o’quv qo’llanma*

**“Durdona” nashriyoti**
**Buxoro – 2022**

**UO'K 004.421(075.8)**
**32.973-018ya73**
**R 91**
Rustamov, Hakim Sharipovich
Algoritmik tillar va dasturlash [Matn] : o'quv qo'llanma / H.Sh. Rustamov .- Buxoro: "Sadriddin Salim Buxoriy" Durdona, 2022.-237 b.

**KBK 32.973-018ya73**

O'quv qo'llanmada oliy ta'lim muassasalarining 60540200-amaliy matematika ta'lim yo'nalishi talabalari uchun algoritmik tillar va dasturlash fanining fan dasturidagi mavzulari asosida bayon qilingan. Mazkur o'quv qo'llanmadan kompyuter ilmi talabalari hamda yuqori sinf o'qituvchilari ham foydalanishlari mumkin.

**Tuzuvchi**: Rustamov Hakim Sharipovich

**Taqrizchilar:**
**Sh. S. Yo'ldoshev** - Buxoro muhandislik texnologiyalari instituti, axborot kommunikatsiya texnologiyalari kafedrasi dotsenti
**O. I. Jalolov** – Buxoro davlat universiteti, amaliy matematika va dasturlash texnologiyalari kafedrasi dotsenti

**O'quv qo'llanma Oliy va o`rta maxsus vazirligining 2022-yil 2-avgustdagi 257-sonli buyrug'iga asosan nashr etishga ruxsat berilgan. Ro'yxatga olish raqami 257-002.**

**ISBN 978-9943-8588-6-2**



# MUNDARIJA

1-mavzu: Dasturlash tillarining rivojlanish tarixi. Python dasturlash tili ..... 5
2-mavzu: Identifikatorlar, o'zgaruvchilar va berilganlar turlari ..... 25
3-mavzu: Python dasturlash tilida arifmetik va mantiqiy amallar ..... 31
4-mavzu: Pyton dasturlash tilida shartli o'tish operatori ..... 39
5-mavzu: Pyton dasturlash tilida For... takrorlash operatori ..... 44
6-mavzu: Python dasturlash tilida While ...takrorlash operatori ..... 48
7-mavzu: Sonlar bilan ishlash funksiyalari ..... 51
8-mavzu: Satrlar bilan ishlash funksiyalari ..... 59
9-mavzu: Satrdagi ma'lumotlarni izlash ..... 62
10-mavzu: Regulyar ifodalar ..... 65
11-mavzu: Ro'yxatlar bilan ishlash ..... 70
12-mavzu: Ro'yxat funksiyalari bilan ishlash ..... 75
13-mavzu: Ro'yxat elementlarini tasodifiy tanlash ..... 78
14-mavzu: Kortej va to'plamlar bilan ishlash ..... 84
15-mavzu: Lug'at yaratish va ular bilan ishlash ..... 88
16-mavzu: Sana va vaqt bilan ishlash ..... 98
17-mavzu: Kalendar moduli va uning imkoniyatlari ..... 102
18-mavzu: Funksiya va uning imkoniyatlari ..... 106
19-mavzu: Generator funksiyalari ..... 108
20-mavzu: Modullar ..... 111
21-mavzu: Ob'ektga yo'naltirilgan dasturlash ..... 118
22-mavzu: Sinf va uning xususiyatlari ..... 121
23-mavzu:Istisnolarni qayta ishlash ..... 128
24-mavzu:To'plamalar va uning metodlari ..... 134
25-mavzu: Fayllar va kataloglar bilan ishlash ..... 140
26-mavzu: Kataloglar bilan ishlash funksiyalari ..... 143
27-mavzu: SQLite asoslari ..... 150
28-mavzu: Where va Having ko'rsatmalaridagi shartlar ..... 155

**Rustamov Hakim Sharipovich**

# ALGORITMIK TILLAR VA DASTURLASH

*60540200 – Amaliy matematika ta'lim yo'nalishi talabalari uchun o'quv qo'llanma*

*Muharrir:* *A. Qalandarov*
*Texnik muharrir:* *G. Samiyeva*
*Musahhih:* *Sh. Qahhorov*
*Sahifalovchi:* *M. Bafoyeva*

Nashriyot litsenziyasi AI № 178. 08.12.2010. Original-maketdan bosishga ruxsat etildi: 22.09.2022. Bichimi 60x84. Kegli 16 shponli. «Times New Roman» garn. Ofset bosma usulida bosildi. Ofset bosma qog`ozi. Bosma tobog`i 14,8. Adadi 100. Buyurtma №486.

"Sadriddin Salim Buxoriy" MCHJ
"Durdona" nashriyoti: Buxoro shahri Muhammad Iqbol ko`chasi, 11-uy.
Bahosi kelishilgan narxda.

"Sadriddin Salim Buxoriy" MCHJ bosmaxonasida chop etildi.
Buxoro shahri Muhammad Iqbol ko`chasi, 11-uy. Tel.: 0(365) 221-26-45

ISBN 978-9943-8588-6-2

**O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI**

**BUXORO DAVLAT UNIVERSITETI**

**Rustamov Hakim Sharipovich**

# ALGORITMIK TILLAR VA DASTURLASH

*60540200 – Amaliy matematika ta’lim yo’nalishi talabalari uchun o’quv qo’llanma*

O‘quv qo‘llanmada oliy ta’lim muassasalarining 60540200-amaliy matematika ta’lim yo’nalishi talabalari uchun algoritmik tillar va dasturlash fanining fan dasturidagi mavzulari asosida bayon qilingan. Mazkur o‘quv qo‘llanmadan kompyuter ilmi talabalari va yuqori sinf o‘qituvchilari ham foydalanishlari ga mo‘ljallangan.

**Tuzuvchi**: Rustamov Hakim Sharipovich

**Taqrizchilar:**

**Sh. S. Yo’ldoshev** - Buxoro muhandislik texnologiyalari instituti, axborot kommunikatsiya texnologiyalari kafedrasi dotsenti

**O. I. Jalolov** – Buxoro davlat universiteti, amaliy matematika va dasturlash texnologiyalari kafedrasi dotsenti

Buxoro davlat universiteti Kengashining 2021-yil 1-noyabrdagi 3-sonli bayonnomasi asosida nashrga tavsiya etilgan

MUNDARIJA

1-mavzu: Dasturlash tillarining rivojlanish tarixi. Python dasturlash tili ..... 7
2-mavzu: Identifikatorlar, literallar, o’zgaruvchilar va berilganlar turlari. ..... 26
3-mavzu: Python dasturlash tilida arifmetik va mantiqiy amallar ..... 31
4-mavzu:Pyton dasturlash tilida shartli o’tish operatori ..... 38
5-mavzu:Pyton dasturlash tilida For… takrorlash operatori ..... 43
6-mavzu: Python dasturlash tilida While …takrorlash operatori ..... 47
7-mavzu: Sonlar bilan ishlash funksiyalari ..... 49
8-mavzu: Satrlar bilan ishlash funksiyalari ..... 51
9-mavzu: Satrdagi ma’lumotlarni izlash ..... 56
10-mavzu: Regulyar ifodalar ..... 59
11-mavzu: Ro'yxatlar bilan ishlash ..... 65
12-mavzu: Ro'yxat funksiyalari bilan ishlash ..... 70
13-mavzu: Ro'yxat elementlarini tasodifiy tanlash ..... 74
14-mavzu: Kortej va to’plamlar bilan ishlash ..... 81
15-mavzu: Lug’at yaratish va ular bilan ishlash ..... 86
16-mavzu: Sana va vaqt bilan ishlash ..... 98
17-mavzu: Kalendar moduli va uning imkoniyatlari ..... 103
18-mavzu: Funksiya va uning imkoniyatlari ..... 108
19-mavzu: Generator funksiyalari ..... 111
20-mavzu: Modullar ..... 114
21-mavzu: Ob'ektga yo'naltirilgan dasturlash ..... 122
22-mavzu: Sinf va uning xususiyatlari ..... 127
23-mavzu:Istisnolarni qayta ishlash ..... 134
24-mavzu:To’plamalar va uning metodlari ..... 141
25-mavzu: Fayllar va kataloglar bilan ishlash ..... 149
26-mavzu: Kataloglar bilan ishlash funksiyalari ..... 153
27-mavzu: SQLite asoslari ..... 159
28-mavzu: Where va Having ko’rsatmalaridagi shartlar ..... 164
29-mavzu: Pythondan SQLite ma'lumotlar bazalariga kirish ..... 167

30-mavzu:Agregat funksiyalar ..... 171
31-mavzu: Python dasturlash tilidan MySQL ma'lumotlar bazalariga kirish... 174
32-mavzu: Python dasturlash tilida grafika bilan ishlash ..... 175
33-mavzu: Internet - dasturlash..... 186
34-mavzu: Tkinter kutubxonasi Oyna ilovalarini ishlab chiqish asoslari ..... 187
35-mavzu: Komponentlardan foydalanish ..... 194
36-mavzu: Tkinter kutubxonasi. Frame, Button va Entry komponentlari..... 197
37-mavzu: Label, Checkbutton, Radiobutton va *Combobox* komponenti komponentlari ..... 212
38-mavzu: Notebook, Progressbar, Sizegrip, va Treeview komponentlari ..... 226
39-mavzu: Listbox, Spinbox va PanedWindow komponentlari ..... 231
40-mavzu: Menu komponenti va uning imkoniyatlari ..... 234
41-mavzu: Standart dialog oynalar bilan ishlash ..... 238
42-mavzu: Parallel dasturlash ..... 242
43-mavzu: Ko’p bosqichli dasturlash ..... 246
44-mavzu: Utilit funksiyalar ..... 249
45-mavzu: Python dasturlash tilida arxivlar bilan ishlash ..... 252
Foydalanilgan adabiyotlar..... 254

СОДЕРЖАНИЕ

Тема 1: История развития языков программирования. Язык программирования Python…….... 5
Тема 2: Идентификаторы, литералы, переменные и типы данных…….. 24
Тема 3: Арифметические и логические операции в языке программирования Python…….. 30
Тема 4: Оператор условного перехода в языке программирования Python ……37
Тема 5: Оператор итерации For… в языке программирования Python ……42
Тема 6: В то время как в языке программирования Python… оператор итерации……. 46
Тема 7: Функции работы с числами ……48
Тема 8: Функции гребли……50
Тема 9: Поиск информации в строке……...55
Тема 10: Регулярные выражения……..58
Тема 11: Работа со списками……....64
Тема 12: Работа со списками функций ……...69
Тема 13: Случайный выбор элементов списка……73
Тема 14: Работа с кортежами и множествами……80
Тема 15: Создание и работа со словарями……...86
Тема 16: Работа с датой и временем ..……97
Тема 17: Модуль календаря и его возможности ……..102
Тема 18: Функция и ее возможности……108
Тема 19: Функции генератора……111
Тема 20: Модули……..114
Тема 21: Объектно-ориентированное программирование…….……..121
Тема 22: Класс и его свойства……126
Тема 23: Обработка исключений……134
Тема 24: Коллекции и их методы……...141
Тема 25: Работа с файлами и каталогами ……..……149
Тема 26: Функции работы с каталогами ……..……..……153
Тема 27: Основы SQLite…… 159
Тема 28: Термины в инструкциях «Где» и «Наличие» ……...164
Тема 29: Доступ к базам данных SQLite из Python ……167
Тема 30: Агрегатные функции ……...171
Тема 31: Доступ к базам данных MySQL из языка программирования Python ……174
Тема 32: Работа с графикой в языке программирования Python ……175
Тема 33: Интернет - программирование …….……..186
Тема 34: Основы разработки оконных приложений с помощью библиотеки Tkinter …….187
Тема 35: Использование компонентов …….……194
Тема 36: Библиотека Tkinter. Рамка, кнопка и элементы входа …….……197

Тема 37: Компоненты Label, Checkbutton, Radiobutton и Combobox ……212
Тема 38. Компоненты Notebook, Progressbar, Sizegrip и Treeview ……226
Тема 39: Компоненты Listbox, Spinbox и PanedWindow ……231
Тема 40: Компонент меню и его опции ……234
Тема 41: Работа со стандартными диалогами ……238
Тема 42: Параллельное программирование ……242
Тема 43: Многоуровневое программирование…… 245
Тема 44: Вспомогательные функции ……249
Тема 45: Работа с архивами на языке программирования Python…… 252

CONTENTS

Topic 1: History of the development of programming languages. Python programming language ……5
Topic 2: Identifiers, literals, variables, and data types. ……..24
Topic 3: Arithmetic and logic operations in the Python programming language ……...30
Topic 4: Conditional transition operator in Python programming language ……..37
Topic 5: For… iteration operator in Python programming language ……42
Topic 6: While in the Python programming language… iteration operator ……...46
Topic 7: Functions of working with numbers…… 48
Topic 8: Rowing Functions…… 50
Topic 9: Searching for information on line ……55
Topic 10: Regular Expressions ……...58
Topic 11: Working with lists ……..64
Topic 12: Working with list functions ……69
Topic 13: Random selection of list items ……...73
Topic 14: Working with tuples and sets ……80
Topic 15: Creating and working with dictionaries ……86
Topic 16: Working with Date and Time ……97
Topic 17: Calendar module and its capabilities ……102
Topic 18: Function and its capabilities ……108
Topic 19: Generator Functions ……111
Topic 20: Modules ……114
Topic 21: Object-oriented programming…….. 121
Topic 22: Class and its properties ……126
Topic 23: Processing Exceptions ……..134
Topic 24: Collections and their methods ……..141
Topic 25: Working with files and directories ……...149
Topic 26: Functions of working with catalogs……..153
Topic 27: Basics of SQLite ……..159
Topic 28: Terms in the Where and Having instructions ……..164
Topic 29: Access to SQLite databases from Python ……167
Topic 30: Aggregate Functions ……171
Topic 31: Access to MySQL databases from the Python programming language……174
Topic 32: Working with graphics in the Python programming language ……175
Topic 33: Internet - programming ……186
Topic 34: Tkinter Library Basics of Window Application Development……187
Topic 35: Use of components……194
Topic 36: Tkinter Library. Frame, Button and Entry Components……...197

Topic 37: Label, Checkbutton, Radiobutton and Combobox Component Components ……..212
Topic 38: Notebook, Progressbar, Sizegrip, and Treeview Components…… 226
Topic 39: Listbox, Spinbox and PanedWindow Components ……..231
Topic 40: Menu component and its options ……..234
Topic 41: Working with standard dialogs ……238
Topic 42: Parallel Programming ……...242
Topic 43: Multilevel Programming…….. 245
Topic 44: Utility Functions ……...249
Topic 45: Working with archives in the Python programming language…… 252

## 1-mavzu: Dasturlash tillarining rivojlanish tarixi. Python dasturlash tili

### Mavzu rejasi:

1.1. Dasturlash tillarining rivojlanish tarixi
1.2. Python dasturlash tilining imkoniyatlari va uni o’rnatish
1.3. Python dasturlash tilida birinchi dastur
1.4. Pythonda dastur tuzilishi
1.5. pip utility yordamida dasturi: qo'shimcha kutubxonalarni o'rnatish

### 1.1. Dasturlash tillarining rivojlanish tarixi

**Dastlabki yuqori darajadagi dasturlash tillari.** XX asrning 50-yillarning o'rtalaridan boshlab, birinchi yuqori darajali dasturlash tillarini (yuqori darajali dasturlash tillari) yaratishga kirishildi. Ushbu dasturlash tillari ma'lum bir kompyuter turiga (mashinadan mustaqil) bog'lanmagan bo’lib, har bir dasturlash tili o’zining translyatoriga ega bo’ladi. Translayator (tarjimon) o’z navbatida **kompilyator** va **interpretator** turlarga bo’linadi.

Kompilyator – yuqori darajadagi dasturlash tilida yozilgan dasturni to’laligicha mashina tiliga tarjima qiladi. Kompilyatorga ega bo’lgan dasturlash tillariga Pascal, Assembler, C, C++ va boshqa dasturlash tillarilari kiradi.

Interpretator – yuqori dasturlash tilida yozilgan dasturni satrma-satr o’qib, mashina tiliga tarjima qiladi. Interpretatorga ega bo’lgan dasturlash tillariga BASIC, Python, Ruby, JavaScript, PHP va boshqa dasturlash tillari misol bo’ladi.

Interpretatorga ega bo’lgan dasturlash tillari kompilyatorga nisbatan tarjima qilishni sekinroq amalga oshiradi. Ya’ni, interpretator orqali tarjima qiluvchi dasturlash tili kompilyatorga ega bo’lgan dasturlash tiliga nisbatan sekinroq ishlaydi.

Hozirgi kunda jamiyat taraqqiyotining barcha sohalarida ko’plab zamonaviy dasturlash tillaridan keng foydalanilmoqa.

**Fortran dasturlash tili.** Dastlabki yuqori darajadagi dasturlash tillaridan biri Fortran dasturlash tili bo’lib, mazkur dasturlash tili 1954-1957 yillarda IBM korporatsiyasi xodimi Jon Backus boshchiligidagi dasturchilar guruhi tomonidan yaratilgan. Fortran dasturlash tili asosan ilmiy va texnik hisob-kitoblarni bajarish uchun mo'ljallangan dasturlash tili hisoblanadi. Fortran nomi **FOR**mula **TRAN**slator (Formula Translator) ning qisqartmasi ya’ni “Formula tarjimoni” sifatida e’tirof etiladi.

1953 yil oxirida Jon Backus IBM 704 kompyuterida dasturlash uchun assemblerga samarali alternativani ishlab chiqishni boshlashni taklif qildi.1954 yil o'rtalarida Fortran tilining spetsifikatsiyasi loyihasi tugallandi. Fortran uchun birinchi qo‘llanma 1957-yil aprel oyida yaratilgan birinchi kompilyator bilan birga 1956-yil oktabr oyida paydo bo‘ldi. Kompilyator optimallashtirdi, chunki mijozlar assemblernikidan past unumdorlikka ega kod ishlab chiqaruvchi yuqori darajadagi dasturlash tilidan foydalanishni rad etishdi.

O'sha paytda jamoa dasturlashning yangi usuliga shubha bilan qaragan va Fortran dasturlashni tezroq va samaraliroq qilishiga ishonmagan. Jon Backusning so'zlariga ko'ra, uning ko'p ishlari "dangasa bo'lishga" qaratilgan. U assemblerda IBM 701 uchun dasturlar yozishni juda yoqtirmasdi. Fortran dasturlash tili olimlar tomonidan hisoblash intensiv dasturlarini yozish uchun keng qo'llanilgan. Murakkab ma'lumotlar turini kiritish uni ayniqsa texnik ilovalar uchun moslashtirdi.

1960 yilga kelib IBM 709, 650, 1620, 7090 kompyuterlari uchun Fortran versiyalari paydo bo'ldi. Uning katta mashhurligi raqobatchi kompyuter ishlab chiqaruvchilarini o'z kompyuterlari uchun Fortran kompilyatorlarini yaratishga undadi. Shunday qilib, 1963 yilga kelib turli platformalar uchun 40 dan ortiq

kompilyatorlar mavjud edi. Shuning uchun ham Fortran birinchi keng tarqalgan dasturlash tili hisoblanadi. Fortran dasturlash tili sobiq sovet ittifoqida G'arbga qaraganda kechroq paydo bo'ldi, chunki dastlab sobiq sovet ittifoqida Algol dasturlash tili yanada istiqbolli dasturlash tili hisoblangan. Sobiq Sovet ittifoqida fiziklarining 1960-yillarda Fortran dasturlash tilida yaratilgan dasturlar yordamida deyarli barcha hisob-kitoblar amalga oshirilgan. Jon Backusning CERN(Conseil Européen pour la Recherche Nucléaire)dagi hamkasblari bilan aloqasi Fortranni joriy etishda muhim rol o'ynadi.

Dastlab sobiq Sovet ittifoqida Fortran kompilyatori 1967 yilda Minsk2 mashinasi uchun yaratilgan, ammo u unchalik mashhur bo'lmagan. Fortranni keng joriy etish 1968 yilda BESM-6 mashinasi uchun FORTRAN-DUBNA kompilyatori yaratilgandan keyin mashhurligi boshlangan. 1972 yilda paydo bo'lgan ES kompyuterlarida allaqachon Fortran dasturlash tilining tarjimoni mavjud edi.

Zamonaviy Fortran dasturlash tilida asosan ilmiy va muhandislik hisob-kitoblari uchun keng qo'llanilgan. Bu raqamli muammolarni hal qilish uchun juda yaxshi, chunki uning mavjudligi davomida ko'plab kutubxonalar yozilgan. U bugungi kungacha qo'llanilgan, lekin muvaffaqiyatli dizayn tufayli emas, balki juda ko'p dastur yozilganligi sababli, uni o'zgartirish va bundan tashqari, qayta yozish mantiqiy emas. Uning tuzilishi kompilyator hisob-kitoblarni juda yaxshi optimallashtirishga yordam beradi.

Olimlar orasida har qanday matematik muammoning Fortranda yechimi bor, degan gap bor va haqiqatan ham minglab Fortran paketlari orasidan matritsalarni ko'paytirish paketi va murakkab integral tenglamalarni yechish paketini va boshqa ko'plab narsalarni topish mumkin bo’lgan. Hozirgi paytda ham mazkur dasturlash tilining zamonaviy versiyalari ishlab chiqilgan.

**ALGOL dasturlash tili.** Algol (**ALGO**rithmic **L**anguage) - dasturlash tili 1958 yilda ETHda (Syurix, Shveytsariya) bir haftalik konferentsiyada keng ko'lamli ilovalar uchun umumiy maqsadli dasturlash tili sifatida ishlab chiqilgan va Xalqaro

axborotni qayta ishlash federatsiyasi tomonidan tashkil etilgan qo'mita tomonidan yanada takomillashtirilgan.

Dastlab, taklif qilingan ALGOL (ALGOritmik til) nomi rad etildi. Ammo bu odatiy holga kelganligi sababli, IALning rasmiy nomi keyinchalik ALGOL 58 ga o'zgartirilishi kerak edi.

Shunday qilib, o'sha paytda mavjud bo'lgan kompyuterlarning hech biri Algol alifbosini tashkil etgan 116 ta harfni kiritish-chiqarishni qo'llab-quvvatlamasligi aniqlandi. Ammo Evropada Algol dasturlash tili ishtiyoq bilan qabul qilindi. U akademik muhitda tezda mashhur bo'ldi, kompilyatorlar hamma joyda ishlab chiqildi, ularning aksariyati amalga oshirishdagi qiyinchiliklarga qaramay, juda muvaffaqiyatli bo'ldi. Algol Buyuk Britaniyadan sobiq ittifoqning Uzoq Sharqiga tarqalib, ilmiy nashrlarda algoritmlarni tavsiflash uchun universal dasturlash tili sifatida haqiqiy dasturlash vositasiga aylandi.

Algolda dastur g'oyasi buyruqlarning erkin ketma-ketligi sifatida emas, balki aniq tasvirlangan va ajratilgan qismlardan iborat blokli tuzilma sifatida paydo bo'ldi. Algol dasturlash tilidagi dasturning asosiy bloki asosiy dasturning o'zi. U blokga o'ralgan, bosh va tugatish kalit so'zlari juftligi bilan ajratilgan bajariladigan qismini, shuningdek, quyi dasturlarning tavsiflarini o'z ichiga oladi. Har bir dastur kichik dasturlarga ajratilgan bo'lib, uning ichida o'ziga xos ma'lumotlar tasvirlangan, nom va rasmiy parametrlar ro'yxati ko'rinishidagi yagona aniqlangan interfeys va kod bloki mavjud.

Bunday holda, blokda qism bloklarni ajratish imkoniyati mavjud mumkin. Strukturaviy boshqaruv tuzilmalari aniqlandi: qatorlar, halqalar, ketma-ket bo'limlar, ular shartli yoki ko'paytiriladigan so'zlar to'plamini bajaradi, shuningdek, bir xil bosh va tugatish kalit so'zlari bilan cheklangan.

Zamonaviy dasturchilar uchun bunday dastur tuzilmasi aniq, biroz eskirgan va har doim ham qulay bo'lmagan ko'rinadi, ammo Algol dasturlash tili paydo bo'lgan paytda bularning barchasi oldinga sezilarli qadam bo'lib, bu esa ularning hajmini oshirishga imkon berdi, ularni ko'rinadigan, tushunarli, tahlil qilish va tuzatish

uchun ochiq. Aynan Algol dasturlash tili va uning yangi avlod tillari asosida dasturlarning to'g'riligini tahliliy isbotlash bo'yicha muvaffaqiyatli ish olib borildi.

Algolda parametrlarni pastki dasturga o'tkazishning ikkita usuli taklif qilingan - nomi va qiymati bo'yicha. Agar ikkinchi usul hech qanday e'tiroz bildirmasa (u hozirgi kunga qadar ko'pchilik tillarda keng qo'llaniladi), unda birinchisi (haqiqiy parametr nomi protseduraga uzatiladi deb taxmin qilinadi va protsedura quyidagicha ishlaydi). agar uning kodi mos yozuvlar nuqtasida yozilgan bo'lsa, bu erda rasmiy parametr o'rniga haqiqiyning nomi yoziladi) kompilyatorlarni amalga oshirishda qiyinchiliklarga va aniqlash qiyin bo'lgan xatolar paydo bo'lishiga olib keldi.

**Lisp dasturlash tili.** Lisp dasturlash tili 1960 yilda J. Makkarti tomonidan taklif qilingan bo‘lib, u sonli bo‘lmagan masalalarni yechish dasturlarini ishlab chiqishga qaratilgan. Ushbu tilning inglizcha nomi - LISP **LIS**t **P**rocessing (ro'yxatni qayta ishlash) iborasining qisqartmasi bo'lib, uning qo'llanilishining asosiy sohasini yaxshi ta'kidlaydi. Ro‘yxatlar ko‘rinishida algebraik ifodalar, grafiklar, chekli guruhlar elementlari, to‘plamlar, xulosa chiqarish qoidalari va boshqa ko‘plab murakkab ob’yektlarni ko‘rsatish qulay. Ro'yxatlar kompyuter xotirasida axborotni ko'rsatishning eng moslashuvchan shaklidir. Ro'yxatlarni qayta ishlash uchun maxsus mo'ljallangan qulay dasturlash tili tezda amalga oshirilsa hech qanday ajablanarli emas.

O'zining deyarli qirq yillik faoliyati davomida ushbu dasturlash tilining bir qator dialektlari paydo bo'ldi. Masalan: Common LISP, Mac LISP, Inter LISP, Standard LISP va boshqalar. Ularning orasidagi farqlar fundamental xususiyatga ega emas bo’lib, asosan bir oz boshqacha o'rnatilgan funktsiyalar to'plamiga va ro'yxatga olish dasturlari ko'rinishidagi ba'zi farqlarga to'g'ri keladi. Shunday ekan, yuqoridagi dasturlash tillaridan biri ustida ishlashni o‘rgangan dasturchi boshqasini bemalol o‘zlashtira oladi.

Lisp dasturlash tilining katta afzalligi uning funksional yo'nalganligidir, ya'ni dasturlash funksiyalar yordamida amalga oshiriladi. Bundan tashqari, funktsiya deganda ma'lum bir sinfning elementlarini boshqa sinfning tegishli elementlari bilan mos keladigan qoida tushuniladi. Taqqoslash jarayonining o'zi dasturning ishlashiga

hech qanday ta'sir ko'rsatmaydi, faqat uning natijasi muhim - funktsiyaning qiymati. Bu katta dasturiy ta'minot tizimlarini yozish nisbatan osonlashtiradi va dasturning yozilishining oddiyligi bilan ajralib turadi. Dasturlarning ravshanligi, ularning funktsiyalarining aniq chegaralanganligi, ularni bajarishda murakkab noqulay ta'sirlarning yo'qligi sun'iy intellekt vazifalari kabi mantiqiy jihatdan murakkab vazifalarni mavjudligi dasturlash uchun majburiy talablardir.

**COBOL dasturlash tili.** (**CO**mmon **B**usiness **O**riented **L**anguage) - Cobol dasturlash tili 1959 yilda ishlab chiqilgan va asosan biznes ilovalarini ishlab chiqish uchun dasturlar yozish, shuningdek, iqtisodiy sohada ishlashga mo'ljallangan dasturlash tili hisoblanadi.

COBOL dasturlash tilining yaratuvchilari COBOL dasturlash tilini mashina tilidan mustaqil ravishda hamda tabiiy ingliz tiliga iloji boricha yaqinroq qilishni maqsad qilib qo'ygan. Ikkala maqsad ham muvaffaqiyatli amalga oshirilgan; COBOL dasturlari hatto mutaxassis bo'lmaganlar uchun ham tushunarli hisoblanadi, chunki ushbu dasturlash tilidagi matnlar hech qanday maxsus sharhlarga muhtoj emas (o'z-o'zini hujjatlashtiruvchi dasturlar) hisoblanadi.

COBOL dasturlash tili juda ham qadimgi dasturlash tili bo'lib, bir vaqtning o'zida juda faol ishlatilgan, shuning uchun ko'plab ilovalar va dialektlar mavjud. COBOL dasturlash tili uchun bir qator standartlar tasdiqlangan. Masalan: 1968, 1974, 1985 va 2002 yillarda va ularning eng so'nggi standarti ob'ektga yo'naltirilgan dasturlash tilini qo'llab-quvvatladi.

COBOL dasturlash tili sizga katta hajmdagi ma'lumotlar bilan samarali ishlash imkonini beradi, u turli xil qidirish, saralash va tarqatish imkoniyatlariga to'la. COBOL dasturlash tilining boshqa afzalliklari odatda uning tuzilganligini o'z ichiga oladi. Shaxsiy kompyuterlar uchun ushbu dasturlash tilidan foydalanib juda kuchli kompilyatorlar ishlab chiqilgan. Ulardan ba'zilari shunchalik samaraliki, shaxsiy kompyuterda tuzatilgan dasturni asosiy kompyuterlarga osongina o'tkazish mumkin.

COBOL dasturlash tilining kamchiliklaridan biri, COBOL dasturlash tilida faqat eng oddiy algebraik hisob-kitoblarni dasturlash mumkinligini eslatishimiz mumkin, murakkab muhandislik hisoblari uchun bu til mos kelmaydi.

**BASIC** (**Beginners' All-purpose Symbolic Instruction Code)** — “Boshlovchilar uchun ko’p maqsadli belgilar ko’rsatmalar kodi“ — dasturlashga mutaxassis bo‘lmagan foydalanuvchilar (muhandislar, olimlar, talabalar) uchun dasturlash tili, 1964-yilda AQShning Dartmound College xodimlari John Kemeny va Thomas Kurtz tomonidan ishlab chiqilgan.

BASIC dasturlash tilining eng gullab-yashnagan va rivojlanishi davri 1970-yillarning oxiri va 1980-yillarning birinchi yarmi deb hisoblash mumkin. Ushbu davrda deyarli barcha shaxsiy kompyuterlar o'zlarining BASIC interpretatoriga ega bo'lib, ular eng oddiy kompyuterlarda ham ko'pincha operatsion tizim funktsiyalarini bajarganlar. Deyarli barcha kuchli kompyuterlarda ham interaktiv BASIC tizimlari mavjud edi. *BASIC kompilyatorlari* deyarli yo'q edi, ilovalarning aksariyati chiziq muharriri va interpretatorning standart gibrididir. Vaqt o'tishi bilan tilning asosiy vositalari takomillashtirildi, bu esa ba'zi bir amalga oshirishda murakkab tarmoq operatorlari, qo'shimcha turdagi tsikllar va parametrli nomli protseduralarning paydo bo'lishiga olib keldi.

IBM PC platformasi uchun BASIC dasturlash tilining bir qancha yangi versiyalari yaratilgan. Microsoft MS-DOS / PC DOS uchun BASIC ni sotdi, jumladan IBM Advanced BASIC (BASICA), GW-BASIC (IBMdan "proshivka" ni talab qilmaydigan BASICA modifikatsiyasi) va QuickBASIC. Bozorga dastlab taniqli Turbo Paskal bilan kirgan Borland 1985 yilda xuddi shu dasturiy muhitga asoslangan Turbo Basic 1.0 tizimini chiqardi (keyinchalik uning vorislari PowerBASIC nomi bilan boshqa kompaniya tomonidan sotilgan). Ba'zi boshqa dasturlash tillar butunlay boshqa tizim qurilgan asos sifatida taniqli BASIC sintaksisidan foydalangan. BASIC dasturlari samaradorligini oshirish istagi to'liq kompilyatorlarning (masalan, yuqorida aytib o'tilgan Turbo Basic kabi), shuningdek, "gibrid" tizimlarning paydo bo'lishiga olib keldi, ularda talqin qilish asosiy usul sifatida saqlanib qoldi. dasturni bajarish, qisman kompilyatsiya

"uchishda" amalga oshirildi, masalan, qisqa davrlarni kompilyatsiya qilish. Bundan tashqari, hech bo'lmaganda IBM platformasida interpretator BASIC dasturini psevdokodga (ko'pincha qaytariladigan) tarjima qilganda va interpretator ob'ekt modulidan va dastur matni bilan kodlangan moduldan bajariladigan dasturni shakllantirganda, BASIC dasturlarining psevdokompilyatsiyasi tarqaldi. Bunday dastur oddiy kompilyatsiya qilingan ob'yekt dasturi kabi ishlaydi va garchi u mohiyatan talqin qilingan bo'lsa ham.

1980-yillarning o'rtalarida BASIC dasturlash tillarida kalkulyatorlarning murakkab modellarida asosiy tilga aylandi, bu vaqtga kelib u to'liq huquqli yuqori darajadagi tildan foydalanishga imkon beradigan kuchga erishdi. Ko'pgina ishlab chiqaruvchilar, hech qanday so'z aytmasdan, kalkulyatorlarning yuqori darajadagi tili uchun asos sifatida BASIC-ni tanlaganliklari, ushbu tilning ko'rsatilgan vaqtda tarqalishi va mashhurligi haqida gapiradi. Dasturlashtiriladigan kalkulyatorlar uchun asosiy yuqori darajadagidasturlash tili sifatida BASIC dan foydalanish hozirgi kungacha davom etmoqda.

1980-yillarning oxiriga kelib, shaxsiy kompyuterlar ancha kuchliroq bo‘lib, BASIC-da foydalanish uchun qulay bo‘lmagan funksiyalarni (masalan, grafik foydalanuvchi interfeysi ) taqdim etdi. Bundan tashqari, qulay dasturlash muhiti va foydali kutubxonalar (masalan, Borland Paskal bilan birga) sanoat tillarining (birinchi navbatda Paskal va C) arzon tarjimonlarining "ta'lim" va "uy" sinfidagi kompyuterlarda paydo bo'lishi. Turbo Vision kutubxonasi) BASIC uchun kuchli raqobat yaratdi. BASIC, uning ko'plab versiyalari hali ham ishlatilgan va sotilganiga qaramay, o'z o'rnini yo'qota boshladi. Shu bilan birga, ishlayotgan BASIC tizimlarining ko'rinishida tez o'zgarishlar boshlandi.

BASIC dasturlash tili - Microsoft Visual Basic dasturlash tilining paydo bo'lishi bilan ikkinchi hayotga ega bo'ldi. Garchi bu til haqiqatda BASIC ekanligiga rozi bo'lish qiyin bo’lsada, bu dasturlash til evolyutsiyasining mantiqiy xulosasi edi va odatiy kalit so'zlar saqlanib qolganiga qaramay, tuzilishi va xususiyatlar to'plami bo'yicha u asl BASICga qaraganda Paskal tiliga yaqinroqdir. Ammo dasturchi malakasi past bo'lgan Windows uchun amaliy foydali dasturlarni tezda yaratish

vositasi sifatida u juda foydali bo'lib chiqdi va tezda Windows platformasida eng ko'p ishlatiladigan tillardan biriga aylandi. Microsoft Word Basic deb nomlangan variantni yaratdi va Word 97 paydo bo'lgunga qadar uni MS Word da ishlatdi. Ilovalar uchun Visual Basic Application (VBA) varianti 1993 yilda Microsoft Excel 5.0 da, so'ngra 1995 yilda Access 95 da va undan keyin boshqa barcha versiyalarda o'rnatilgan. Microsoft Office to'plamiga kiritilgan hamda Internet Explorer 3.0 va undan yuqori versiyalari, shuningdek, Microsoft Outlook VBScript interpretatorni o'z ichiga olgan.

**Paskal** (inglizcha: *Pascal*) — eng mashhur dasturlash tillaridan biri bo‘lib, o‘rta maktab hamda universitetlarning birinchi kurslarida dasturlashni o‘rgatishda qo‘llaniladi. Ushbu dasturlash tili boshqa bir qator dasturlash tillarini o’rganish uchun asos bo‘lib xizmat qiladi. Paskal imperativ va protsessual dasturlash tili hisoblanadi. Ushbu dasturlash tili dasturlash va ma’lumotlar strukturasidan foydalangan holda yaxshi dasturlash amaliyotlarini rag‘batlantirish uchun kichik va samarali til sifatida ishlab chiqilgan.

Shveysariyaning Syurix shahridagi oliy texnika maktabining professori Niklaus Virt tomonidan 1970-yillarda yaratilgan bo‘lib, 1979-80-yillarda Pascal dasturlash tilining xalqaro standarti tasdiqlangan. Mazkur dasturlash tili o‘zining soddaligi, mantiqiyligi va samaraligi tufayli butun dunyoga tez tarqaldi. Hozirgi paytda barcha hisoblash mashinalari, xususan, mikro EHMlar ham shu dasturlash tilida ishlash imkoniyatiga ega. Dasturlar matnining to‘g‘riligini osonlik bilan tekshirish mumkinligini, ularning ma’nosi yaqqol ko‘zga tashlanishi va oddiyligi bilan ajralib turadi.

Virtning “Algoritmlar + Ma’lumotlar tuzilmalari = Dasturlar” (en:Algorithms_+_Data_Structures_=_Programs) kitobi asosida Paskal dasturlash tili ALGOL60 tilining namunasi bo‘yicha ishlab chiqilgan. Virt ALGOL-X sa’y-harakatlari doirasida tilni takomillashtirish jarayonida ishtirok etdi va ALGOL-W deb nomlangan versiyani taklif qildi. Ammo, bu qabul qilinmadi va ALGOL-X jarayoni to‘xtab qoldi. 1968-yilda Virt ALGOL-X jarayonidan voz kechishga va ALGOL-W’ni yanada takomillashtirishga qaror qildi va uni 1970-yilda Paskal

dasturlash tili sifatida ishlab chiqardi. ALGOL skalyarlari va massivlaridan tashqari, Paskal murakkab ma’lumotlar turlarini aniqlash hamda ro‘yxatlar, tarmoqlanish va grafiklar kabi dinamik va rekursiv ma’lumotlar tuzilmalarini yaratish imkonini beradi.

C (va C oilasidagi ko‘pgina tillardan) farqli o‘laroq, Paskal har qanday murakkab darajadagi protsedura ta’riflariga ruxsat beradi, shuningdek, quyi dasturlar (protseduralar va funksiyalar) ichida ko‘plab ta’riflar va deklaratsiyalarga ruxsat beradi. Shunday qilib, dastur sintaktik jihatdan bitta protsedura yoki funksiyaga o‘xshaydi. Bu ALGOL60’ning blok tuzilishiga o‘xshaydi, lekin ixtiyoriy blok bayonotlaridan faqat protseduralar va funksiyalar bilan cheklanganligi bilan ajralib turadi.

Paskal 1970-yillarda, ayniqsa rivojlanayotgan minikompyuter bozorida muvaffaqiyat qozondi. Kompilyatorlar ko‘plab mikrokompyuterlar uchun ham mavjud edi, chunki bu soha 1970-yillarning oxirida paydo bo‘lgan. U 1980-yillarda universitet darajasidagi dasturlash kurslarida o‘qitish tili sifatida keng qo‘llanilgan va shu davrda tijorat dasturiy ta’minotini yozish uchun ishlab chiqarishda ham foydalanilgan. 1980-yillarning oxiri va 1990-yillarning boshlarida UNIX’ga asoslangan tizimlar ommalashgani va ayniqsa C++ dasturlash tilining chiqarilishi bilan Paskal dasturlash tili o‘rniga C dasturlash tili oilasi undan o‘zib ketdi.

Obyektga yo‘naltirilgan dasturlash uchun mo‘ljallangan Object Pascal nomli dasturlash tili 1985-yilda ishlab chiqilgan. Bu 1980-yillarning oxirida Apple Computer va Borland tomonidan qo‘llanilgan va keyinchalik Microsoft Windows platformasida Delphi dasturlash tili ishlab chiqilgan.

**Delphi** (*talaffuzi*- delfi) —Borland firmasi tomonidan ishlab chiqarilgan dasturlash tili hisoblanadi. Delphi dasturlash tili - Pascal va Object Pascal dasturlash tillaridan bir qancha kengaytirishlar va to‘ldirishlar orqali kelib chiqqan bo‘lib, u ob’yektga yo‘naltirilgan dasturlash tili hisoblanadi. Avvaldan ushbu dasturlash muhiti faqatgina Microsoft Windows Operatsion tizimi uchun dasturlar yaratishga mo'ljallangan, keyinchalik esa GNU/Linux hamda Kylix tizimlari uchun

moslashtirildi, lekin 2002-yilgi Kylix 3 versiyasidan so'ng ishlab chiqarish to'xtatildi, ko'p o'tmay esa Microsoft.NET tizimini qo'llab quvvatlashi to'g'risida e'lon qilindi.

Lazarus proekti amaliyotidagi (Free Pascal) dasturlash tili Delphi dasturlash muhitida GNU/Linux, Mac OS va Windows CE platformalari uchun dasturlar yaratishga imkon beradi.

Delphi— Paskal dasturlash tilining rivojlangan davomchisi sifatida bo‘lgan Turbo Paskal tilining rivojlanishining natijasi hisoblanadi. Paskal tilida asosan Procedura va funksiyalar yordamida dasturlar tuzilgan bo’lib, Turbo Paskal5.5–versiyasidan boshlab obyektga mo‘ljallangan xususiyatlarni o’zida mujassamlashtirdi. Natijada, Delphi — obyektga mo‘ljallangan dasturlash tili, ya’ni metodli sinflar xususiyatlari hamda ulardan tashkil topuvchilarining kompilyatsiya kodi tarkibiga qo‘shdi.

**C++ dasturlash tili.** C++ (/ˌsiːˌplʌsˈplʌs/) – umumiy maqsadli dasturlash tili bo‘lib, Bjarne Stroustrup tomonidan C dasturlash tilining kengaytirilgan versiyasi yoki “C sinflari bilan” ishlashni amalga oshiruvchi dasturlash tili sifatida yaratilgan. Vaqt o'tishi bilan C++ dasturlash tili sezilarli darajada takomillashtirilib imkoniyatlari kengaytirildi. Zamonaviy C++ dasturlash tili hozirgi kunda past darajadagi xotira manipulyatsiyasi imkoniyatlaridan foydalanishdan tashqari ob'ektga yo'naltirilgan, umumiy va funktsional xususiyatlarga ega bo’lgan dasturlash tili sifatida ajralib turadi. U deyarli har doim kompilyatsiya qilingan til sifatida amalga oshiriladi va ko'plab ishlab chiqaruvchilar C++ kompilyatorlarini taqdim etadilar, jumladan Free Software Foundation, LLVM, Microsoft, Intel, Oracle, IBM va boshqalar.

C++ dasturlash tili tizimli dasturlashni, o'rnatiladigan tizimlarni, cheklangan resurslarga ega dasturiy ta'minotni va katta tizimlarga yo'naltirilgan dasturlarni ishlab chiqishga mo’ljallangan bo'lib, uning dizaynida unumdorlik, samaradorlik va foydalanish moslashuvchanligi bilan ajralib turadi. C++ boshqa ko‘plab kontekstlarda ham foydali deb topildi, asosiy kuchli tomonlari dasturiy infratuzilma va resurslar cheklangan ilovalar, jumladan ish stoli ilovalari, video o‘yinlar,

serverlar (masalan, elektron tijorat, Web-qidiruv yoki ma’lumotlar bazalari) va unumdorligi muhim bo’lgan ilovalarni yaratishda asosiy dasturlash tili hisoblanadi.

C++ dasturlash tili Xalqaro Standartlashtirish Tashkiloti (ISO) tomonidan standartlashtirilgan bo‘lib, eng so‘nggi standart versiyasi ISO tomonidan 2020-yil dekabr oyida ISO/IEC 14882:2020 (norasmiy ravishda C++20 nomi bilan tanilgan) sifatida ratifikatsiya qilingan va nashr etilgan. C++ dasturlash tili dastlab 1998 yilda ISO/IEC 14882:1998 sifatida standartlashtirilgan bo‘lib, keyinchalik unga C++03, C++11, C++14 va C++17 standartlari bilan o‘zgartirishlar kiritilgan. Hozirgi C++ 20 standarti ularni yangi funksiyalar va kengaytirilgan standart kutubxona bilan almashtiradi. 1998 yilda dastlabki standartlashtirishdan oldin C++ dasturlash tili daniyalik Bjarne Stroustrup tomonidan 1979 yildan beri Bell laboratoriyasida C tilining kengaytmasi sifatida ishlab chiqilgan bo’lib, C dasturlash tiliga o'xshash samarali va moslashuvchan tilni tashkil qilish uchun yuqori darajadagi xususiyatlarni ta'minladi.

**C# dasturlash tili.** ( / s i ʃ ɑːr p / si sharp ) umumiy maqsadli, ko‘p paradigmali dasturlash tili hisoblanadi. C# dasturlash tili statik terish, leksik qamrovli, imperativ, deklarativ, funktsional, ob'ektga yo'naltirilgan (sinfga asoslangan) va komponentlarga yo'naltirilgan dasturlashni o'z ichiga oladi.

C# dasturlash tili 2000 yilda Microsoft kompaniyasining xodimi Anders Hejlsberg tomonidan ishlab chiqilgan va keyinchalik 2002 yilda Ecma (ECMA-334) va 2003 yilda ISO (ISO/IEC 23270) tomonidan xalqaro standart sifatida tasdiqlangan. Microsoft .NET Framework va Visual Studio bilan birga C# tilini taqdim etdi. To'rt yil o'tgach, 2004 yilda Mono deb nomlangan bepul va ochiq manbali loyiha kross-platformali kompilyatori va ish vaqti muhitini ta'minlay boshladi. C# dasturlash tili oradan o‘n yil o‘tgach, Microsoft Visual Studio Code (kodmuharriri), Roslyn (kompilyator)va yagona .NET platformasini (dasturiy ta’minot ramkasi) chiqardi, ularning barchasi C# ni qo‘llab-quvvatlaydi va bepul, ochiq kodli va kross-platformali dasturlash tili hisoblanadi.

Mazkur ma’lumotning 2021 yil holatiga ko'ra, C# dasturlash tilining eng so'nggi versiyasi C# 10.0 bo'lib, u 2021 yilda .NET 6.0 da ishlab chiqarilgan.

**Java dasturlash tili.** Java dasturlash tili - yuqori darajali, sinfga asoslangan, ob'ektga yo'naltirilgan dasturlash tili bo'lib, u iloji boricha kamroq kod yozishni amalga oshirishga bog'liq bo'lishi uchun mo'ljallangan. Bu umumiy maqsadli dasturlash tili bo‘lib, dasturchilarga *bir marta yozish va istalgan joyda ishga tushirish* imkonini beradi, kompilyatsiya qilingan Java kodi Java-ni qo‘llab-quvvatlaydigan barcha platformalarda qayta kompilyatsiya qilishni talab qilmasdan ishlashi mumkinligini anglatadi. Java ilovalari odatda bayt-kodga kompilyatsiya qilinadi, ular asosiy narsadan qat'iy nazar har qanday Java virtual mashinasida ishlaydi. Java tilining sintaksisi C va C++ ga o‘xshaydi, lekin ularning har biriga qaraganda kamroq past darajadagi imkoniyatlarga ega. Java dasturlash tilining real ish vaqti odatda an'anaviy kompilyatsiya qilingan tillarda mavjud bo'lmagan dinamik imkoniyatlarni (masalan, aks ettirish va ish vaqti kodini o'zgartirish) taqdim etadi. 2019 yil holatidagi Java GitHub ma’lumotlariga ko‘ra, ayniqsa mijoz-server web-ilovalari uchun ishlatiladigan eng mashhur zamonaviy dasturlash tillaridan biri bo‘lib, 9 milliondan ortiq ishlab chiquvchilariga ega.

Java dastlab Sun Microsystems kompaniyasida Jeyms Gosling tomonidan ishlab chiqilgan va 1995 yil may oyida Sun Microsystems Java platformasining asosiy komponenti sifatida chiqarilgan. Asl va mos yozuvlar Java kompilyatorlari, virtual mashinalari va sinf kutubxonalari dastlab Sun kompaniyasi tomonidan xususiy litsenziyalar ostida ishlab chiqarilgan. 2007 yil may oyidan boshlab, Java Community Process spetsifikatsiyalariga muvofiq, Sun o'zining ko'pgina Java texnologiyalarini faqat GPL-2.0 litsenziyasi ostida qayta litsenziyaladi. Oracle o'zining HotSpot nuqtasini taklif qiladi. Java virtual mashinasi, ammo rasmiy ma'lumotnoma ilovasi OpenJDK JVM (Java Virtual Mashina) bo'lib, u bepul ochiq kodli dasturiy ta'minot bo'lib, ko'pchilik ishlab chiquvchilar tomonidan qo'llaniladi va deyarli barcha Linux distributivlari uchun standart JVM hisoblanadi.

2021-yil oktabr holatiga ko‘ra, Java 17 eng so‘nggi versiya hisoblanadi. Java 8, 11 va 17 joriy uzoq muddatli qo‘llab-quvvatlash (LTS) versiyalaridir. Oracle 2019-yil

yanvar oyida Java 8 LTS-ning eski versiyasi uchun so‘nggi nol xarajatli ommaviy yangilanishni tijorat maqsadlarida foydalanish uchun chiqardi, garchi u Java 8-ni shaxsiy foydalanish uchun umumiy yangilanishlar bilan cheksiz muddatga qo‘llab-quvvatlaydi.

**JavaScript dasturlash tili.** ( / ˈdʒ ɑː v ə s k r ɪ p t / ), JavaScript dasturlash tili qisqartirilgan holda JS deb nomlanib, HTML va CSS bilan bir qatorda World Wide Web ning asosiy texnologiyalaridan biri bo'lgan dasturlash tili hisoblanadi. Web-saytlarning deyarli 97% dan ortig'i ushbu dasturlash tilidan foydalangan holda amalga oshiriladi. JavaScript dasturlash tili hodisalarga asoslangan, funksional, imperative va ob’ektga yo’naltirilgan dasturlash tilidir.
JavaScript-bu ECMAScript standartiga mos keladigan yuqori darajali, ko'pincha o'z vaqtida kompilyatsiya qilingan dasturlash tilidir. U dinamik yozish, prototipga asoslangan ob'ektga yo'naltirish va birinchi darajali funktsiyalarga ega. Bu ko'p paradigmali bo'lib, hodisalarga asoslangan, funktsional va imperativ dasturlash uslublarini qo'llab-quvvatlaydi. Unda matn, sanalar, regulyar ifodalar, standart ma’lumotlar tuzilmalari va Hujjat obyekti modeli bilan ishlash uchun amaliy dasturlash interfeyslari (API) mavjud.

Java va JavaScript dasturlash tillarining nomlari o‘xshash bo‘lsa-da, bu ikki dasturlash tili bir-biridan sintaksisi va tegishli standart kutubxonalari hamda dizayn jihatidan tubdab farq qiladi.

## 1.2. Python dasturlash tilining yaratilish tarixi, imkoniyatlari va uni o’rnatish

Python dasturlash tilini yaratilish tarixi asosan, 1980-yillarning oxiri 1990-yillarning boshlariga to’g’ri keladi. O`sha davrlarda uncha taniqli bo`lmagan Gollandiyaning (Centrum Wiskunde & Informatica) CWI instituti xodimi Gvido van Rossum ABC dasturlash tilini yaratilish loyihasida ishtirok etgan edi. ABC dasturlash tili Basic dasturlash tili o`rniga talabalarga asosiy dasturlash konsepsiyalarini o`rgatish uchun mo`ljallangan dasturlash tili edi. Kunlardan birida

Gvido o’zi ishlab turgan dasturlash tilida dastur yozishdan charchadi. Natijada ikki hafta davomida o`zining Macintosh kompyuterida boshqa oddiy dasturlash tilining interpretatorini yozdi, bunda u albatta ABC dasturlash tilining ba’zi bir g`oyalarini o`zlashtirdi.

Shuningdek, Python dasturlash tili uchun ilgari keng foydalanilgan BASIC, Algol-68, C, C++, Modul3 ABC, SmallTalk dasturlash tillarining ko`plab xususiyatlarini o`zida mujassamlashtirib olishga harakat qildi. Natijada, Gvido van Rossum bu dasturlash tilini internet orqali tarqata boshladi. O`sha paytlarda Gvido BBC ning “Monti Pythonning havo sirki” komediyasining muxlisi bo’lganligi bois, o`zi yaratgan dasturlash tilini Monti Python nomiga Python deb atadi (Ilon nomiga emas).

Mazkur zamonaviy dasturlash tilining oddiyligi, dastur sintaksisining soddaligi tufayli unga qiziquchilar va tushunadigan foydalanuvchilar soni shiddat bilan ko’payib, zamoniylashib bordi. Dasturlash tilining shunchaki kichik interpretatori bir nechta funksiyalarga ega edi. Python dasturlash tili 1991-yil dastlabki obyektga yo`naltirilgan dasturlash tillarini biri sifatida 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 dasturlash tilida yozdi va bo`sh vaqtlarida dasturlash 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 Python dasturlash tilini 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, hozirgi kunga qadar Python dasturlash tilining bir qancha versiyalari ishlab chiqildi. Quyidagi jadvalda Python dasturlash tilining yaratilish tarixining jadvali keltirilgan.

| | |
|---|---|
| Python 1.0 - 1994 yil yanvar<br>Python 1.5 - 1997 yil 31 dekabr<br>Python 1.6 - 2000 yil 5 sentyabr<br>Python 2.0 - 2000 yil 16 oktyabr<br>Python 2.1 - 2001 yil 17 aprel<br>Python 2.2 - 2001 yil 21 dekabr<br>Python 2.3 - 2003 yil 29 iyul<br>Python 2.4 - 2004 yil 30-noyabr<br>Python 2.5 - 2006 yil 19 sentyabr<br>Python 2.6 - 2008 yil 1 oktyabr<br>Python 2.7 - 2010 yil 3 iyul | Python 3.0 - 2008 yil 3 dekabr<br>Python 3.1 - 2009 yil 27 iyun<br>Python 3.2 - 2011 yil 20 fevral<br>Python 3.3 - 2012 yil 29 sentyabr<br>Python 3.4 - 2014 yil 16 mart<br>Python 3.5 - 2015 yil 13 sentyabr<br>Python 3.6 - 2016 yil 23 dekabr<br>Python 3.7 - 2018 yil 27-iyun<br>Python 3.8 - 2019 yil 14 oktyabr<br>Python 3.9 - 2020 yil 5 oktyabr<br>Python 3.10 - 2021 yil 4 oktyabr |

Hozirgi vaqtda asosan uchinchi versiyasidan keng foydalanilmoqda.

**Python dasturlash tilining imkoniyatlari. Python** dasturlash tili o’zgaruvchini e’lon qilmagan holda dasturning turli qismlarida bir xil o’zgaruvchilar dasturda har xil turdagi qiymatlarni qabul qilishi mumkin bo’lgan, dinamik tarzda tiplashtirilgan yuqori darajada ishlab chiqilgan umumiy maqsadli zamonaviy dasturlash tilidir. Python dasturlash tili asosan ob'ektga yo'naltirilgan dasturlash tili bo'lib, o’zining g'ayrioddiy xususiyatilari bilan boshqa dasturlash tillaridan tubdan ajralib turadi. Ya’ni, dasturning kod bloklarining bo'sh joylar bilan yozilishi va bo’sh joylarning ahamiyati asosiy hisoblanadi. Python dasturlash tilining asosiy sintaksisi uning boshqa dasturlash tillariga nisbatan minimal ko’rinishga egaligidir. Shuning uchun ham amalda kamdan-kam hollarda hujjatlarga murojaat qilish zarurati tug'iladi. Tilning o'zi interpretator sifatida tanilgan va dastur kodlarini yozishda tayyor skriptlar ishlatiladi. Python dasturlash tilining kamchiligi sifatida C yoki C++ kabi kompilyatsiya qilingan dasturlash tillarida yozilgan shunga o'xshash kodga nisbatan ko'pincha past tezlik va unda yozilgan dasturlarning yuqori xotira iste'moli hisoblanadi.

Standart kutubxona matnni qayta ishlashdan tortib tarmoq ilovalarini yozish vositalarigacha bo'lgan foydali portativ funktsiyalarning katta to'plamini o'z ichiga oladi. Matematik modellashtirish, asbob-uskunalar bilan ishlash, veb-ilovalarni yozish yoki o'yinlarni ishlab chiqish kabi qo'shimcha funktsiyalar ko'p sonli uchinchi tomon kutubxonalari, shuningdek C yoki C ++ tillarida yozilgan kutubxonalar integratsiyasi orqali amalga oshirilishi mumkin, Python esa. tarjimonning o'zi ushbu tillarda yozilgan loyihalarga qo'shilishi mumkin [25] . Python-da yozilgan dasturiy ta'minot uchun ixtisoslashtirilgan ombor ham mavjud - PyPI [45]. Ushbu ombor paketlarni operatsion tizimga osongina o'rnatish vositasini taqdim etadi va Python uchun de-fakto standartiga aylandi [46] . 2019 yil holatiga ko'ra, u 175 000 dan ortiq paketlarni o'z ichiga olgan [45] .

Python eng mashhur tillardan biriga aylandi va ma'lumotlar tahlili, mashinani o'rganish , DevOps va veb-ishlab chiqishda , shu jumladan o'yinlarni ishlab chiqishda qo'llaniladi. O‘qishga qulayligi, oddiy sintaksisi va kompilyatsiyaga hojat yo‘qligi tufayli til dasturlashni o‘rgatish uchun juda mos bo‘lib, sizga e’tiboringizni algoritmlar, tushunchalar va paradigmalarni o‘rganishga qaratish imkonini beradi. Nosozliklarni tuzatish va eksperiment o'tkazishga tilni izohlash mumkinligi katta yordam beradi [47] [25] . Til Google yoki Facebook kabi ko'plab yirik kompaniyalar tomonidan qo'llaniladi [25] . 2021 yil oktabr holatiga ko'ra Python 1-o'rinni egalladiTIOBE (*The Importance Of Being Earnest* ) dasturlash tillarining mashhurlik reytingi 11,27% [48] . Python 2007, 2010, 2018 va 2020 yillarda TIOBE yilning eng yaxshi tili deb e'lon qilingan [49] .

| 2022 yil mart | 2021 yil mart | O'zgartirish | Dasturlash tili | Reytinglar | O'zgartirish |
|---|---|---|---|---|---|
| 1 | 3 | ▲ | Python | 14,26% | +3,95% |
| 2 | 1 | ▼ | C | 13,06% | -2,27% |
| 3 | 2 | ▼ | Java | 11,19% | +0,74% |
| 4 | 4 | | C++ | 8,66% | +2,14% |
| 5 | 5 | | C# | 5,92% | +0,95% |
| 6 | 6 | | Visual Basic | 5,77% | +0,91% |
| 7 | 7 | | JavaScript | 2,09% | -0,03% |
| 8 | 8 | | PHP | 1,92% | -0,15% |
| 9 | 9 | | Assambleya tili | 1,90% | -0,07% |
| 10 | 10 | | SQL | 1,85% | -0,02% |

**Python dasturlash tili imkoniyatlari.** Python dasturlash tili o'rganishga oson, sintaksisi juda ham qulay bo’lgan hamda birga imkoniyatlari yuqori bo'lgan zamonaviy dasturlash tillari qatoriga kiradi.

Python yuqori darajadagi ma'lumotlar strukturasi va oddiy lekin amaradorobyektga 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 qilinadigan til.

 Kengayishga 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 ishla sh

 Robot texnikada

 Matematik va ilmiy hisoblashlarni programmalash

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

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

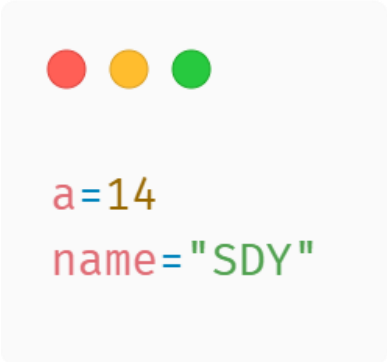
## 2-mavzu: Identifikatorlar, literallar, o’zgaruvchilar va berilganlar turlari.

**Mavzu rejasi:**

1.O'zgaruvchilarni nomlash.
2.Ma'lumotlarning turlari.
3.O'zgaruvchilarga qiymat berish.
4.Ma'lumotlarni bir turdan boshqasiga o’zgartirish
5.O'zgaruvchini o’chirish

Python dasturlash tilida boshqa dasturlash tillaridagi kabi o’zgaruvchilar aniq bir turdagi berilganlarni saqlash uchun xizmat qiladi. Python dasturlash tilida o’zgaruvchilar alfavit belgilari yoki tag chizig’i belgisi bilan boshlanishi va tarkibi son, alfavit belgilari, tag chizig’i belgilaridan iborat bo’lishi, ya`ni bir so’z bilan aytganda identifikator bo’lishi kerak. Bundan tashqari o’zgaruvchi nomi Python dasturlash tilida ishlatiladigan kalit so’zlar nomi bilan mos tushmasligi shart. Masalan, o’zgaruvchi nomi *and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield* kabi kalit so’zlar nomi bilan mos tushishi mumkin emas.

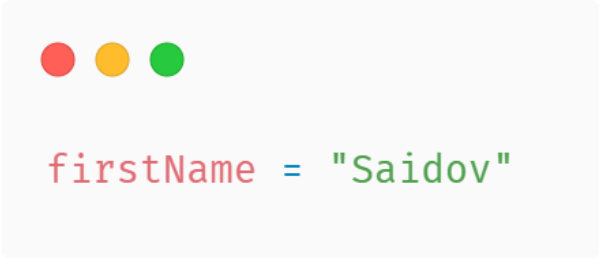
Masalan, o’zgaruvchini aniqlash (hosil qilish) quyidagicha amalga oshiriladi:




1.4.1-rasm

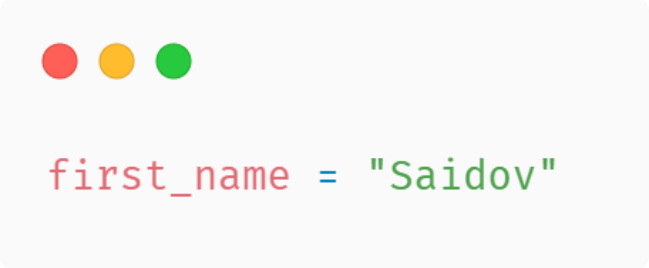
Yuqorida *a* va *name* o’zgaruvchilari yaratildi va ularga qiymat berildi. Shuni alohida ta`kidlash kerakki, Python dasturlash tilida o’zgaruvchini dastlab e`lon qilish degan tushuncha mavjud emas (masalan: c++ tilida *int a* kabi o’zgaruvchi e`lon qilinadi), balki o’zgaruvchi kiritiladi va unga qiymat beriladi (masalan: *a=14*). Berilgan qiymatga ko’ra interpretator o’zgaruvchining turini aniqlaydi. Python dasturlash tilida o’zgaruvchilarni nomlashning ikki turi: “camel case” va “underscore notation” turlaridan foydalanish tavsiya qilingan.

“camel case” turida o’zgaruvchiga nom berilganda, agar o’zgaruvchi nomi alohida so’zlar birikmasidan tashkil topgan bo’lsa, ikkinchi so’zdan boshlab har bir so’zning birinchi harfi katta harfda (katta registr) bo’lishi talab qilinadi. Masalan:




1.4.2-rasm

“underscore notation” turida esa so’zlar orasiga tag chizig’i “_” belgisi qo’yiladi. Masalan:




1.4.3-rasm

O’zgaruvchilar biror bir turdagi berilganlarni saqlaydi. Python dasturlash tilida bir necha xildagi berilganlar turlari mavjud bo’lib, ular odatda to’rtta guruhga ajratiladi:

sonlar, ketma-ketliklar, lug’atlar va to’plamlar:

*bool* (*boolean*) – True va False mantiqiy qiymatlar uchun;

*int* – butun sonlar uchun, butun turdagi songa kompyuter xotirasida 4 bayt joy

ajratiladi;

*float* – suzuvchan nuqtali sonlar (haqiqiy sonlar) uchun, haqiqiy sonlarni

saqlash uchun kompyuter xotirasidan 8 bayt joy ajratiladi; *complex* – kompleks sonlar uchun;

*str* – satrlar uchun, Python 3.x versiyasidan boshlab satrlar bu- Unicode

kodirovkasidagi belgilar ketma-ketligini ifodalaydi;

*bytes* – 0-255 diapazondagi sonlar ketma ketligi uchun

*byte array* – baytlar massivi uchun;

*list* – ro’yxatlar uchun;

*tuple* – kortejlar uchun;

*set* – tartiblanmagan unikal ob`ektlar kollektsiyasi uchun;

*frozen set* – set singari, faqat u o’zgartirilishi mumkin emas (immutable);

*dict* – lug’atlar uchun. Har bir element kalit so’z va qiymat juftligi ko’rinishida ifodalaniladi.

Python –dinamik turlarga ajratuvchi dasturlash tili hisoblanadi. Yuqorida aytib o’tilganidek, Python dasturlash tilida o’zgaruvchi turi unga yuklangan qiymat orqali aniqlanadi. Agarda o’zgaruvchiga bittalik (‘) yoki ikkitalik (“) qo’shtirnoq yordamida satr yuklansa, o’zgaruvchi *str* turiga ega bo’ladi, agarda o’zgaruvchiga butun son yuklansa – *int*, haqiqiy son yuklansa (masalan: 3.14) yoki eksponentsial ko’rinishdagi qiymat yuklansa (masalan: 11e-1) u *float* turiga ega bo’ladi.

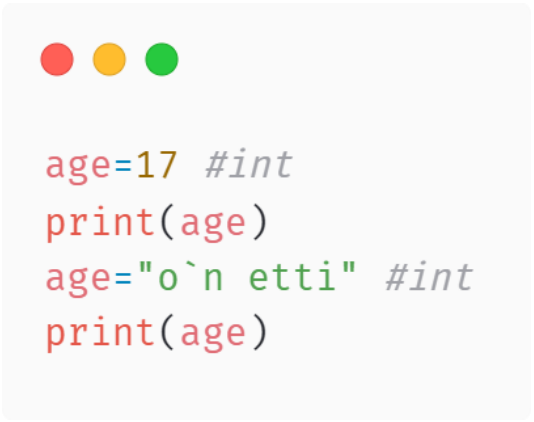
Masalan:

```
user_id = 234  #
int x = 1.2e2  # = 1200.0
float y = 6.7e-3 # = 0.0067
float z = 1.223 # float
user_password = "sdy123"  #  str b = True # bool
```

1.4.4-rasm

Python dasturlash tilida haqiqiy (float) turidagi o’zgaruvchilar [$-10^{308}$ , $+10^{308}$] oraliqdagi sonlar bilan hisoblash ishlarini amalga oshirsa bo’ladi, lekin faqat 18 ta raqamlar ketma-ketligi ko’rinadi (konsol ekraniga chiqarilganda). Ixtiyoriy  katta yoki  kichik sonlarni o’zgaruvchidagi ifodasi 18 ta belgidan oshib ketsa, u holda eksponentsial orqali yaxlitlab ifodalanadi.

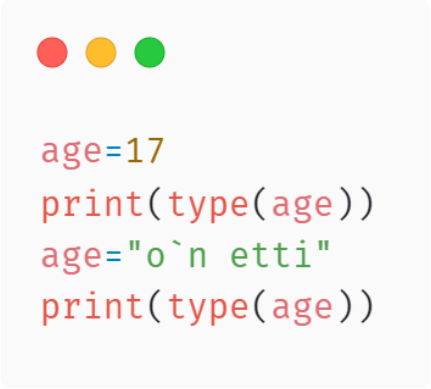
Shuni ham ta`kidlash kerakki, Python dasturlash tilida o’zgaruvchiga yangi qiymat berish orqali uning turi o’zgartirilishi mumkin. Masalan:




1.4.5-rasm

Ushbu dasturda dastlab *age =17* ifodasi orqali *age* o’zgaruvchisi *int* turiga ega edi. Keyingi *age = "o`n etti"* ifoda bilan uning turi *str* turiga o’zgartirildi. Bundan keyingi jarayonlarda *age* o’zgaruvchisi eng ohirgi yuklangan qiymat turiga mos bo’ladi.

O’zgaruvchilarning turini aniqlashda **type()** – funksiyasidan foydalaniladi. Masalan:




1.4.6-rasm

Konsol ekranidagi natija:

*<class 'int'>*

*<class 'str'>*

## Nazorat savollari

1.Pythonda qanday ma’lumot turlari bor?

2.Pythonda o’zgaruvchining turini qaysi funksiya orqali aniqlaymiz.

3.Pythondagi har bir tur uchun bitta o’zgaruvchi e’lon qiling va unga mos qiymatlarni bering.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

## 3-mavzu: Python dasturlash tilida arifmetik va mantiqiy amallar

**Mavzu rejasi:**

1. Matematik operatorlar,
2. Ikkilik operatorlar.
3. Ketma-ketlik operatorlari
4. O’zlashtirish operatorlari.
5.Operatorning bajarilish darajasi

**Arifmetik amallar.** Python dasturlash tilida asosiy arifmetik amallar o’z ma`nosi bo’yicha qo’llaniladi:

**+ - qo’shish amali:**

Ikki sonni yig’indisi

```
print(6 + 5)  # 11
```

1.5.1-rasm

**// - butun qismli bo’lish amali:**

Ikki sonni bo’linmasi (ushbu amal bo’lish natijasining faqat butun qismini qaytaradi, qoldiq qismi tashlab yuboriladi)

```
print(6 // 5)  # 1
```

1.5.2-rasm

Ifodada bir nechta arifmetik amallar ketma-ket kelgan bo’lsa, ular prioriteti (ustunligi) bo’yicha bajariladi. Dastlab, yuqori prioritetga ega bo’lgan amallar bajariladi. Amallarning prioriteti kamayish tartibida quyidagi jadvalda ifodalangan:

| **Amallar** | **Yo’nalish** |
|---|---|
| ** | Chapdan-o’ngga |
| *, /, //, % | Chapdan-o’ngga |
| +, - | Chapdan-o’ngga |

Misol sifatida quyidagi ifodani qaraymiz:


1.5.3-rasm

Bu erda dastlab eng yuqori prioritetga ega bo'lgan amal – darajaga ko'tarish amali bajariladi (2 ** 5 =32). Keyin ko'paytma (32 * 3 = 96), butun qismli bo'lish (12 // 7 = 1), qo'shish (1 + 96 = 97) va ayirish (97 – 4 = 93) amallari bajariladi. Ifoda bajarilishi natijasida 93 soni konsol ekraniga chiqariladi.

Amallarni qavsga olish orqali ularning bajarilish ketma-ketligini o'zimiz xoxlagan tartibga keltirib olishimiz ham mumkun. Masalan, yuqoridagi ifodani quyidagicha qayta yozamiz:


1.5.4-rasm

Natijada konsol ekraniga -31 soni chiqariladi.

Shuni alohida ta`kidlash kerakki, arifmetik amallar butun sonlar uchun qanday tartibda bajarilsa, suzuvchan nuqtali haqiqiy sonlar uchun ham xuddi shunday bo'ladi. Agarda ifodada loaqal bitta haqiqiy son ishtirok qilsa natija haqiqiy turda ifodalanadi.

Yuqoridagi barcha arifmetik amallarni o’zlashtirish amali (=) bilan birgalikda (arifmetik amal va undan keyin “=” belgisi ketma-ket yoziladi) ishlatish mumkin. Masalan: +=, -=, *=, /=, //=, %=, **=. Bunday hollarda ifodaning o’ng tomonidagi barcha amallar hisoblanib, chiqqan natija chap tomondagi o’zgaruvchi natijasi bilan mos arifmetik amal bajariladi va natija yana chap tomondagi o’zgaruvchiga yuklanadi. Masalan:

```
son=2
son+=3 #son=son+3 ga teng kuchli amal
print(son) #4
son-=1; print(son) #4
son*=4; print(son) #16
son//=2; print(son) #8
son**=; print(son) #64
```

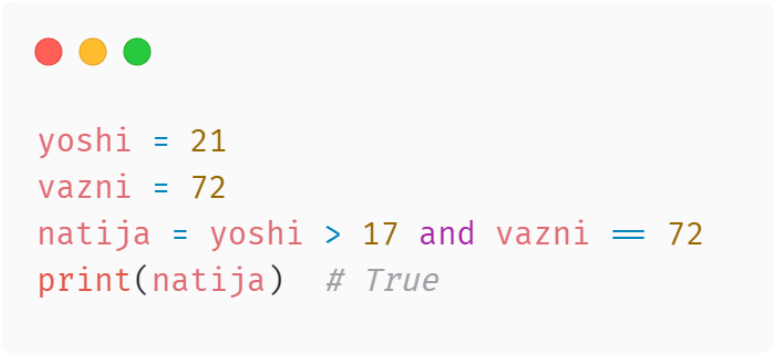
1.5.4-rasm

Yuqoridagi misolda hisoblash natijalari kommentariyalarda ko’rsatilgan.

**Mantiqiy amallar.**

Murakkab shartli ifodalarni yozish, odatda mantiqiy amallar yordamida amalga oshiriladi. Python dasturlash tilida quyidagi mantiqiy operatorlar mavjud:

***and*** (mantiqiy ko’paytirish). Murakkab ifodadagi biror bir qism ifodani qiymati *False* bo’lsa, ifodaning yakuniy qiymati *False* , aks holda *True* qiymat qaytaradi. Masalan:



```
yoshi = 21
vazni = 72
natija = yoshi > 17 and vazni == 72
print(natija)  # True
```

1.11.1-rasm

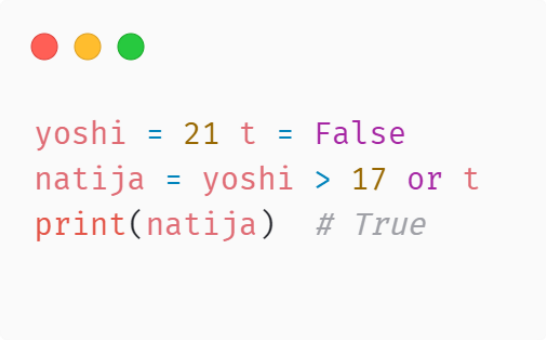
Yuqoridagi dasturda murakkab mantiqiy amal ikki qismdan yoshi > 17 va vazni > 56 qismlardan tashkil topgan bo'lib, ular *and* mantiqiy operatori bilan birlashtirilgan. Agarda ikkala mantiqiy amal *True* qiymat qaytarsa ifodaning qiymati *True* bo'ladi, aks holda *False* qiymat qaytaradi.

Matriqiy ifodalarda faqatgina taqqoslash amallaridan foydalanish shart emas. Ixtiyoriy mantiqiy amal yoki *boolean* turidagi qiymatlar (*True*, *False*) ham ishlatilishi mumkin. Masalan:

```
yoshi = 21
vazni = 72
t = True
natija = yoshi > 17 and vazni > 56 and t
print(natija)  # True
```

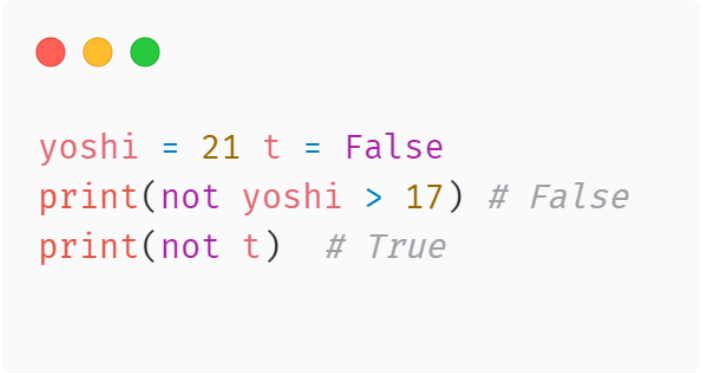
1.11.2-rasm

***or*** (mantiqiy qo'shish). Agarda ifodadagi biror bir qism ifoda *True* qiymat qaytarsa, yakuniy natija ham *True*, aks holda *False* bo'ladi.




1.11.3-rasm

***not*** (mantiqiy inkor). Ifodaning qiymatini *True* bo’lsa, natija *False* va aksincha.




1.11.4-rasm

*and* operatorining biror bir operandi *False* qiymatga ega bo’lsa, u holda boshqa operand qiymati tekshirib (hisoblanib) o’tirilmaydi, har doim natija *False* bo’ladi. Bunday xususiyat ish unumdorligini bir oz bo’lsada oshirish imkonini beradi. Xuddi shunaqa xususiyat *or* operatori uchun ham o’rinli. Ya`ni *or* operatorining biror bir operandi qiymati *True* qiymatga ega bo’lsa, boshqa operandlar tekshirilmaydi, natija sifatida har doim *True* qiymati qaytariladi.

Agar bitta ifodada bir nechta mantiqiy operatorlar qatnashgan bo’lsa, u holda ularning ustunligiga (prioritetiga) alohida e`tibor qatarish kerak. Dastlab *not* operatori keyin *and* va eng so’ngra *or* operatori bajariladi. Masalan:

```
yoshi = 22 xolati = False vazni = 58
natija = vazni == 58 or xolati and not yoshi > 21  # True
print(natija)
```

1.11.5-rasm

Ushbu dasturda keltirilgan ifodadagi mantiqiy amallar kuyidagi ketmaketlikda bajariladi:

1. *not yoshi > 21* mantiqiy ifoda *False* qiymat qaytaradi;
2. *xolati and False* (*not yoshi > 21*) esa *False* qiymat qaytaradi;
3. *vazni == 58 or True* (*xolati and not yoshi > 21*) esa *True* qiymat qaytaradi.

Shuni alohida ta`kidlash kerarki, mantiqiy ifodalarda mantiqiy amallarning bajarilish ketma-ketligini qavslar (, ) yordamida o'zgartirish mumkin.

Nazorat savollari

1.Amallar prioriteti nima?

2.Berilgan 3 xonali sonning 10liklar xonasidagi raqamni chiqaruvchi dastur tuzing. (// va % amalllaridan foydalaning)

3. Pythonda kiritilgan matnni o'zini chiqaruvchi dastur yozing. Dasturni komentariyalarda har bir qator uchun tushuntirib bering.

4.Pythonda *or* va *and* mantiqiy operatorlarni farqini tushuntiring?

5.Pythonda inkor opeatorini qaysi?

6.3 ta son beriladi va shu sonlardan eng kata va eng kichigining o'rta arifmetigini hisoblovchi dastur tuzing?.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

## 4-mavzu:Pyton dasturlash tilida shartli o’tish operatori

**Mavzu rejasi:**

1.Tarmoqlanuvchi operator If... Else
2.Tarmoqlanuvchi operatorning  if . . . elif va boshqa ko’rinishlari

*if* shartli o’tish operatorida shart qo’llanilib, uning natijasiga ko’ra dastur bajarilishi u yoki bu qatorga  yo’naltiriladi. U quyidagi umumiy ko’rinishga ega:

***if*** *mantiqiy ifoda**:***

*ifodalar*

***elif*** *mantiqiy ifoda**:***

*ifodalar*

***else:***

*ifodalar*

*if* shartli o’tish operatorining eng sodda ko’rinishida *if* kalit so’zidan keyin mantiqiy ifoda yoziladi va undan keyin albatta ikki nuqta (:) qo’yiladi. Keyingi qatordan amallar yoziladi. Har bir amal alohida qatorda yozilishi yoki ularni nuqta

vergul (;) bilan ajratgan holda bitta qatordan yozish talab qilinadi. Shuni alohida ta`kidlash kerakki Python dasturlash tilida boshqa tillardagi kabi if shart amalini tana qismini ifodalovchi maxsus belgilar mavjud emas (masalan: c++, c# da *{,}* blok belgilari ishlatiladi). Shu sababli uning tana qismidagi ifodalar *if* kalit so'ziga nisbatan bitta xat boshi (to'rtta probel) belgisi tashlab yoziladi. Masalan:

```
yoshi = 21
if yoshi >18:
    print("Kirishga ruxsat beriladi")
print("Tamom")
```

1.15.1-rasm

Bu erda *if* kalit so'zidan keyin *yoshi >18* mantiqiy ifoda kelgan. Tana qismi bitta ifodadan tashkil topgan, ya`ni *print("Kirishga ruxsat beriladi")* va u *if* ga nisbatan bitta xat boshi tashlab yozilgan. Keyingi qatordagi *print("Tamom")* ifodasi *if* ning tana qismiga tegishli emas, shuning uchun u *if* bilan bir ustunda yozishgan va bu xabar shart bajarilish-bajarilmasligidan qat`iy nazar har doim konsol ekraniga chiqariladi.

Agarda *print("Tamom")* ifodasi oldiga bitta xat boshi qo'ysak, u holda ushbu ifoda ham *if* blokiga tegishli bo'lib qoladi, ya`ni

```
yoshi = 21
if yoshi >18:
    print("Kirishga ruxsat beriladi")
    print("Tamom")
```

1.15.2-rasm

Ushbu holatda shart bajarilsa, ikkala xabar ham konsol ekraniga chiqariladi, aks holda hech biri chiqarilmaydi.

*if* shart ifodasi *false* qiymat qaytaradigan holatda qandaydir amal bajarilishini aniqlash uchun *else* blokida bajarilishi kerak bo'lgan amallar yoziladi. Masalan:

Agar *yoshi >18* shart bajarilsa *if* blokidagi aks holda *else* blokidagi amallar bajariladi.

```
yoshi = 21
if yoshi >18:
    print("Kirishga ruxsat beriladi")
else:
    print("Kirishga ruxsat beriladi")
```

1.15.3-rasm

Bir necha alternativ shartlarni ishlatish uchun qo'shimcha *elif* blokidan foydalaniladi.

```
# ax^2+bx+c=0 kvadrat tenglama echimlari soni
a= int(input("a="))
b= int(input("b="))
c= int(input("c="))
d = b**2 - 4*a*c
if d >0:
    print("Tenglama 2 ta haqiqiy echimga ega")
elif d == 0:
    print("Tenglama 1 ta haqiqiy echimga ega")
else:
    print("Tenglama haqiqiy echimga ega emas")
```

1.15.4-rasm

**Ichma-ich joylashgan *if* shart amali.** *if* shart operatori o'z navbatida boshqa *if* shart operatorlaridan tashkil topgan bo'lishi mumkin. Bunday holatga ichma – ich joylashgan shart ifodasi deyiladi. Masalan:

```
protsent = int(input("Protsentni kiriting: "))
if protsent > 10:
    print("10% dan katta")
if protsent > 20:
        print("20% dan katta")
```

1.15.5-rasm

Yuqoridagi misolda ichki *if* ifodasi tashqarisidagiga nisbatan bitta xat boshi tashlab yozilili shart, aks holda ichma – ich joylashgan shatr operatori bo'lmay, alohida shart operatori hosil qilingan bo'ladi.

Quyidagi *if* operatoriga misolda oylik maoshdan shkala bo'yicha tutib qolinadigan jami daromad solig'ini hisoblovchi dastur tuzilgan:

```
# Qoidaga ko`ra daromad solig`i eng kam ish haqiga(EKIH) bog`liq
maosh = int(input("Oylin summasini kiriting"))
EKIH = int(input("Eng kam ish haqini kiriting"))
dar_soliq =0
if maosh < 6 * EKIH:
    dar_soliq= maosh*0.065
elif maosh< 10 * EKIH:
    dar_soliq = 6 * EKIH * 0.065 + (maosh - 6* EKIH)* 0.165
else:
    dar_soliq = 6 * EKIH * 0.065 + 4 * EKIH * 0.165 +  (maosh - 10* EKIH)* 0.225
print("Oylikdan ushab qolingan darmod solig`i: ",dar_soliq)
```

1.15.6-rasm

**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

## Nazorat savollari

1.Pythonda ichma-ich joylashgan ifodalarni tushuntiring?

2. Temurning bir poda qo‘ylari bor. U sizga qo‘ylarining jami oyoqlari sonini aytadi. Siz esa podadagi qo‘ylarda jami bo‘lib nechta quloq borligini topishingiz kerak. Agar Temur sanashda adashib ketgan bo‘lsa, -1 sonini, aks holda, masalada so‘ralgan sonni chop eting.

3.Son beriladi shu son qaysi oyga tegishli ekanini topuvchi dastur tuzing.Agar son 12 dan kata bo’lsa “Bunday oy mavjudmas” degan natija qaytaring. Bu dasturni ichma-ich shart amali yordamida bajaring.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

## 5-mavzu:Pyton dasturlash tilida For… takrorlash operatori

**Mavzu rejasi:**

1.Takrorlanuvchi operator For …
2.Range() va enumerate() funksiyalari

### *for* takrorlash operatori

Yana bir takrorlash operatori – *for* operatori hisoblanadi. *for* takrorlash operatori qandaydir sonlar kollektsiyasidagi har bir son uchun chaqiriladi. Sonlar kollektsiyasi *range*() funksiyasi, *list*() funksiyasi yoki [,] qavslarda foydalanuvchi tomonidan shakllantirilgan ro’yxatlar orqali hosil qilinadi. Quyida *for* takrorlash operatorining formal aniqlanishi keltirilgan: *for* int_var *in* funksiya_range:

*instruktsiyalar*

*for* kalit so’zidan keyin *int_var* o’zgaruvchisi (o’zgaruvchi nomi ixtiyoriy bo’lishi mumkin) keladi va u faqat butun turdagi qiymatlar qabul qiladi, undan keyin *in* kalit so’zi (*in* operatori) va *range* funksiyasi chaqirilgan va oxirida “:” belgisi bilan takrorlash operatori asosiy qismi tugaydi. *for* takrorlash operatorining

tana qismi bir yoki bir nechta instruktsiyalardan tashkil topishi mumkin va ular asosiy qismga nisbatan bitta xat boshi tashlab yoziladi.

Takrorlash operatori bajarilganda *range()* funksiyasi hosil qilgan sonlar kollektsiyasidan sonlar ketma-ket *int_var* o'zgaruvchisiga uzatiladi. Sikl bo'yicha barcha sonlar ketma-ket o'tib bo'lingandan keyin takrorlash operatori o'z ishiti tugatadi.

Quyida 1 dan *n* gacha bo'lgan sonlar yig'indisini hisoblash dasturi *for* operatori yordamida amalga oshirilgan:

```
sum = 0
n = int(input("n="))
for i in range(1,n+1):
    sum = sum + i
print("summa(1+...+n) =", sum)
```

1.16.2-rasm

Dastlab konsol ekranidan butun son kiritiladi. Siklda *i* o'zgaruvchisi aniqlangan bo'lib, u *range()* funksiyasidan qaytarilgan qiymatni o'zida saqlaydi. Bu erda *range()* funksiyasi 2 ta parametr qabul qilgan. Birinchisi sonlar kollektsiyasini boshlang'ich qiymati va ikkinchisi oxirgi qiymati ( oxirgi qiymat kollektsiyaga kirmaydi). Natijada *range()* funksiyasi [1,…,*n*-1] intervaldagi sonlarni ketma-ket qiymat sifatida qaytaradi va har bir qiymat uchun sikl operatorining tana qismi bajariladi.

***range* funksiyasi**. *range* funksiyasining quyidagi shakllari mavjud:

*range(stop)* – *0* dan *stop* gacha (*stop* kirmaydi) bo'lgan barcha sonlarni qaytaradi; *range(start, stop)* – *start(kiradi)* dan *stop (kirmaydi)* gacha bo'lgan barcha

butun sonlarni qaytaradi; *range(start, stop, step) – start (kiradi)* dan *stop (kirmaydi)* gacha bo'lgan
barcha butun sonlar *step* qadam bilan hosil qilinadi va qaytaradi.

Masalan:

```
print(list(range(5)))        #[0, 1, 2, 3, 4]
print(list(range(1,5)))      #[1, 2, 3, 4]
print(list(range(1,5,2)))    #[1, 3]
print(list(range(-5,5,3)))   #[-5, -2, 1, 4]
```

1.16.3-rasm

Bu erda *range(5)* funksiyasi [*0, 1, 2, 3, 4*] oraliqdagi sonlar kollektsiyasini qaytargan va qaytarilgan sonlarni ro'yxatda jamlash uchun *list()* funksiyasi qo'llanilgan. *range()* funksiyasining boshqa holatlarda qanday qiymatlar hosil qilishini yuqoridagi dasturdan ko'rish mumkin.

**Ichma-ich joylashgan sikllar.** Biror bir takrorlash operatori tanasida boshqa takrorlash operatorining ishlatilishiga ichma-ich joylashgan sikl deyiladi. Masalan:

```
for i in range(1, 5):
    for j in range(1, 5):
        print(i + j, end="\t")
    print("\n")
```

1.16.4-rasm

Natija konsolda quyidagi ko'rinishda chiqariladi:


1.16.5-rasm

**Rasm №2.** Natijaning konsol ekranidagi ko'rinishi

Yuqoridagi holatda *for i in range(1,5)* bo'yicha tashqi siklning har bir iteratsiyasi uchun, *for j in range(1,5)* bo'yicha ichki sikl bajariladi.

**Nazorat savollari**

1.Pythonda *range* metodini tushuntiring?

2.Pythonda *for siklini* tushuntiring?

3. Pythonda 100 gacha bo'lgan butun sonlardan juftlar yig'indisidan toqlar yig'indisining ayirmasini hisoblang.

4. Pythonda ichma-ich takrorlash sikllarini tushuntiring.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. тит. англ.

**6-mavzu: Python dasturlash tilida While …takrorlash operatori**

**Mavzu rejasi:**

1.While takrorlash operatori.
2.Continue operatori: takrorlashning keyingi iteratsiyasiga o‘tish.
3.Break operatori: takrorlashni bekor qilish yoki buzish

### while takrorlash operatori

while takrorlash operatori quyidagi umumiy ko’rinishga ega:

while *shart ifodasi*:

*instruktsiyalar while* kalit so’zidan keyin shart ifodasi ko’rsatiladi va ushbu shart ifodasi rost qiymat (*True*) bo’lar ekan amallar ketma-ketligi takror va takror bajarilishda davom ettiriladi. *while* operatorining barcha insturtsiyalari undan keyingi qarorda yoziladi va u *while* kalit so’zidan bitta xat boshi tashlab yoziladi. Masalan:

```
sum = 0
n = int(input("n="))
i = 1
while i <= n:
    sum = sum + i
    i += 1
print("summa(1+...+n) =", sum)
```

1.16.1-rasm

Yuqoridagi misolda 1 dan *n* gacha bo’lgan sonlar yig’indisi hisoblash dasturi *while* operatori yordamida amalga oshirilgan. E`tibor berilsa *while* operatorining insturktsiyalari undan keyingi qatorda bitta xat boshi tashlab yozilgan. Ushbu holatda *while* operatori 2 ta instuktsiyalardan tashkil topgan (*sum = sum + i* va *i += 1*).

**Sikldan chiqish. *break* va *continue* operatorlari.** Sikllarni boshqarish uchun *break* va *continue* kabi maxsus operatorlardan foydalaniladi. *break* operatori sikldan chiqish uchun ishlatiladi. *continue* operatori siklning navbatdagi iteratsiyasiga o'tish uchun ishlatiladi.

Odatda *break* operatori siklda shart operatorlari bilan birga qo'llaniladi, masalan:

```
while True:
    ch = input("Chiqish uchun 'Y' klavishini bosing")
    if ch.lower() == 'y':
        break
    s=0
    n = int(input("n="))
    for i in range(1,n+1):
        s += i
    print("Summa(1, ... ,n)=",s)
```

1.16.6-rasm

Yuqoridagi dasturda foydalanuvchi tomonidan kiritilgan *n* uchun 1 dan *n* gacha bo'lgan sonlar yig'indisini hisoblash amalga oshirilgan. Agar foydalanuvchi yana boshqa son uchun yig'indini hisoblamoqchi bo'lsa, dasturdan chiqib ketmasdan uni davom ettirishi mumkin. Buning uchun u 'Y" belgisidan boshqa ixtiyoriy belgini ekrandan kiritishi kerak. Sikldan chiqish sharti *if ch.lower() == 'y'* da tekshirilgan.

**Nazorat savollari**

1.Pythonda *break* va *continue* operatorlarining farqini tushuntiring.
2.Pythonda *while* takrorlash siklini tushuntiring?
3. Pythonda 100 gacha bo'lgan butun sonlardan juftlar yig'indisidan toqlar yig'indisining ko'paytmasini hisoblang.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

## 7-mavzu: Sonlar bilan ishlash funksiyalari

**Mavzu rejasi:**

1.Sonlar bilan ishlashning o'rnatilgan funktsiyalari va usullari

2.Math moduli. Matematik funktsiyalar.

3.Random moduli. Tasodifiy sonlari shakllantirish.

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

```
son = 13.46

# sonning round metodi
rounded_number = round(son)
print(rounded_number)

# Output: 13

absolute_number = abs(son)
print(absolute_number)

# Output: 20
```

1.8.1-rasm

Haqiqiy sonlar ustida bajarilishi mumkin bo`lgan ba`zi 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.

```
print(1.5.is_integer())
#False
print(1.0.is_integer())
#True
print(1.4142135623730951.is_integer())
#False
```

1.8.2-rasm

```
# import the math module
import math

# 0 sonning ildizi
print(math.sqrt(0))

# 4 sonning ildizi
print(math.sqrt(4))

# 3.5 sonning ildizi
print(math.sqrt(3.5))
```

1.8.3-rasm

```
# Math kutubxonasini yuklaymiz
import math

# PI sonini chop etamiz
print (math.pi)
```

1.8.4-rasm

**Nazorat savollari**

1.Pythonda sonni butun yoki haqiqiylikka tekshiruvchi metod qaysi va u qanday natijalar qaytardi?

2.Pythonda π sonini 16 lik sanoq sistemasiga o'tkazing?

3.Pythondagi Math kutubxonasidagi *sqrt* metodini tushuntiring.

4. Pythondagi Math kutubxonasidagi *abs* metodini tushuntiring.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. тит. англ.

**8-mavzu: Satrlar bilan ishlash funksiyalari**

**Mavzu rejasi:**

1.Satrlarni yaratish.
2.Maxsus belgilar.
3.Satrlar ustida amallar.
4.Satrlarni formatlash.

5.Format() usuli.
6.Satrlar bilan ishlash funksiyalari va usullari

Satrlar – qo’shtirnoq ichiga olingan *Unicode* kodidagi belgilar ketma-ketligi orqali ifodalanadi. Python dasturlash tilida satriy ifoda apostrof (‘) va (“) qo’shtirnoqlar orqali berilishi munkin. Uchta ketma-ket kelgan apostrof ham satrlarni ifodalashda ishlatiladi.

```
ismi = "Yusupov"
familiyasi = 'Yusuf'
print(ismi, familiyasi)  # Yusupov Yusuf
```

1.13.1-rasm

Satrlar ustida eng keng tarqalgan amallardan biri bu ularni birlashtirish yoki konkatenatsiya amali hisoblanadi. Satrlarni birlashtirish uchun + amali qo’llaniladi. Masalan:

```
ism = "Yusuf"
yosh = 33
info = "Ismi: "+ ism + " yoshi: "+ str(yosh)
print(info)
# Ismi: Yusuf yoshi: 33
```

1.13.2-rasm

Agar satr va sonlarni birlashtirish talab qilinsa, u holda *str()* funksiyasi yordamida sonni satr turiga keltirish kerak bo'ladi. Masalan:

```
ism = "Yusuf" yosh = 33
info = "Ismi: "+ ism + " yoshi: "+ str(yosh) print(info)
# Ismi: Yusuf yoshi: 33
```

1.13.3-rasm

**Maxsus belgilar**: Python dasturlash tilida boshqa tillardagi kabi quyidagi maxsus belgilar mavjud:

| **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) |

Ekran bilan ishlash ketma-ketliklari.

Quyidagi misolda yuqoridagi barcha maxsus belgilarni qo’llangan holat uchun dastur keltirilgan.

```
print("1-chi kurs\n\"O\'MU\"\ttalabasi")
```

1.13.4-rasm

Konsol ekraniga quyidagicha natija chiqariladi:

*1-chi kurs*

*"O'MU"  talabasi*

**Satrlarni taqqoslash:** Satrlarni taqqoslashda satrda ishtirok etayotgan belgilarning registriga alohida e`tibor qaratish lozim. Har qanday raqam ixtiyoriy alfavit belgisidan shartli kichik hamda katta registrli alfavit belgilari kichik registrli avfavit belgilaridan shartli kichik sanaladi. Masalan:

```
str1 = "1a" str2 = "ab"
str3 = "Aa" print(str1 > str2)  # False, chunki str1 ning birinchi
                    # belgisi raqam
print(str2 > str3)  # True, chunki str2 ning birinchi
# belgisi kichik registrga ega
```

1.13.5-rasm

Yuqoridagi dasturda *“1a”>”ab”* sharti *False* qiymat qaytaradi. Chunki raqam alfavit belgisidan shartli kichik hisoblanadi. Shuni alohida ta`kidlash kerakki, ikki satr solishtirilganda ularning mos tarkibiy elementlari solishritiladi(*“1a”>”ab”* holatda, dastlab 1 va “a” tekshiriladi). Agarda solishtirish natijasi teng bo’lsa navbatdagi mos elementlari solishtiriladi. Jarayon birinchi teng bo’lmagan holat

topilganda yoki satrlardan birining oxiriga yetib kelinganda tugatiladi. Agar satrlarning dastlabki barcha mos elementlari teng, faqat ularning uzunliklari farqli bo'lsa, u holda uzunligi katta satr shartli katta bo'ladi.

Masalan: *"abcd"<"abcde"* sharti *True*

Bundan tashqari satrlar ustuda amal bajaradigan *upper*() va *lower*() funksiyalari mavjud bo'lib, satr tarkibidagi alfavit belgilarni mos ravishda kichik va katta registrlilariga almashtirish uchun ishlatiladi. Masalan:

```
str1 = "Kitob"
str2 = "kitob"
print(str1 == str2)  # False - chunki ularni birinchi
                     # harflari turli registrda
print(str1.lower() == str2.lower())  # True chunki ikkala
                      # satr ham kichik registrga keltirilgan
```

1.13.6-rasm

**Nazorat savollari**

1.Pythonda **\v** va **\n** ning faqrini tushuntiring?
2.Pythonda *lower()* va *upper()* metodining vazifasini tushuntiring?
3."AssalomuAlaykum" matniga shunday maxsus belgi qo'yingki "Assalomu" birinchi qatorda va "Alaykum" ikkinchi qatorga chiqaruvchi dastur tuzing.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**9-mavzu: Satrdagi ma’lumotlarni izlash**

**Mashg’ulot rejasi:**

1.Satrdagi ma’lumotlarni qidirish va almashtirish.
2.Satrning kontent turini tekshirish.
3.Satr sifatida berilgan ifodalarni hisoblash.
4.Bytes ma'lumotlar turi.Satrni shifrlash

**Satr ostilarini olish**

Agar zarur bo'lsa, satrdan nafaqat bitta belgini, balki satr ostisini ham olishimiz mumkin. Buni amalga oshirish uchun quyidagi sintaksisdan foydalanamiz:

- string[: end] – 0-indeksdan boshlab *end*gacha belgilar ketma-ketligini oladi;
- string[start: end] – *start* indeksdan *end* indeksigacha bo’lgan belgilarni oladi;
- string[start: end: step] – *step* qadam bilan *start* indeksdan boshlab *end* indeksgacha bo’lgan belgilar kerma-ketligini oladi.

Satr ostisini olish uchun barcha variyantlardan foydalanamiz:

```
string = "hello world"

#  0 dan 5 gacha belgilar
sub_string1 = string[:5]
print(sub_string1)  # hello

#  2 dan 5 gacha belgilar
sub_string2 = string[2:5]
print(sub_string2)  # llo

# 2 dan 9 gacha bitta belgi tashlab
sub_string3 = string[2:9:2]
print(sub_string3)  # lowr
```

4.1.4-rasm

Satr uzunligi (satrdagi belgilar soni)ni olish uchun *len()* funksiyasidan foydalanish mumkin:

```
print(ord("A"))     # 65
```

4.1.5-rasm

```
string = "hello world"
length = len(string)
print(length)   # 11
```

4.1.6-rasm

**Satrdan izlash**

Satrdan *term* satr ostisini qidirish uchun *term in string* ifodasidan foydalanamiz. Agar satr ostisi topilsa ifoda *True* qiymat qaytaradi, aks holda *False* qiymat qaytaradi:

```
string = "hello world"
exist = "hello" in string
print(exist)  # True

exist = "sword" in string
print(exist)  # False
```

4.1.7-rasm

Satrlarni ajratish

*for* sikl operatori yordamida satrning barcha elementlarini ajratib olish mumkin:

```
string = "hello world"
for char in string:
    print(char)
```

4.1.8-rasm

**Nazorat savollari**

1.Pythonda *ord* va *chr* metodlar farqini tushuntiring.

2. Pythonda “Havo isiyapti” matnini har bir harfini alohida satrda ASCII kodini chiqaruvchi dastur tuzing.

3.Pythonda matnning ichidan ma’lum bir so’zni qidiruvchi dastur tuzing.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**10-mavzu: Regulyar ifodalar**

**Mashg’ulot rejasi:**

1.Oddiy(regulyar) ifoda sintaksisi.
2.Qolip(shablon)ning birinchi mosligini topish
3.Qolip(shablon) bilan barcha mosliklarni toppish.
4.Satrda almashtirish

Satr turida aniqlangan *format()* metodi satrlarni formatlash imkonini beradi.

Formatlashda satrda aniqlangan to'ldiruvchilar o'rniga ularning qiymatlarini qo'yish mumkin. Satrda to'ldiruvchilar maxsus "{}" figurali qavslar ichida aniqlanadi.

Parametrlarni nomlash

Formatlanayotgan satrda partametrlarni aniqlash mumkin va ularga *format()* metodi ichida qiymat beriladi:

```
text = "Hello, {first_name}.".format(first_name="Tom")
print(text)  # Hello, Tom.

info = "Name: {name}\t Age: {age}".format(name="Bob", age=23)
print(info)  # Name: Bob     Age: 23
```

4.3.1-rasm

Bundan tashqari, *format()* metodida argumentlar satrdagi parametrlar bilan bir xil nom bilan aniqlanishi shart. Shunday qilib, agar parametr birinchi holatda bo'lgani kabi first_name deb ataladigan bo'lsa, unda qiymati tayinlangan argument ham first_name deb nomlanadi.

O'rinlar bo'yicha parametrlar

Parametrlarni nomlashdan tashqari noldan boshlab raqamlash ham mumkin, bu holda *format()* metodiga parametrlarning faqat qiymatlari uzatiladi va parametrlar kelish tartibi bo'yicha satrga joylashtiriladi:

```
info = "Name: {0}\t Age: {1}".format("Bob", 23)
print(info)     # Name: Bob  Age: 23
```

4.3.2-rasm

Bunday holda parametni satrda bir necha bor foydalanish mumkin:

```
text = "Hello, {0} {0} {0}.".format("Tom")
```

4.3.3-rasm

O’rin almashtirish

O’rin almashtirish va maxsus to’ldiruvchilar satrga formatli qiymalarni berishning yana bir usuli hisoblanadi. Formatlash uchun quyidagi maxsus to’ldiruvchilardan foydalanishimiz mumkin:

- s – satr qo’yish uchun;
- d – butun son qo’yish uchun;
- f – haqiqiy son qo’yish uchun. Bu tur uchun kasr qismidagi xonalar sonini nutqa orqali berish mumkin.
- % – 100 ga ko’paytiradi va foiz belgisini qo’shadi;  e – sonni eksponentsial ko’rinishda chiqaradi.

Maxsus to’ldiruvchining umumiy ko’rinishi quyidagicha:

```
{:to'ldiruvchi}
```

4.3.4-rasm

To’diruvchilarga bog’liq ravishda qo’shimcha parametrlar qo’shish mumkin. Masalan, *float* turidagi son uchun quyidagicha:

```
{:[belgilar_soni][vergul][.kasr_qismdagi_belgilar_soni] to'diruvchi}
```

4.3.5-rasm

*format*() metodi chaqirilganda, unga argument sifatida to'ldiruvchi o'rniga yoziladigan qiymatlar beriladi:

```
welcome = "Hello {:s}"
name = "Tom"
formatted_welcome = welcome.format(name)
print(formatted_welcome)  # Hello Tom
```

4.3.6-rasm

*format()* natijasi sifatida formatlangan yangi satr qaytadi.

**Butun sonlarni formatlash:**

```
source = "{:d} belgilar"
number = 5
target = source.format(number)
print(target)   # 5 belgilar
```

4.3.7-rasm

Agar formatlanayotgan son 999 dan katta bo'lsa, sonning raqamlarni guruhlarga ajratish uchun verguldan foydalanamiz:

```
source = "{:,d} belgilar "
print(source.format(5000))  # 5,000 belgilar
```

4.3.8-rasm

Haqiqiy son, yani, *float* turidagi sonlarning kasr qismidagi xonalar sonini aniq qilib belgilash uchun to’ldiruvchi oldidan, nuqtadan so’ng ularning sonini qo’yishimiz mumkin:

```
number = 23.8589578
print("{:.2f}".format(number))    # 23.86
print("{:.3f}".format(number))    # 23.859
print("{:.4f}".format(number))    # 23.8590
print("{:,.2f}".format(10001.23554))    # 10,001.24
```

4.3.9-rasm

Yana bir parametr harflardagi formatlangan qiymatning minimal kengligini belgilash imkonini beradi:

```
print("{:10.2f}".format(23.8589578))    #      23.86
print("{:8d}".format(25))               #       25
```

4.3.10-rasm

Foizini ko'rsatish uchun "%" kodini ishlatish yaxshiroq:

```
number = .12345
print("{:%}".format(number))          # 12.345000%
print("{:.0%}".format(number))        # 12%
print("{:.1%}".format(number))        # 12.3%
```

4.3.11-rasm

Eksponentsial belgilarda raqamni ko'rsatish uchun "e" to’ldirgichi ishlatiladi:

```
number = 12345.6789
print("{:e}".format(number))        # 1.234568e+04
print("{:.0e}".format(number))      # 1e+04
print("{:.1e}".format(number))      # 1.2e+04
```

4.3.12-rasm

## format() metodisiz formatlash

Umuman olganda, *format()* metodidan foydalanmasdan ham satrlarni formatlash mumkin:

*satr%(paratmetr1, paratmetr 2, paratmetrN)*

bu erda formatlarnishi kerak bo’lgan satrda barcha to’ldiruvchilar (% belgili to’ldiruvchi bundan mustasno) figurali qavslarga olinmasdan yoziladi va satrdan keyin % belgisi qo’yiladi, undan so’ng qavs ichida mos argumentlar ketma-ketligi yoziladi. Foiz belgisi to’diruvchilar old qismida ko'rsatiladi:

```
info = "Ism: %s \t Yosh: %d" % ("Tom", 35)
print(info)   # Ism: Tom     Yosh: 35
```

4.3.13-rasm

Formatlanayotgan satr va argumentlar ro’yxati orasidagi % belgisi – operator vazifasini bajaradi va natija sifatida formatlangan yangi sartni qaytaradi.

Bundan tashqari, raqamlarni formatlash usullari ham qo'llaniladi:

```
number = 23.8589578
print("%0.2f  - %e" % (number, number))   # 23.86  -2.385896e+01
```

4.3.14-rasm

**Nazorat savollari**

1.Pythonda *format* metodini tushuntiring.

2. Pythonda *format* metodisiz qanday qilib matnlarni chop etish mumkinligini tushuntiring.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**11-mavzu: Ro'yxatlar bilan ishlash**

**Mashg’ulot rejasi:**

1.Ro‘yxat yaratish.
2.Ro'yxatlar ustida amallar bajarish.
3.Ko'p o'lchovli ro'yxatlar
4.Ro‘yxat elementlarini takrorlash.

Ro’yxat (*list*) bu elementlar to’plami yoki ketma-ketligini saqlash uchun mo’ljallangan berilganlar turini ifodalaydi. Ro’yxatlarni hosil qilish uchun kvadrat

qavs([]) ichida uning barcha elementlari vergul bilan ajratilgan holda keltiriladi. Ko’pincha, boshqa tillarda shunga o’xshash berilganlar turini massiv deb ataladi.

Masalan quyida sonlar ro’yxatini aniqlaymiz:

*sonlar = [1, 2, 3, 4, 5]*

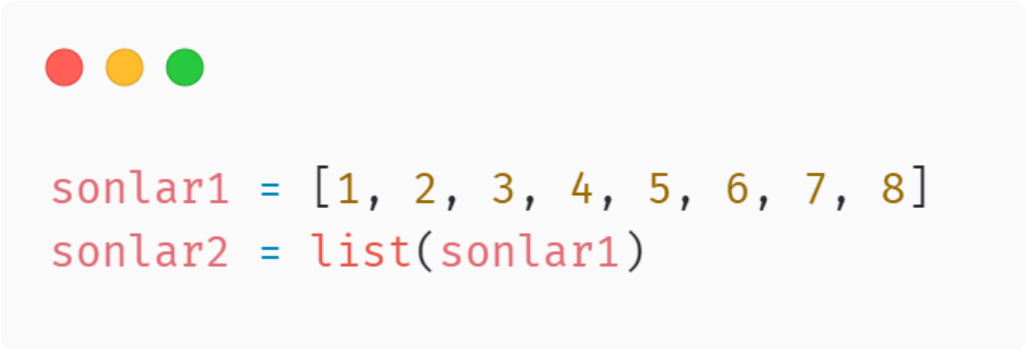
Ro’yxatlarni hosil qilish uchun *list()* konstruktoridan ham foydalaniladi:

*sonlar1 = [ ]*

*sonlar2 = list()*

Yuqoridagi ikkita ro’yxat o’xshash bo’lib, ular bo’sh ro’yxatni aniqlayapti.

*list*() konstruktori ro’yxat hosil qilish uchun parametr sifatida boshqa ro’yxatni ham qabul qilishi mumkin:




2.1.1-rasm

Ro’yxat elementiga murojaatni amalga oshirish uchun uning ro’yxatdagi tartib raqamini ifodalovchi indeksi orqali amalga oshiriladi. Indekslar noldan boshlanadi. Ya`ni ikkinchi elementning indeksi 1 ga teng bo’ladi. Ro’yxat elementlariga oxiridan (teskari tartibda) murojaatni amalga oshirish uchun manfiy indekslardan foydalaniladi. Ro’yxatning eng oxirgi elementiga murojaat uchun -1, oxiridan oldingi element uchun -2 indekslari mos keladi va h.k.. Ro’yxatdagi elementning qiymatini o’zgartirish uchun shu element indeksi orqali unga yangi qiymat yuklash orqali amalga oshiriladi:

```python
sonlar = [1, 2, 3, 4, 5, 6, 7, 8]

print(sonlar[0])    #1
print(sonlar[1])    #2
print(sonlar[-1])   #8
print(sonlar[-2])   #7

sonlar[0] = 100
print(sonlar[0])    #100
```

2.1.2-rasm

**Ro’yxatni butun songa ko’paytirish.** Python dasturlash tilida Ro’yxatni butun songa ko’paytirish amali ham kiritilgan bo’lib, odatda bunday amallar ro’yxatdagi barcha elementlarga ayni bir xil qiymatlarni yuklash uchun qo’llaniladi. Masalan, 6 ta elementining barcha qiymatlari 0 ga teng bo’lgan ro’yxatni aniqlash uchun quyidagicha kod yoziladi:

```python
son = [0] * 6
print(son) # [0, 0, 0, 0, 0, 0]
```

2.1.3-rasm

Bundan tashqari, sonlar ketma-ketligining ro’yxatini *range()* funksiyasidan foydalanib hosil qilish ham mumkin bo’lib, u uchta shaklga ega:

- *range(end)* – 0 dan *end* gacha (*end* kirmaydi) bo’lgan sonlar ketma-ketligi hosil qilinadi.
- *range(start, end)* – *start* dan *end* gacha (*end* kirmaydi) bo’lgan sonlar ketma-ketligi hosil qilinadi.

- *range(start, end, step)* – *start* dan *end* gacha (*end* kirmaydi) bo’lgan sonlar ketma-ketligi *step* qadam bilan hosil qilinadi.

Quyidagi ikkita ifoda orqali ayni bir xil bo’lgan ro’yxatlar aniqlangan bo’lib, ikkinchi ro’yxatni hosil qilishda *range* funksiyasidan foydalanish orqali kod hajmi bir muncha qisqartirilgan:

```
nums = list(range(6))
print(nums) #[0, 1, 2, 3, 4, 5]

nums = list(range(3,6))
print(nums) #[3, 4, 5]

nums = list(range(2,6,2))
print(nums) #[1, 3, 5]
```

2.1.4-rasm

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

2.1.5-rasm

Ro’yxatlar bir turdagi ob`ektlardan tashkil topishi shart emas. Undagi har bir element turli xil toifaga ega bo’lishi mumkin:

```
objs = [1.2, True , 100, 'Satr', object]
print(objs) #[1.2, True, 100, 'Satr', <class 'object'>]
```

2.1.6-rasm

**Elementlarni birma-bir ko’rib chiqish**. Ro’yxat elementlarini birma-bir ko’rib chiqish uchun *for* yoki *while* takrorlash operatorlaridan foydalanish mumkin. *for* orqali birma-bir ko’rib chiqish:

```
poytaxtlar = ['London', 'Parij', 'Moskva', 'Tashkent']
for poytaxt in poytaxtlar:
    print(poytaxt)
```

2.1.7-rasm

*while* orqali birma-bir ko’rib chiqish:

```
poytaxtlar = ['London', 'Parij', 'Moskva', 'Tashkent']
i = 0
while i < len(poytaxtlar):
    print(poytaxtlar[i])
    i += 1
```

2.1.8-rasm

Bu erda *while* sikli orqali birma-bir ko’rib chiqishda *len* funksiyasidan foydalanilgan bo’lib, uning yordamida ro’yxat uzunligi olinadi. Bundan tashqari sanagich vazifasini bajaruvchi *i* o’zgaruvchisi bilan ro’yxatdagi barcha elementlarga murojaat amalga oshirilgan.

**Ro’yxatlarni taqqoslash**. Ikkita ro’yxat bir xil elementlardan tashkil topgan bo’lsa, bunday ro’yxatlar teng hisoblanadi.

```
nums1 = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
nums2 = list(range(10))
if nums1 == nums2:
    print("Bu ro`yxatlar aynan teng.")
else:
    print("Bu ro`yxatlar teng emas.")
```

2.1.9-rasm

Bu holatda ikkila ro’yxat teng hisoblanadi.

**Nazorat savollari**

1.Ro’yxat nima va Pythonda ro’yxatlar qanday yaratilishi mumkin?

2. *range()* funksiyasining ishlashini misollar bilan tushuntiring.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**12-mavzu: Ro'yxat funksiyalari bilan ishlash**

**Mavzu rejasi:**

1.Ro’yxatlar generatorlari va generator  ifodalari.
2.Map(), zip(), filter() va reduce() funksiyalari.
3.Ro'yxat elementlarini qo'shish va o'chirish.
4.Ro'yxatdagi elementni topish va ro'yxatdagi qiymatlar haqida ma'lumot olish.
5.Ro‘yxatni aylantirish va aralashtirish

**Ro’yxatlar bilan ishlashda qo’llaniladigan funksiyalar va metodlar**.

Ro’yxatlar bilan ishlashda qo’llaniladigan bir nechta metodlar mavjud bo’lib, ularning eng asosiylari quyida keltirilgan:

- ***append(item)***: ro’yxat oxiriga *item* elementini qo’shish;
- ***insert(index, item)***: Ro’yxatga *index* indeksi bo’yicha *item* elementini qo’shish;
- ***remove(item):*** ro’yxatdan *item* elementini o’chirish. Ushbu metod ro’yxatdagi birinchi uchragan *item* elementini o’chiradi. Agar bunday element ro’yxatda mavjud bo’lmasa *ValueError* istisno holati ro’y beradi;
- ***clear():*** ro’yxatni tozalash, ya`ni ro’yxatdagi barcha elementlarni o’chirish;
- ***index(item)***: ro’yxatdagi *item* elementining joylashgan indeksini qiymat sifatida qaytaradi. Agar bunday element ro’yxatda mavjud bo’lmasa, u holda *ValueError* istisno holati ro’y beradi;
- ***pop([index]):*** ro’yxatdan *index* indeksi bo’yicha elementni o’chiradi va qiymat sifatida qaytaradi. Agar indeks ko’rsatilmasa ro’yxatdan oxirgi elementni o’chiradi va qiymat sifatida qaytaradi. Bundan tashqari agar ro’yxat bo’sh bo’lsa, u holda *ValueError* istisno holati ro’y beradi;
- ***count(item):*** ro’yxatdagi *item* elementlar sonini qiymat sifatida qaytaradi;
- ***sort([key]):*** ro’yxat elementlarini tartiblaydi. Kelishuv bo’yicha, ro’yxat elementlarini o’sish bo’yicha tartiblaydi. *key* parametri orqali tartiblash funksiyasini (mezonini) berish mumkin;
- ***reverse():*** ro’yxat elementlarini teskari tartibda joylashtirish uchun qo’llaniladi.

Bundan tashqari, Python ro’yxatlar bilan ishlashda qo’llaniladigan bir nechta standart funksiyalarni ham o’z ichiga qamrab olgan:

- ***len(list):*** ro’yxat uzunligini (elementlari sonini) qiymat sifatida qaytaradi;
- ***sorted(list,[key]):*** tartiblangan ro’yxatni qiymat sifatida qaytaradi;

```python
users = ["Tom", "Bob"]

# ro`yxat oxiriga element qo`shish
users.append("Alice")  # ["Tom", "Bob", "Alice"]
# ro`yxatdagi ikkinchi o`ringa element qo`shish
users.insert(1, "Bill")  # ["Tom", "Bill", "Bob", "Alice"]
# element indeksini olish
i = users.index("Tom")
# indeks bo`yicha elementni ro`yxatdan o`chirish
removed_item = users.pop(i)  # ["Bill", "Bob", "Alice"]
# ro`yxatdagi oxirgi qiymatni olish
last_user = users[-1]
# ro`yxatdan oxirgi elementni o`chirish
users.remove(last_user)  # ["Bill", "Bob"]

print(users)
# ro`yxatning barcha elementlarini o`chirish users.clear()
```

2.1.10-rasm

**Elementni ro'yxatda mavjudligini tekshirish**. Odatda, *index*, *remove* kabi metodlardan foydalanilganda, ularning parametrlarida ko'rsatilgan element ro'yxatda mavjud bo'lmasa, istisno holati yuzaga keladi. Bunday holatlarning oldini olish uchun, oldin ushbu elementning ro'yxatda mavjudligini tekshirish kerak bo'ladi. Buning uchun *in* kalit so'zidan foydalaniladi:

```python
companies = ["Microsoft", "Google", "Oracle", "Apple"] item = "Oracle"
 # o'chirish kerak bo'lgan element
if item in companies:
    companies.remove(item)

print(companies) # ['Microsoft', 'Google', 'Apple']
```

2.1.11-rasm

Yuqoridagi holatda, agar ro'yxatda *item* elementi mavjud bo'lsa, *item in companies* ifodasi *True* qiymat qaytaradi. Shuning uchun *if* shart ifodasi agar shu

element mavjud bo’lsagina navbatdagi amallarni bajaradi, natijada istisno holatining oldi olinadi.

***count()* metodi**. Ushbu metod orqali ro’yxatda biror element nechi marta qatnashganligi topiladi. Masalan:

```
nomlar = ["Anvar","Sobir","Sobir","Qosim"]
nom_soni = nomlar.count("Sobir")
print(nom_soni) #2
```

2.1.12-rasm

**Nazorat savollari**

1. *count()* funksiyasining ishlashini misollar bilan tushuntiring.
2.Elementni ro’yhatda mavjudligini qanday tekshirish mumkin?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

13-mavzu: **Ro'yxat elementlarini tasodifiy tanlash**

**Mashg’ulot rejasi:**

1.Elementlarni tasodifiy tanlash.
2.Ro'yxatni saralash.
3.Ro'yxatni sonlar bilan to'ldirish.
4.Ro'yxatni satrga aylantirish

**Tartiblash**. Ro’yxatdagi elementlarni o’sib borish bo’yicha tartiblash uchun ***sort()*** metodi qo’llaniladi:

```
nomlar = ["Anvar","Sobir","Sobir","Qosim"]
nomlar.sort()
print(nomlar)  # ['Anvar', 'Qosim', 'Sobir', 'Sobir']
```

2.1.13-rasm

Agar teskari tartibda tartiblash kerak bo’lsa, ***sort()*** metodidan keyin ***reverse()*** metodidan foydalanish kifoya qiladi:

```
nomlar = ["Anvar","Sobir","Sobir","Qosim"]
nomlar.sort()
nomlar.reverse()
#nomlar.sort(reverse=True) deb ham ishlatish mumkin
print(nomlar)  # ['Sobir', 'Sobir', 'Qosim', 'Anvar']
```

2.1.14-rasm

Shuni alohida ta`kidlash lozimki, tartiblashda ob`ektlar taqqoslanadi. Ob`ekt sifatida son kelsa muammo yo’q, o’sish yoki kamayib borish bo’yicha tartiblanadi. Lekin ob`ekt sifatida satr kelsa, u holda ular mos belgilari bo’yicha taqqoslanadi. Taqqoslashda belgilarning ASCII kodlari taqqoslanadi. Shuning uchun har qanday

katta registrga ega lotin yozuvidagi belgi kichik registrdagi lotin yozuvidagi belgidan kichik bo'ladi, masalan: "Abc" < "abc":

```
nomlar = ["anvar","Sobir","sobir","Qosim","tolib"]
nomlar.sort()
print(nomlar) # ['Qosim', 'Sobir', 'anvar', 'sobir', 'tolib']
```

2.1.15-rasm

Tartiblashda ***sort()*** metodi o'rniga standart ***sorted()*** funksiyasidan ham foydalanish mumkin bo'lib, uning quyidagi ikkita shakli mavjud:

```
nomlar = ["anvar","Sobir","sobir","Qosim","tolib"]
sorted_nomlar = sorted(nomlar, key = str.lower)
print(sorted_nomlar)
# ['anvar', 'Qosim', 'Sobir', 'sobir', 'tolib']
```

2.1.16-rasm

***sorted()*** funksiyasidan foydalanilganda qiymat sifatida ushbu funksiya tartiblangan elementlardan tashkil topgan yangi ro'yxatni qaytaradi, tartiblanayotgan ro'yxat esa o'zgarishsiz qoladi.

**Minimal va maksimal qiymatlar**. Python dasturlash tilida ***max***, ***min*** deb nomlanuvchi mos ravishda ro'yxatdan eng maksimal va eng minimal qiymatlarni topish uchun mo'ljallangan standart funksiyalari mavjud.

```
sonlar = [12, 45, 23, -35, 2]
print(min(sonlar))  # -35
print(max(sonlar))  # 45
```

2.1.17-rasm

**Ro’yxatlarni ko’chirish**. Ro’yxat – o’zgaruvchan (mutable) turga mansub bo’lib, agar ikkita o’zgaruvchi ayni bir ro’yxatga murojaat qilayotgan bo’lsa, u holda birining o’zgarishi ikkinchisiga ham ta`sir qiladi:

```
vil1 = ["Toshkent", "Xorazm", ]
vil2 = vil1 vil2.append('Buxoro')
print(vil1)  # ['Toshkent', 'Xorazm', 'Buxoro']
print(vil2)  # ['Toshkent', 'Xorazm', 'Buxoro']
```

2.1.18-rasm

Bu misolda har ikkala *vil1* va *vil2* o’zgaruvchilar yordamida ayni bir ro’yxatga murojaat bo’lgan. Shuning uchun *vil2* o’zgaruvchisi orqali ro’yxatga yangi element qo’shilganda mos ravishda *vil1* ham o’zgargan. Bu holat yuzaki ko’chirish (*shallow copy*) deyiladi. Albatta ro’yxatlarni ko’chirganda ikkita alohida ro’yxat hosil qiladigan tarzda ham ko’chirish (*deep copy*) mumkin. Buning uchun ichki ***copy*** moduldagi *deepcopy()* metodidan foydalaniladi:

```
import copy
vil1 = ["Toshkent", "Xorazm", ]
vil2 = copy.deepcopy(vil1)
vil2.append('Buxoro')
# ikkita alohida ro`yxat hosil bo`ldi
print(vil1)  # ['Toshkent', 'Xorazm']
print(vil2)  # ['Toshkent', 'Xorazm', 'Buxoro']
```

2.1.19-rasm

**Ro’yxat qismini ko’chirish**. Agar butun bir ro’yxatni emas, balki uning bir qismini ko’chirish zarur bo’lsa, u holda maxsus sintaksisdan foydalaniladi. Ushbu sistaksis quyidagi shakllarda  qo’llaniladi: ***list(:end)*** - *end* parametri orqali ro’yxatning qaysi elementigacha ko’chirish

kerakligini bildiruvchi indeks nomeri beriladi; ***list(start:end)*** – *start*, *end* parametrlar orqali ro'yxatning *start* dan *end* gacha

bo'lgan elementlarini ko'chirish kerakligini bildiruvchi indeks nomerlari beriladi; ***list(start:end:step)*** – *start*, *end*, *step* parametrlar orqali ro'yxatning *start* dan *end* gacha bo'lgan elementlarini *step* qadam bilan ko'chirish kerakligini bildiruvchi qiymatlar beriladi. *step* parametrning kelishuv bo'yicha qiymati 1 ga tengdir.

```
vil = ["Toshkent", "Xorazm", 'Buxoro', 'Navoi', 'Jizzax']
vil1 = vil[:2]
print(vil1)  # ['Toshkent', 'Xorazm']
vil2 = vil[2:4]
print(vil2)  # ['Buxoro', 'Navoi']
vil3 = vil[1:5:2]
print(vil3)  # ['Xorazm', 'Navoi']
```

2.1.20-rasm

**Ro'yxatlarni birlashtirish**. Ro'yxatlarni birlashtirish uchun (+) amali qo'llaniladi:

```
vil1 = ["Toshkent", "Xorazm", 'Buxoro']
vil2 = ['Navoi', 'Jizzax']
vil = vil1 + vil2
print(vil)
```

2.1.21-rasm

**Ichma – ich joylashgan ro'yxatlar**. Ro'yxat elementlari son, satr kabi oddiy turdagi qiymatlargina bo'lib qolmay, balki ro'yxatni ham ifodalashi mumkin. Odatda bunday ro'yxatlardan jadvallar bilan ishlashda ko'p foydalaniladi. Ushbu holatda tashqi ro'yxatning har bir elementi jadvaldagi bitta qatorni ifodalovchi ro'yxatdan iborat bo'ladi:

```python
ishchilar = [
    ["Tolib", 33],
    ["Akmal", 30],
    ["Botir", 27] ]

print(ishchilar[0])  # ["Tolib", 33]
print(ishchilar[0][0])  # Tolib
print(ishchilar[0][1])  # 33
```

2.1.22-rasm

Bu erda ichki ro’yxatni elementiga murojaat qilish uchun [][] indekslar juftligidan foydalanilgan. Xususan, *ishchilar[0][1]* – birinchi ichki ro’yxatning ikkinchi elementiga murojaat bo’lgan.

Ro’yxatga elementlarni qo’shish, o’chirish, o’zgartirish kabi jarayonlar oddiy ro’yxatlardagi kabi amalga oshiriladi:

```
ishchilar = [
    ["Tolib", 33],
    ["Akmal", 30],
    ["Botir", 27] ]

print(ishchilar[0])  # ["Tolib", 33]
print(ishchilar[0][0])  # Tolib
print(ishchilar[0][1])  # 33
# ro`yxat yaratish
ishchi = list()
ishchi.append("Rustam")
ishchi.append(21)
# Tashqi ro`yxatga yaratilgan ro`yxatni qo`shish
ishchilar.append(ishchi)
print(ishchilar[-1])  # ["Rustam", 21]
# Tashqi ro`yxatning oxirgi elementiga element qo`shish
ishchilar[-1].append("+998909737066")
print(ishchilar[-1])  # ['Rustam', 21, '+998909737066']
# tashqi ro`yxatning oxirgi elementining oxirgi elementini
# o`chirish
ishchilar[-1].pop()
print(ishchilar[-1])  # ["Rustam", 21]
# tashqi ro`yxatning oxirgi elementini o`chirish
ishchilar.pop(-1)
# birinchi elementni o`zgartirish
ishchilar[0] = ["Sobir", 18]
print(ishchilar) # [['Sam', 18],['Akmal', 30],['Botir', 27]]
```

2.1.23-rasm

Murakkab ro’yxatlar elementlariga murojaat ichma – ich joylashgan takrorlash operatorlari orqali amalga oshirilishi mumkin:

```python
ishchilar = [
    ["Tolib", 33],
    ["Akmal", 30],
    ["Botir", 27] ]
 for ishchi in ishchilar:
    for i in ishchi:
        print(i, end="|")
# Konsolga quyidagi ma`lumotlar chiqariladi
# Tolib|33|Akmal|30|Botir|27|
```

2.1.24-rasm

**Nazorat savollari**

1. Ro’xatning oxirgi elementini o’chirishning bir nechta usullarini misollar bilan tushuntiring.
2.Berilgan N(N>10) ta elementli ro’yxatni kamayish tartibida tartiblovchi, so’ngra uning 4, 7 va 9-elementlarini o’chiruvchi dastur tuzing?
3.Sizga A ro’yxat berilgan. *deepcopy()* metodi va ro’yxatni ko’chirishdan foydalanib, A ro’yxatning juft indexli elementlaridan iborat B ro’yxatni yasang va uni o’sish tartibida tartiblab, A ning dastlabki va B ning oxirgi qiymatini chiqaring.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**14-mavzu: Kortej va to’plamlar bilan ishlash**

**Mashg’ulot rejasi:**

1.Kortej funksiyalari bilan ishlash.
2.To’plamlar.
3.Itertools moduli.
4.Ketma-ket elementlarni filtirlash

Kortej (tuple) elementlar ketma-ketligini ifodalovchi, ko’p jihatlari bo’yicha ro’yxatga o’xshaydigan, lekin undan farqli ravishda o’zgarmaydigan (immutable) tur hisoblanadi. Shuning uchun kortejga yangi element qo’shish, undan elementni o’chirish yoki o’zgartirish kiritishga ruxsat etilmaydi.

Kortejni hosil qilish uchun oddiy qavs “(, )” dan foydalanib, unda elementlar vergul bilan ajratilgan tarzda joylashtiriladi:

```
user = ("Akmal", 21)
print(user) # ('Akmal', 21)
```

2.2.1-rasm

Bundan tashqari kortejni aniqlash uchun elementlar ketma – ketligi vergul bilan ajratilgan holda oddiy qavslarsiz ham amalga oshirsa bo’ladi:

```
user = ("Akmal", 21)
print(user) # ('Akmal', 21)
```

2.2.2-rasm

Shuni alohida ta`kidlash kerakki, kortej faqat bitta elementdan tashkil topsa ham vergul ishlatiladi:

```
user = "Akmal",
print(user) # ('Akmal',)
```

2.2.3-rasm

Ro’yxat asosida kortej hosil qilish uchun maxsus ***tuple*** funksiyasidan foydalaniladi va uning argumentiga qiymat sifatida ro’yxat beriladi:

```
user_list = ["Yusuf", 'Tolib', 'Rustam']
user_tuple = tuple(user_list)
print(user_tuple)  # ('Yusuf', 'Tolib', 'Rustam')
```

2.2.4-rasm

Kortej elementlariga murojaat xuddi ro’yxatlardagi kabi indekslari orqali amalga oshiriladi. Indekslar kortej boshiga nisbatan 0 dan, kortej oxiriga nisbatan -1 dan boshlanadi:

```
users = ("Yusuf", 'Qodir', 'Erkin', 'Oybek')
print(users[0]) # Yusuf
print(users[2]) # Erkin
print(users[-1])  # Oybek
print(users[1:3])  # ('Qodir', 'Erkin')
```

2.2.5-rasm

Kortej o’zgarmaydigan tur bo’lganligi sababli uning elementini o’zgartirib bo’lmaydi. Masalan: *users[0] = “Rahim”* kabi kod yozilsa, Python interpretatori xatolik to’g’risida xabar chiqaradi.

Kortej elementlarini, ularning soniga mos o’zgaruvchilarga birdaniga yuklash ham mumkin:

```
users = ("Yusuf", 'Qodir', 'Erkin',)
a, b, c = users
print(a)  # Yusuf
print(b)  # Qodir
print(c)  # Erkin
```

2.2.6-rasm

Kortejlarning bunday xususiyati ayniqsa funksiyalar bilan ishlashda juda foydali hisoblanadi. Masalan, funksiya natija sifatida birdan ortiq qiymat qaytarsa, aslida kortej turidagi bitta qiymat qaytarayotgan bo’ladi. Funksiyadan qaytgan qiymatlarni bir nechta o’zgaruvchilarga yuklash orqali, alohida bir - biriga bog’liq bo’lmagan qiymatlarga ega bo’linadi:

```
def get_user():
    name = "Yusuf"
    age = 33
    is_married = True
    return name, age, is_married

user = get_user()
print(user[0])  # Yusuf
print(user[1])  # 33
print(user[2])  # True

# yoki alohida o`zgaruvchilarga yuklanadi
name, age, ismarried = get_user()
print(name)  # Yusuf
print(age)  # 33
print(ismarried)  # True
```

2.2.7-rasm

***len()*** funksiyasi orqali kortejning uzunligi (elementlari soni) topiladi:

```python
user = ("Erkin", 30, True)
print(len(user))  # 3
```

2.2.8-rasm

Kortej elementlariga *for* va *while* takrorlash operatorlari orqali murojaat quyidagicha amalga oshiriladi:

*for* orqali

```python
user = ("Erkin", 30, True)
 for u in user:
    print(u)
```

2.2.9-rasm

*while* orqali

```python
user = ("Erkin", 30, True)
i=0
while i < len(user):
  print(user[i])
  i +=1
```

2.2.10-rasm

Kortejda biror elementning mavjudligini xuddi ro'yxatlardagi kabi ***in*** operatoridan foydalanib amalga oshiriladi:

```
user = ("Erkin", 30, True)
if 'Erkin' in user:
    print("Foydalanuvchining ismi Erkin")
```

2.2.11-rasm

**Murakkab kortejlar**. Kortej o’z ichiga boshqa kortejlarni element sifatida qamrab olsa, bunday kortejlar murakkab kortejlar hisoblanadi:

```
davlatlar = (("O`zbekiston", 33.8,(
        ("Toshkent", 2.65),
        ("Samarqand", 1.1),
        ("Urganch", 0.223))
    ),
    ("Qozog`iston", 17.5, (
        ("Nur Sultan", 1.1),
        ("Olmaota", 2.3)
        )
    ))
for davlat in davlatlar:
    nomi,aholi_soni,shaharlar = davlat
    print(nomi,'-',aholi_soni)
    print("Katta shaharlari:")
    for shahar in shaharlar:
        print(shahar[0],'-',shahar[1])
```

2.2.12-rasm

Bu erda *davlatlar* korteji ikki elementdan tashkil topgan bo’lib, ular ham kortejlardir va davlat nomi, aholisi, katta shaharlari to’g’risidagi ma`lumotlarni ifodalaydi. O’z navbatida shaharlar ham yana kortejlardan tashkil topgan bo’lib, ularning har bir elementi shahar nomi va aholisini ifodalaydigan kortejdan tashkil topgan. Xullas, jami ichma-ich joylashgan 4 ta kortejdan iborat berilganlarning murakkab strukturasiga misol keltirilgan.

**Nazorat savollari**

1.Kortej nima?

2. *Kortej*ning ro’yxatdan farqini misollar bilan birga tushuntiring.

3.Ro’xatning oxirgi elementini o’chirishning bir nechta usullarini misollar bilan tushuntiring.

4.Bir nechta murakkab Kortejlarni yarating va uning elementlarini ekranga chop etuvchi dastur yozing.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

## 15-mavzu: Lug’at yaratish va ular bilan ishlash

**Mashg’ulot rejasi:**

1.Lug'at yaratish.
2.Lug'atlar ustida amallar bajarish.
3.Lug'at elementlarini takrorlash.
4.Lug'atlar bilan ishlash usullari.
5.Lug'at generatorlari.

Python dasturlash tilida ro’yxatlar va kortejlar bilan bir qatorda lug’atlar (dictinary) deb nomlanuvchi berilganlarning ichki tuzilmasi mavjud. Lug’atlar ham xuddi ro’yxatlar kabi elementlar to’plamini saqlaydi. Lug’atdagi har bir element unikal kalitga ega bo’ladi va unga biror bir qiymat bog’lanadi.

Lug’at quyidagicha sistaksis bo’yicha aniqlanadi:

*dictionary = { kalit1:qiymat1, kalit2:qiymat2, ....}*

Quyida lug’atlarga misol keltirilgan:

```
users = {1: "Tom", 2: "Bob", 3: "Bill"}

elements = {"Au": "Oltin", "Fe": "Temir", "H": "Vodorod","O": "Kislorod"}
```

2.3.1-rasm

Bu yerda *users* ro’yxatida kalit sifatida son, qiymat sifatida satr qo’llanilgan. *element* ro’yxatida esa qiymat sifatida ham kalit sifatida ham satr ishlatilgan.

Lekin kalitlar va qiymatlar bir turga mansub bo’lishi shart emas. Ular har xil turdagi qiymatlar bo’lishi mumkin:

```
objects = {1: "Tom", "2": True, 3: 100.6}
```

2.3.2-rasm

**Ro’yxatlar yordamida lug’at xosil qilish**. Lug’atlar tuzilmaviy jihatidan ro’yxatlarga o’xshamasada, lekin ba`zi bir maxsus ro’yxatlar asosida *dict()* funkuyasi orqali ro’yxatlar hosil qilish mumkin. Buning uchun ro’yxat o’z navbatida ro’yxatlar to’plamidan tashkil topgan bo’lishi kerak. Ichki ro’yxatlar ikkita elementlardan tashkil topishi shart bo’lib, mos ravishda birinchi element kalitga, ikkinchi element qiymatga akslantiriladi:

```
users_list = [["909837022", "Tolib"],
    ["909939343", "Bobur"],     ["903943493", "Alibek"] ]
users_dict = dict(users_list)
print(users_dict)
    # {'909837022': 'Tolib', '909939343':'Bobur', '903943493': 'Alibek'}
```

2.3.3-rasm

Xuddi shu tarzda kortejlarni ham lug'atlarga aylantirish mumkin. Buning uchun ikki o'lchamli kortejning ichki kortejlari o'z navbatida ikkitadan elementdan tashkil topgan bo'lishi shart:

```
users_tuple = ( ("909837022", "Tolib"),
    ("909939343", "Bobur"),     ("903943493", "Alibek") )
users_dict = dict(users_tuple)
print(users_dict)
# {'909837022': 'Tolib', '909939343':'Bobur', '903943493': 'Alibek'}
```

2.3.4-rasm

**Lug'at metodlari.**

**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.

**Lug'at elementini o'zgartirish.** Lug'at elementiga murojaat qilish uning kaliti yordamida amalga oshiriladi:

*dictionary*[*kalit*]

Masalan lug'at elementiga murojaat qilish va uni o'zgartirish quyidagicha amalga oshiriladi:

```
users = {
    "Bir": "Tolib",
    "Ikki": "Bobur",
    "Uch": "Alisher"  }
# Lug'atning "Bir" kalitli elementiga murojaat uchun
print(users["Bir"])  # Tolib
# Lug'atdagi "Uch" kalitli element qiymatini o'zgartiramiz
users["Uch"] = "Baxtiyor"
print(users["Uch"])  # Baxtiyor
```

2.3.5-rasm

Lug'at elementiga kaliti orqali qiymat berganda shunday kalit lug'atda mavjud bo'lmasa, u holda lug'atga yangi element qo'shiladi. Masalan, yuqoridagi misolda *users*[*"To'rt"*] = *"Ibrohim"* tarzida yangi element qo'shishimiz mumkin, Chunki lug'atda *"To'rt"* kalitli element mavjud emas.

Lekin, lug’atda mavjud bo’lmagan kalit orqali uning elementiga murojaat qilinganda, Python interpretatori *KeyError* turidagi istisno xatoligi yuzaga kelganligi haqida xabar chiqaradi. Masalan, yuqoridagi misol uchun *user = users*[*“Besh”*] kabi ishlatsak xatolik ro`y beradi. Bunday istisno xalotlarning oldini olish uchun Python dasturlash tilida ***Kalit in Lug’at*** ifodasidan foydalaniladi. Ushbu ifoda agarda shunday kalitli element lug’atda mavjud bo’lsa *True* qiymat, aks holda *False* qiymat qaytaradi, masalan:

```
bahoDict = {"A": 5, "B": 4, "C": 3}
key = "D"
if key in bahoDict:
    baho = bahoDict[key]
    print(baho)
else:
    print("Element topilmadi")
# Javob: Element topilmadi
```

2.3.6-rasm

Shu bilan birga, lug’atning biror elementini olish uchun *get* metodidan ham foydalanish mumkin bo’lib u ikki xil shaklda qo’llaniladi:

- *get*(*key*) – lug’atning *key* kalitli elementni qaytaradi. Agar lug’atda *key* kalitli element mavjud bo’lmasa *None* qiymati qaytariladi.
- *get(key, default)* - lug’atning *key* kalitli elementni qaytaradi. Agar lug’atda *key* kalitli element mavjud bo’lmasa *default* qiymati qaytariladi.

Masalan:

```python
bahoDict = {"A": 5, "B": 4, "C": 3}
key = "A"
baho = bahoDict.get(key)
print(baho)  # 5
# yoki
key = "D"
baho = bahoDict.get(key, "Noma'lum qiymat")
print(baho)  # Noma'lum qiymat
```

2.3.7-rasm

**Lug’atdan elementni o’chirish.** Lug’atdan kalit orqali elementni o’chirish uchun *del* operatoridan foydalaniladi:

```python
bahoDict = {"A": 5, "B": 4, "C": 3, "D": 2}
print(bahoDict) # {'A': 5, 'B': 4, 'C': 3, 'D': 2}

del bahoDict["C"]
print(bahoDict) # {'A': 5, 'B': 4, 'D': 2}
```

2.3.8-rasm

Shuni alohida ta‟kidlash lozimki, agar lug’atda bunday kalit mavjud bo`lmasa *KeyError* istisno xatoligi yuzaga keladi. Ushbu xatolikni oldini olish uchun dastlab bunday kalit lug’atda bor yoki yo’qligini tekshirish tavsiya qilinadi:

```python
bahoDict = {"A": 5, "B": 4, "C": 3, "D": 2}
baho = "A"
if baho in bahoDict:
  son = bahoDict[baho]
  del bahoDict[baho]
  print(son, "o'chirildi")
else:
  print("Element topilmadi")
# Javob: 5 o'chirildi
```

2.3.9-rasm

O’chirishning boshqa bir usuli – *pop*() metodi orqali amalga oshiriladi. U ikki xil shaklda qo’llaniladi:

*pop*(*key*) – *key* kaliti bo’yicha elementni o’chiradi va qiymat sifatida o’chirilgan elementni qaytaradi. Agar berilgan kalit bo’yicha element topilmasa, *KeyError* istisno holati yuzaga keladi; *pop*(*key, default*) – *key* kaliti bo’yicha elementni o’chiradi va qiymat sifatida o’chirilgan elementni qaytaradi. Agar berilgan kalit bo’yicha element topilmasa, *default* qiymati qaytariladi.

```python
bahoDict = {"A": 5, "B": 4, "C": 3, "D": 2}
key = "A"
baho = bahoDict.pop(key)
print(baho)  # 5
# ikkinchi marta yana shu kalit bo`yicha o`chirishga urinamiz
baho2 = bahoDict.pop(key, "Bunday baho mavjud emas!")
print(baho2)  # Bunday baho mavjud emas!
```

2.3.10-rasm

Agar lug’atdagi barcha elementlarni o’chirish talab qilinsa, *clear*() metodidan foydalanish mumkin:

```
bahoDict = {"A": 5, "B": 4, "C": 3, "D": 2}
# Lug'atning barcha elementlarini ekranga chiqaramiz
print(bahoDict)  # {'A': 5, 'B': 4, 'C': 3, 'D': 2}
bahoDict.clear()
# clear metodini qo'llagandan so'ng yana
# lug'atning barcha elementlarini ekranga chiqaramiz
print(bahoDict)  # {}
```

2.3.11-rasm

**Lug’atlarni ko’chirish va birlashtirish.** Lug’atlarni ko’chirish uchun *copy()* metodidan foydalanilib, qiymat sifatida ushbu lug’atning elementlaridan tashkil topgan boshqa lug’at hosil qilinadi, masalan:

```
l = {"ismi": "Sardor", "yoshi": 34}
l2 = l.copy()
print(l)  # {'ismi': 'Sardor', 'yoshi': 34}
print(l2)  # {'ismi': 'Sardor', 'yoshi': 34}
```

2.3.12-rasm

Lug’atlarni birlashtirish uchun *update()* metodidan foydalaniladi:

```
l1 = {"ismi": "Sardor", "yoshi": 34}
l2 = {"kursi": 1, "yo`nalishi": "IAT"}
l1.update(l2)
print(l1)  # {'ismi': 'Sardor', 'yoshi': 34, 'kursi': 1,'yo`nalishi': 'IAT'}
print(l2)  # {'ismi': 'Sardor', 'yoshi': 34}
```

2.3.13-rasm

Yuqoridagi holatda *l2* tarkibi o’zgarishsiz qoladi va *l1* lug’at tarkibiga boshqa lug’at elementlari qo’shiladi.

**Lug’at elementlariga murojaat.** Lug’at elementlariga murojaat uning kaliti orqali amalga oshiriladi. Ayniqsa *for* operatori orqali lug’at elementlarini uning kaliti orqali olish juda qulay hisoblanadi:

```
talabalar = {
    "+99890123": "Tolmas",
    "+99890124": "Bobur",     "+99890125": "Alisher" }
for tal in talabalar:
   print(tal, " - ", talabalar[tal])
```

2.3.14-rasm

Javobga quyidagi natija chiqariladi:

+99890123 - Tolmas

+99890124 - Bobur

+99890125 - Alisher

bu erda *for* operatoridagi *t* o’zgaruvchiga ketma – ket lug’at kaliti qiymatlari yuklanadi (chapdan o’ngga qarab) va shu kalit orqali lug’at elementiga murojaat amalga oshiriladi.

Lug’at elementlariga murojaat qilishning yana bir usuli *items*() metodini qo’llash orqali amalga oshiriladi. Yuqoridagi dastur kodi *items()* metodi orqali quyidagicha yoziladi va ayni bir xil natijaga erishiladi:

```
talabalar = {
    "+99890123": "Tolmas",
    "+99890124": "Bobur",     "+99890125": "Alisher" }
for nomer, ism in talabalar.items():
    print(nomer, " - ", ism)
```

2.3.15-rasm

*items*() metodi qiymat sifatida kortejlar to’plamini qaytaradi. Har bir kortej elementi kalit (*nomer*) va qiymatlar (*ism*) juftligidan tashkil topadi.

Lug’atdan faqat kalitlarini olish uchun *keys*() va faqat qiymatlarini olish uchun *values*() metodlaridan foydalaniladi, masalan:

```
talabalar = {
    "+99890123": "Tolmas",
    "+99890124": "Bobur",
    "+99890125": "Alisher" } # lug'atning kalitlariga murojaat
print("Kalitlar:")
for kalit in talabalar.keys():
    print(kalit, end='; ')
# lug'atning qiymatlariga murojaat
print("\n Qiymatlar:")
for qiymat in talabalar.values():
    print(qiymat, end='; ')
```

2.3.16-rasm

Ushbu dastur ishga tushirilganda quyidagi javob ekranga chiqariladi:

*Kalitlar:*

*+99890123; +99890124; +99890125;*

*Qiymatlar:*

*Tolmas; Bobur; Alisher;*

**Kompleks (murakkab) lug’atlar**. Lug’atlar faqatgina *int*, *str*, *float*, *bool* kabi oddiy turlarga oid berilganlardangina emas, balki *list*, *tuple*, *set*, *dict* kabi murakkab tuzulmaviy berilganlardan ham tashkil topishi mumkin:

```
loginData = {     "Zafar":
        {
            "email": "zafar@nuu.uz",
            "tel": "+99890933",
            "manzil": "Univer ko`chasi 4"
        },
    "Rustam":
        {
            "email": "rustam@nuu.uz",
            "tel": "+998902222",
            "manzil": "Dekanat ko`chasi 105"
        } }
print(loginData)
```

2.3.17-rasm

Yuqorida keltirilgan misolda *loginData* lug'ati (tashqi lug'at) o'z navbatida boshqa lug'atlar (ichki lug'atlar) dan tashkil topgan. Buday hollarda ichki lug'atni elementlariga quyidagi tarzda murojaat qilinadi:

```
lData = loginData["Zafar"]["tel"]
print(lData)
```

2.3.18-rasm

lug'atda mavjud bo'lmagan kalit orqali uning elementiga murojaat amalga oshirilganda Python interpretatori *KeyError* turidagi istisno xatoligini yuzaga keltiradi:

```
lData = loginData["Zafar"]["telegram"] #KeyError
```

2.3.19-rasm

bu erda “telegram” kalit so’zi mavjud emas. Shuning uchun istisno xatoligi ro’y berdi. Bunday xatoliklarning oldini olish uchun dastlab kalitning lug’atda bor yoki yo’qligini tekshirish tavsiya qilinadi:

```
key = "telegram"
if key in loginData["Zafar"]:
    print(loginData["Tom"]["telegram"])
else:
    print("telegram kaliti topilmadi")
```

2.3.20-rasm

Umuman olganda, murakkab lug’atlar (ichma – ich joylashgan) ustuda amallar oddiy lug’atlardagi kabi amalga oshiriladi.

**Nazorat savollari**

1.Pythonda Lug’atlardan qanday foydalaniladi? Misollar bilan tushuntirib bering.
2. Berilgan N natural son bir xonali bo’lsa uni matn ko’rinishida, aks holda “*Bunday son lug’atda mavjud emas”* yozuvini ekranga chiqarish dasturini lug’atdan foydalanib tuzing. *Masalan, N=1 bo’lsa “bir” chiqarilishi kerak.*
3.Berilgan ro’yxatning 2 marta qatnashgan elementlarini ekranga chop etuvchi dasturni lug’atlardan foydalanib tuzing.
*{Maslahat: Agar ro’yxat qiymati lug’atda mavjud bo’lmasa lug’atga uni 1 qiymatni bergan holda qo’shamiz,aks holda lug’atdagi qiymati 1 ga oshiriladi. Natijaga lug’atning qiymati 2 ga teng bo’lgan elementlarining kalitlari chiqariladi.}*

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд.,

перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**16-mavzu: Sana va vaqt bilan ishlash**

**Mavzu rejasi:**

1.Joriy sana va vaqtni olish.
2.Sana va vaqtni formatlash.
3.Datetime moduli: sana va vaqtni manipulyatsiya qilish.

Sana va vaqt bilan ishlaydigan quyidagi asosiy funksiyalar *datetime* modulida jamlangan:

- date; □ time;
- datetime.

### date sinfi

Sana bilan ishlash uchun *datetime* modulida aniqlangan *date* sinfidan foydalaniladi. *date* sinfi obyektini yaratish uchun uchta parameter: yil, oy va kun parametrlarini qabul qiladigan *date* sinfi konstruktoridan foydalaniladi:

```
date(year, month, day)
```

7.1.1-rasm

Masalan, qandaydir sana yaratamiz:

```
import datetime

yesterday = datetime.date(2017, 5, 2)
print(yesterday)  # 2017-05-02
```

7.1.2-rasm

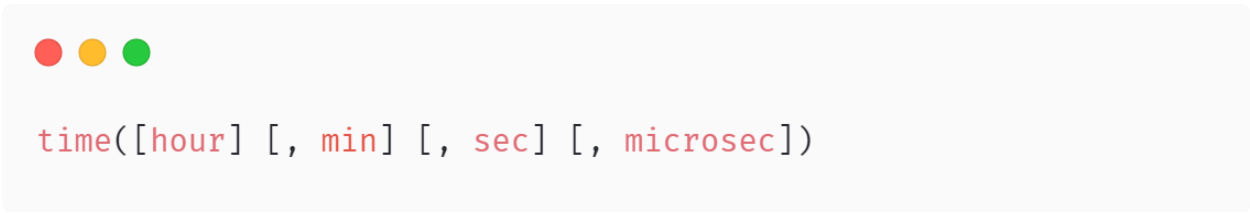
*today()* metodidan foydalanish orqali joriy sanani olish mumkin:

```
from datetime import date

today = date.today()
print(today)
print("{}.{}.{}".format(today.day, today.month, today.year))
```

7.1.3-rasm

*day*, *month* va *year* xususiyatlari orqali mos ravishda kun, oy va sanani olish mumkin. ***time* sinfi**

Vaqt bilan ishlash uchun time sinfidan quydagicha foydalaniladi:



```
time([hour] [, min] [, sec] [, microsec])
```

7.1.4-rasm

Konstruktor soat, minut, sekund va mikro sekundlarni ketma-ket ravishda qabul qiladi va bu parametrlar shart bo'lmagan parametrlardir. Agarda birorta parameter konstruktorda berilmasa, u holda kelishuv bo'yicha nol qiymat olinadi.

```
from datetime import time

current_time = time()
print(current_time)  # 00:00:00

current_time = time(16, 25)
print(current_time)  # 16:25:00

current_time = time(16, 25, 45)
print(current_time)  # 16:25:45
```

7.1.5-rasm

## datetime sinfi

*datetime* sinfi bir vaqtning o'zida sana va vaqt bilan ishlash imkoniyatini yaratadi. *datetime* sinfi obyektini yaratish uchun quyidagi konstruktor ishlatiladi:

```
datetime(year, month, day [, hour] [, min] [, sec] [, microsec])
```

7.1.6-rasm

Birinchi uchta parametr yil, oy va kunlar zaruriy parametrlar hisoblanadi, qolgan uchtasi esa shart bo'lmagan, hamda, kelishuv bo'yicha ular nol qiymat oladi.

```python
from datetime import datetime

deadline = datetime(2017, 5, 10)
print(deadline)  # 2017-05-10 00:00:00

deadline = datetime(2017, 5, 10, 4, 30)
print(deadline)  # 2017-05-10 04:30:00
```

7.1.7-rasm

Joriy vaqtni va sanani olish uchun *now()* metodidan foydalaniladi.

```python
from datetime import datetime

now = datetime.now()
print(now)
print("{}.{}.{}  {}:{}".format(now.day, now.month, now.year,
now.hour, now.minute))
print(now.date())
print(now.time())
```

7.1.8-rasm

*day*, *month*, *year*, *hour*, *minute*, *second* parametrlari orqali sana va vaqtning alohida qiymatlarini olish mumkin. *date()* va *time()* metodlari orqali esa mos ravishda alohida sana va vaqtni olish mumkin.

### Satrdan sanaga o’tkazish

*datetime* sinfida *strptime()* metodi mavjud bo’lib, u satr ko’rinishidagi berilgani vaqtga o’tkazadi. Bu metod ikkita parametr qabul qiladi:

```
strptime(str, format)
```

7.1.9-rasm

Birinchi *str* parametri sana va vaqtning satr ko'rinishi, hamda ikkinchi *format* parametri satrdagi sana va vaqt orasi qanday ajratilganligi formati.

Formatni aniqlash uchun quyidagi kodlarni ishlatamiz:

- %d – oy kuni son ko'rinishida;
- %m – oyning tartib raqami;
- %y – yil ikkita raqamdan iborat;
- %Y – yil to'rta raqamdan iborat;
- %H – soat 24 soatlik formatda;
- %M – minut;
- %S – sekund.

Har xil formatlarga misol:

```
from datetime import datetime
deadline = datetime.strptime("22/05/2017", "%d/%m/%Y")
print(deadline)  # 2017-05-22 00:00:00

deadline = datetime.strptime("22/05/2017 12:30", "%d/%m/%Y %H:%M")
print(deadline)  # 2017-05-22 12:30:00
deadline = datetime.strptime("05-22-2017 12:30", "%m-%d-%Y %H:%M")
print(deadline)  # 2017-05-22 12:30:00
```

7.1.10-rasm

**Nazorat savollari**

1.Pythonda *datetime* sinfining mohiyatini tushuntiring?

2.Pythonda *string* tipdan qanday qilib vaqt formatiga o’tqazilishini tushuntiring?

3.Pythonda vaqt formatlarining afzalliklarini tushuntiring?

4.Pythonda datetime sinfi orqali shu oyning oxirgi sanasi qaysi kun ekanligini aniqlovchi dastur tuzing?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

## 17-mavzu: Kalendar moduli va uning imkoniyatlari

**Mavzu rejasi:**

1.Kalendar moduli: kalendarni ko'rsatish va undan foydalanish xususiyatlari.
2.Kodni (parchalarini) bajarish vaqtini o'lchash.

### Sana va vaqtni formatlash

Ushbu sinflar doirasida sana va vaqt obyektlarini formatlash uchun *strftime(format)* metodi mavjud. Bu metod formatlashni ko’rsatuvchi bitta parametr qabul qiladi.

Formatlashni amalga oshirishimiz uchun quyida aniqlangan formatlash kodlaridan birini ishlatish mumkin:

- %a – hafta kuni uchun abbreviatoriya. Masalan, Wed – Wednesday so’zidan (kelishuv bo’yicha ingiliz tilidagi so’zlar ishlatiladi);

- %A – hafta kun to’liq, masalan, Wednesday;
- %b – oy kuni uchun abbreviatoriya. Masalan, Oct (October so’zining qisqartmasi);
- %B – oy nomi to’liq, masalan, October;
- %d – oy kuni, nol qo’shilgan, masalan, 01;
- %m – oy raqami, nol qo’shilgan, masalan, 05;
- %y – yil ikkita raqamdan iborat;
- %Y – yil to’rta raqamdan iborat;
- %H – soat 24 soatlik formatda, masalan, 13
- %I – soat 12 soatlik formatda, masalan, 01
- %M – Minut;
- %S – sekund.
- %f – mikrosekund;
- %p - AM/PM ko’rsatgich;
- %c – sana va vaqt, joriy mahalliy bo’yicha formatlangan;
- %x – sana, joriy mahalliy bo’yicha formatlangan;  %X - vaqt, joriy mahalliy bo’yicha formatlangan.

Har xil formatlar:

```
from datetime import datetime
now = datetime.now()
print(now.strftime("%Y-%m-%d"))
print(now.strftime("%d/%m/%Y"))
print(now.strftime("%d/%m/%y"))
print(now.strftime("%d %B %Y (%A)"))
print(now.strftime("%d/%m/%y %I:%M"))
```

7.2.1-rasm

## Sana va vaqtlarni qo’shish va ayirish

Sana va vaqt bilan ishlashda ma’lum bir vaqt oraliqdagi sanani qo’shish yoki ayirish zarurati tug’uladi. *datetime* modulida bu ishlarni amalga oshirish uchun maxsus *timedelta* sinfi aniqlangan. Ushbu sinf ma’lum bir vaqt oralig’dani aniqlaydi.

Vaqt oralig’ini aniqlash uchun *timedelta* sinfi konstruktori quyidagicha ishlatiladi:

```
timedelta([days] [, seconds] [, microseconds] [, milliseconds]
[,minutes] [, hours] [, weeks])
```

7.2.2-rasm

Konstrutorga mos ketma-ketlikda kunlar, sekundlar, mikrosekundlar, milisekundlar, minutlar, soatlar va haftalarni berish mumkin.

Bir qancha oraliqlarni aniqlaymiz:

```
from datetime import timedelta

three_hours = timedelta(hours=3)
print(three_hours)  # 3:00:00
```

7.2.3-rasm

*timedelta* obyektini ishlatish orqali qo’shish va ayirish amallarini bajarishimiz mumkin. Masalan, ikki kundan keyingi sanani olamiz:

```
from datetime import timedelta, datetime

now = datetime.now()
print(now)
two_days = timedelta(2)
in_two_days = now + two_days
print(in_two_days)
```

7.2.4-rasm

Yoki 10 soatu 15 minut oldin qanday vaqt bo’lganini aniqlaymiz, buning uchun joriy vaqtdan 10 soatu 15 minutni ayrish kerak:

```
from datetime import timedelta, datetime

now = datetime.now()
till_ten_hours_fifteen_minutes = now - timedelta(hours=10, minutes=15)
print(till_ten_hours_fifteen_minutes)
```

7.2.5-rasm

### timedelta xususiyatlari

*timedelta* sinfi bir qancha xususiyatlarga ega bo’lib, ular orqali vaqt oraliqlarini olish mumkin:

- days – kunlar miqdorini qaytaradi;
- seconds – sekundlar miqdorini qaytaradi;
- microseconds – mikrosekundlar miqdorini qaaytaradi.

Bundan tashqari umimiy sekundlar miqdorini qaytaruvchi *total_seconds()* metodi ham mavjud, hamda bunga kunlar, sekundlar va mikrosekundlar kiradi.

Masalan, ikki sana oralig'idagi vaqtni aniqlaymiz:

```
from datetime import timedelta, datetime

now = datetime.now()
twenty_two_may = datetime(2022, 12, 22)
period = twenty_two_may - now
print("{} kun  {} sekund   {} mikrosekund".format(period.days, period.seconds,
period.microseconds))
# 135 kun  16024 sekund   750740 mikrosekund

print("Hammasi: {} sekund".format(period.total_seconds()))
# Hammasi: 11680024.75074 sekund
```

7.2.6-rasm

## Sanalarni solishtirish

Satrlar va sonlar kabi sanani ham standart taqqoslash amallari yordamida solishtirish mumkin:

```
from datetime import datetime

now = datetime.now()
deadline = datetime(2019, 5, 22)
if now > deadline:
    print("Dasturni topshirish muddati o'tdi")
elif now.day == deadline.day and now.month == deadline.month and now.year == deadline.year:
    print("Bugun dasturni topshirish muddati")
else:
    period = deadline - now
    print("Qoldi {} kun".format(period.days))
```

7.2.7-rasm

## Nazorat savollari

1.Pythonda *timedelta* sinfining mohiyatini tushuntiring?

2.Pythonda yangi yilgacha necha soat, necha daqiqa va necha soniya qolganini aniqlovchi dastur tuzing(vaqtlarni ayirish usulidan foydalaning)?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

18-mavzu: **Funksiya va uning imkoniyatlari**

**Mavzu rejasi:**

1.Funktsiyani aniqlash va uning chaqirish.

2.Funktsiya aniqlanishining joylashuvi

3.Muhim bo’lmagan parametrlar va kalitlarni moslashtirish.

4.Funktsiyadagi parametrlarning o'zgaruvchan soni

Funksiyalar ma`lum bir vazifani bajaradigan va dasturning boshqa qismlarida qayta ishlatilishi mumkin bo’lgan kod blokini ifodalaydi. Funksiyaning rasmiy ta`rifi quyidagicha:

```
def funksiya_nomi([parametrlar ro`yxati]):
```

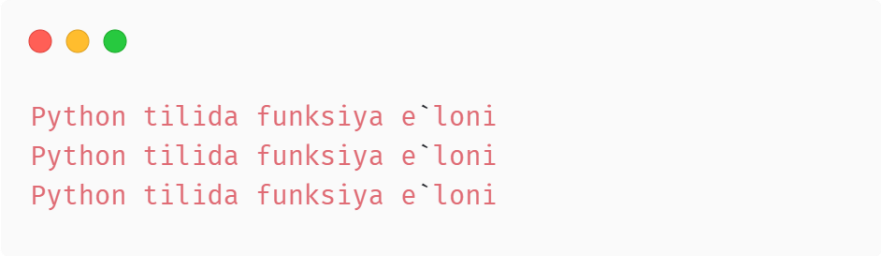

1.17.1-rasm

Ushbu funksiyaning nomi Display bo’lib, u parametrga ega emas. Bu funksiya chaqirilganda konsol ekraniga **"Python tilida funksiya e`loni!"** satri chiqariladi.

Funksiyani chaqirish uchun uning nomi va oddiy qavslar ichida mos parametrlariga qiymatlar berish orqali amalga oshiriladi, masalan:


1.17.2-rasm

Bu funksiya uch marta chaqirilmoqda va konsol ekraniga quyidagi ma`lumotlar chiqariladi:




1.17.3-rasm

**Rasm №3.**Natijaning konsolga chiqarilishi.

Quyida parametrli funksiyaning aniqlanishiga misol keltirilgan:

```
def Salom(ismi):
    print("Salom",ismi)

Salom("Tolib")
Salom("Rustam")
```

1.17.4-rasm

bu funksiya *ismi* nomli parametrga ega bo'lib, funksiya chaqirilganda parametrga turli qiymatlar berilgan va natijada konsolga quyidagi ma`lumotlar chiqarilgan:

```
# Salom Tolib

# Salom Rustam
```

1.17.5-rasm

**Nazorat savollari**

1.Pythonda fungsiya e'lon qilishni tushuntiring?

2. Pythonda  fungsiyadagi *return* operatorining vazifasini tushuntiring.

3. Pythonda qiymat qaytaruvchi va qiymat qaytarmaydigan fungsiyalar afzallik jihatlari haqida fikr yuriting?

4. Pythonda yoshingizni  kiritsangiz tug'ilganizga necha oy bo'lganini qaytaruvchi fungsiya tuzing.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

## 19-mavzu: Generator funksiyalari

**Mavzu rejasi:**

1.Generator funktsiyalari.

2.Funktsiya dekoratorlari.

3.Rekursiya. Faktorial hisoblash.

3.Global va local(mahalliy) o'zgaruvchilar.

4.O'rnatilgan funktsiyalar

O’zgaruvchilarning ko’rinish sohasi ularning dasturdagi qo’llanilishi mumkin bo’lgan qismi (kontekst)ga aytiladi. Python dasturlash tilida kontekstlarning ikki turi mavjud: global va lokal.

Global o’zgaruvchilar barcha funksiyalardan tashqarida aniqlangan bo’ladi va ixtiyoriy funksiyaning ichkarisida foydalanilish imkonini beradi. Masalan:

```
ismi = "Sardor"
def Salom():
    print("Salom", ismi)
def Xayr():
    print("Xayr", ismi)
Salom()
Xayr()
```

1.18.1-rasm

Bu erda har ikkala funksiyada *ismi* - lokal o’zgaruvchilari aniqlangan va ularning qo’rinish sohalari o’zi joylashgan funksiyaning ichida bo’lib, ularning har biri faqat o’zi joylashgan funksiya ichida amal qiladi.

Agar  lokal o’zgaruvchi va global o’zgaruvchi bir xil nomga ega bo’lsa, u holda lokal o’zgaruvchi o’zining ko’rinish sohasida global o’zgaruvchini “yashirib” qo’yadi. Masalan:

```
def Salom():
    ismi = "Tolib"
    familiyasi = "Otaboyev"
    print("Salom", ismi, familiyasi)
def Xayr():
    ismi = "Tolib"
    print("Xayr", ismi)
Salom()
Xayr()
def Salom():
    ismi = "Tolib"
    familiyasi = "Otaboyev"
    print("Salom", ismi, familiyasi)
def Xayr():
    ismi = "Tolib"
    print("Xayr", ismi)
Salom()
Xayr()
```

1.18.2-rasm

```
ismi = "Tolib"
def Salom():
    print("Salom", ismi)
def Xayr():
    ismi = "G'olib"
    print("Xayr", ismi)
    Salom()  # Salom Tolib
Xayr() # Xayr G'olib
```

1.18.3-rasm

1.18.3-rasmda 1-qatorda *ismi* deb nomlangan global o'zgaruvchi aniqlangan va xuddi shu nom bilan *Xayr()* funksiyasining ichida (6-qatorga qarang) ham lokal o'zgaruvchi aniqlangan. Funksiya ichida aniqlangan lokal o'zgaruvchi, funksiya ichida global o'zgaruvchi "yashirib" qo'yadi. Shuning uchun *Xayr*() funksiyasi chaqirilganda javobga lokal o'zgaruvchining qiymati chiqarilgan.

Agar funksiyalarning ichida global o'zgaruvchining qiymatini o'zgartirish talab qilinsa, u holda *global* kalit so'zidan foydalaniladi.

```
def Xayr():
    global ismi
ismi = "G'olib"
print("Xayr", ismi)

#Xayr G'olib
```

1.18.4-rasm

Funksiya ichkarisida global o’zgaruvchining qiymati o’zgartirilishidan oldin *global* kalit so’zi orqali ko’rsatib o’tilishi shart (2-qatorga qarang).

Odatda boshqa dasturlash tillaridagi kabi Python dasturlash tilida ham global o’zgaruvchilardan iloji boricha kamroq foydanish tavsiya qilinadi.

**Nazorat savollari**

1.Pythonda fungsiyaning lokal o’zgaruvchilarni tushuntiring?

2. Pythonda fungsiyaning global o’zgaruvchilarni tushuntiring?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**20-mavzu: Modullar**

**Mavzu rejasi:**

1.Import ko’rsatmasi.

2.From ko’rsatmasi.

3.Modul qidirish yo'llari

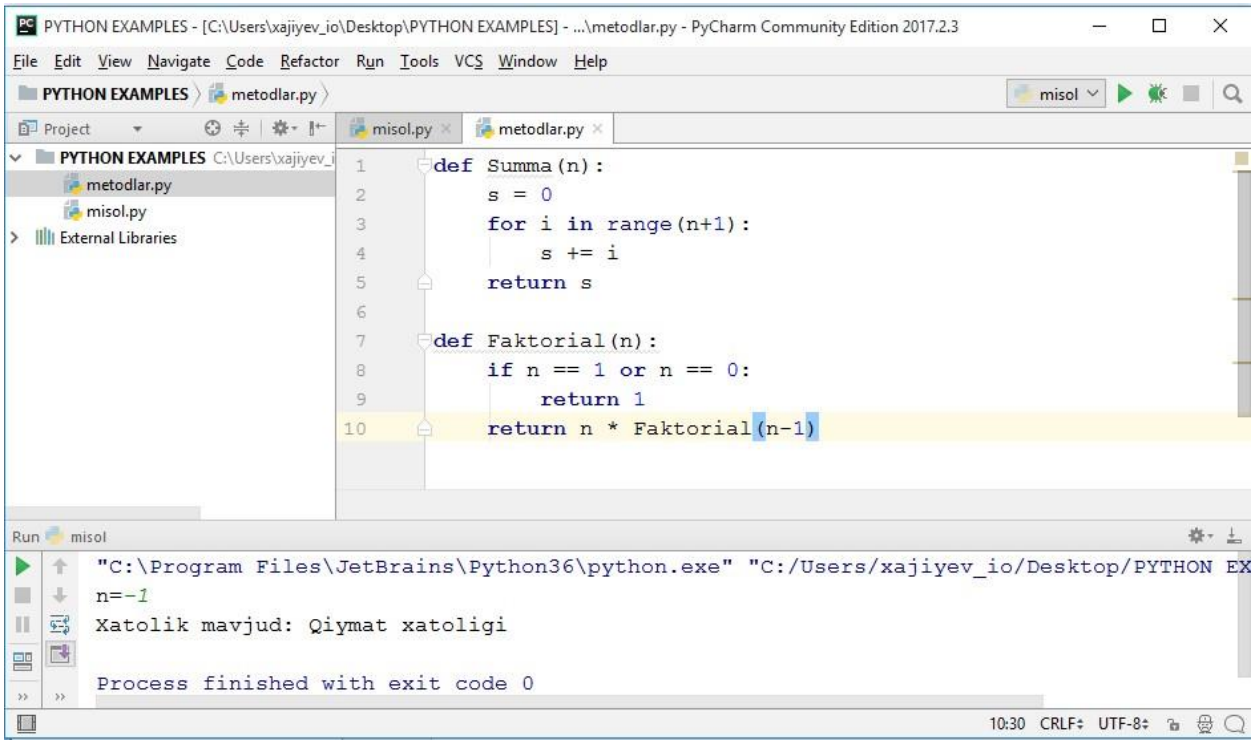
4.Modullarni qayta yuklash.

5.Paketlar

Python dasturlash tilida modullar alohida faylda yozilgan va boshqa dasturlarda qayta qo’llanilishi mumkun bo’lgan kodlar majmuini ifodalaydi.

Modullarni hosil qilish uchun ***.py*** kengaytmali fayl ochiladi va unga bir yoki bir nechta funksiyalar yoziladi. Faylning nomi keyinchalik modulning nomi sifatida qo'llaniladi.

Quyida Pycharm muhitida loyiha ikkita fayldan, ***misol.py*** nomli asosiy fayl va ***metodlar.py*** qo'shimcha tashqi modulni ifodalovchi fayldan tashkil topgan holatga misol keltirilgan(Python dasturlash tilida modullarni yaratish va ulash.) :




1.21.1-rasm

Yuqoridagi rasmda (1.21.1-rasmga qarang) ***metodlar.py*** nomli faylda *Summa(n)* va *Faktorial(n)* deb ataluvchi ikkita funksiya aniqlangan, ya`ni:

```
def Summa(n):
    s = 0
    for i in range(1, n + 1):
        s += i
    return s
def Faktorial(n):
    if n == 1 or n == 0:
        return 1
    return n * Faktorial(n - 1)
```

1.21.2-rasm

Ushbu funksiyalarni boshqa bir faylda  yoziladigan kodda modul sifatida ulab qo’llanilishi quyidagi dasturda ko’rsatilgan:

```
import metodlar
 try:
    n = int(input('n='))
    print('Summa:', metodlar.Summa(n))
print('Faktarial', metodlar.Faktorial(n)) except:
    print("Qiymat kiritish xatoligi ro`y berdi")
```

1.21.3-rasm

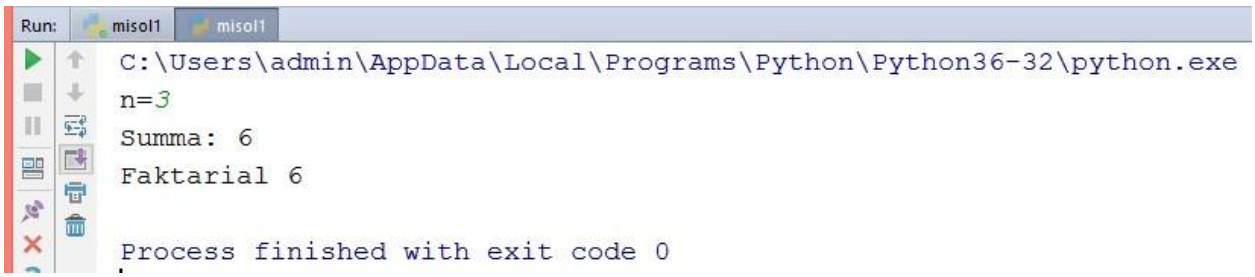
Moduldan foydalanish uchun dastlab uni dasturga ulash talab qilinadi. Buning uchun ***import*** kalit so’zi va undan keyin modul fayli nomi ko’rsatilishi kerak: *import metodlar*.

Modul funksionallariga murojaat qilish uchun uning **nomlar fazosi** olishishi kerak. Odatda u modul nomi bilan mos tushadi. Bizning holatda bu *metodlar* deb nomlangan.

Modulning nomlar fazosi olingandan keyin uning ichidagi funksiyalarga *nomlar_fazosi.funksiya* sxemasi bo’yicha murojaat qilinadi:

*metodlar.Faktorial(n).*

***misol1.py*** fayli kompilyatsiya qilinganda unga ulangan modullarda joylashgan funksiyalarga murojaat amalga oshiriladi. Xususan, dastur ishlaganda qonsolda quyidagicha ma`lumotlar chiqariladi:




1.21.4-rasm

**Nomlar fazosini sozlash.** Odatda modullar import qilinganda, nomlar fazosi va modul fayli nomlari bir xil bo'ladi va modul funksionallariga shu nomlar fazosi orqali murojaat qilinadi. Lekin ba`zi holatlarda nomlar fazosi nomining haddan ziyod uzunligi noqulayliklar keltirib chiqarishi mumkin. Bunday holatlarda nomlar fazosiga ***as*** kalit so'zidan foydalanib, psevdonom berish va u orqali modul funksionallariga murojaatlarni amalga oshirish mumkin. Masalan:

```
import metodlar as mt
 try:
    n = int(input('n='))
    print('Summa:', mt.Summa(n))
    print('Faktarial', mt.Faktorial(n))
 except:
    print("Qiymat kiritish xatoligi ro`y berdi")
```

1.21.5-rasm

Bu holatda nomlar fazosi *mt* deb nomlangan.

Modullarni ulashning boshqa usullaridan biri ***from*** kalit so'zidan foydalangan holda amalga oshiriladi. Bunda joriy modulning global nomlar fazosiga ulanilayotgan modulning funksionallari chaqiriladi:

```
from metodlar import Summa
print('Summa:',Summa(n))
```

1.21.6-rasm

Bu holatda ***metodlar*** modulidan ***Summa*** funksiyasi global nomlar fazosiga import qilingan. Shuning uchun import qilingan funksiyadan foydalanilganda (funksiya joriy faylda chaqirilganida), xuddi bu funksiya shu faylda aniqlangandek, uning oldida nomlar fazosi ko’rsatilmasligi ham mumkin.

Moduldagi barcha funksiyalarni global nomlar fazosiga birdaniga import qilish uchun maxsus belgi * dan foydalaniladi.

```
from metodlar import *
print('Summa:',Summa(n))
print('Faktarial',Faktorial(n))
```

1.21.7-rasm

Shuni alohida ta`kidlash kerakki, moduldagi barcha funksiyalarni bunday tarzda global nomlar fazosiga import qilish kolliziyaga olib kelishi mumkin. Masalan, agar joriy faylda ham import qilinayotgan moduldagi funksiya nomi bilan ayni mos tushadigan funksiya yoki o’zgaruvchilar bo’lsa kolliziya holati bo’ladi va dastur ishga tushirilganda xatolikka olib kelishi mumkin. Shuning uchun odatda bu tarzda modulni ulashga maslahat berilmaydi.

**Modul nomi**. 5-rasmdagi keltirilgan dasturda *misol.py* fayli asosiy modulni ifodalab, unga *metodlar.py* moduli ulangan edi. *misol.py* ni ishga tushirganda barcha zarur ishlar bajariladi. Shuningdek, *metodlar.py* ning yakka o’zi ishga tushirilsa hech qanaqa ish bajarilmaydi va konsolga hech nima chiqmaydi. Chunki

ushbu modulda faqatgina funksiyalar aniqlangan va boshqa ish bajarilmagan. Lekin *metodlar.py* ga shunday o'zgartirishlar kiritish mumkinki, uning alohida o'zi ham bajarilishi yoki u boshqa modulga ulanishi ham mumkin bo'ladi.

Modul bajarilganda muhit uning nomini aniqlaydi va uning nomini *__name__* (oldin va keyin ikkita tag chizig'i mavjud) global o'zgaruvchisiga yuklaydi. Agar modul bajariluvchi bo'lsa, uning nomi *__main__* (har ikkala tomonida ikkita tag chizig'i bor) ga teng bo'ladi. Agar modul boshqa modulda ishlatilayotgan bo'lsa, u holda bajarilish paytida uning nomi mos ravishda fayl nomi bilan bir xil bo'ladi (faqat ***.py*** kengaytmasisiz). *metodlar.py* fayliga quyidagicha o'zgartirishlar kiritamiz:

```python
def Summa(n):
    s = 0
    for i in range(1, n + 1):
        s += i
    return s
def Faktorial(n):
    if n == 1 or n == 0:
        return 1
    return n * Faktorial(n - 1)
def Main():
 n = 4
 s = Summa(n)
 f = Faktorial(n)
 print('Summa:', s)
 print('Faktorial:', f)
if __name__ == "__main__":
   Main()
```

1.21.8-rasm

Endi bu modul boshqa modullarga bog'liq bo'lmagan holda o'zi ham bajarilishi mumkin. Odatda modullarning bunday aniqlanishi undagi funktsionallarni

tekshirib ko’rish uchun qilinadi. Yuqoridagi holatda qo’shimcha *Main* funksiyasi aniqlanib, unda test qilinishi kerak bo’lgan funksiyalar chaqirilgan. Agar shu modulning o’zi bajarilganda (kompilyatsiya qilinganda), u holda *if* sharti bajariladi va natijada *Main* funksiyasi chaqiriladi. Albatta qo’shimcha *Main* aniqlanishi shart emas. Test qilish uchun barcha funksiyalarni *if* operatorining ichida chaqirish ham kifoya bo’lar edi.

Shunday qilib, *metodlar.py* skriptining o’zi alohida ishga tushirilsa, o’z – o’zidan *Python* *__name__* global o’zgaruvchisiga *__main__* nomini beradi. Agar u boshqa skriptga qo’shimcha modul sifatida ulangan bo’lsa va boshqa skriptda chaqirilsa, u holda *__name__* global o’zgaruvchisiga *metodlar* nomi beriladi.

**Import ko’rsatmasi**

Import ko’rsatmasi modulni import qilish imkonini beradi. O'rnatilgan modullarni ulash uchun biz bir necha bor ushbu qo'llanmaga murojaat qildik. Masalan, biz strftime() funksiyasidan foydalanib joriy sanani olish uchun vaqt modulini kiritdik:

vaqtni import qilish # Vaqt modulini import qilish

chop etish (vaqt .strftime (" % d .% m .% Y " )) # Joriy sanani chop etish

Import bayonoti quyidagi formatga ega:

import <Modul nomi 1> [<taxallus 1> sifatida] [, . . . ,

<Modul nomi N> [taxallus N > sifatida] ]

Import kalit so'zidan keyin modul nomi keladi. Iltimos, ism kengaytmalar va fayl yo'llarini o'z ichiga olmaydi. Modullarni nomlashda shuni yodda tutingki, import operatsiyasi bir xil nomdagi identifikatorni yaratadi, ya'ni modul nomi o'zgaruvchilarni nomlash qoidalariga to'liq mos kelishi kerak. Raqam bilan boshlanadigan nom bilan modul yaratish mumkin, lekin bunday modulni import qilib bo'lmaydi. Bundan tashqari, siz modul nomlarini kalit so'zlar, o'rnatilgan

identifikatorlar va standart kutubxonaga kiritilgan modul nomlari bilan moslashtirishdan qochishingiz kerak.

Bir vaqtning o'zida vergul bilan ajratilgan bir nechta modullarni import qilishingiz mumkin. **Masalan,**

```
import time, math
print(time.strftime("%d.%m.%Y"))
print(math.pi)
```

Modulni import qilgandan so'ng, uning nomi modul ichida belgilangan atributlarga kirishingiz mumkin bo'lgan identifikatorga aylanadi. Modul atributlariga nuqta belgisi yordamida kirish mumkin. Masalan, **math** moduli ichida joylashgan Pi doimiysiga quyidagicha murojaat qilishingiz mumkin:

**math.pi**

funksiyasi modul atributining satr sifatida berilgan nomi bilan qiymatini olish imkonini beradi. Ushbu funktsiya yordamida siz atribut nomini dinamik ravishda yaratishingiz mumkin - dasturni bajarish paytida. Funktsiya formati:

} : l k bwI"– ˚ ¸#ˆ<r w ˇ"# r• I ˙. ˇ , #˚ ˆ.˙ ˇT(†c

Belgilangan atribut topilmasa, istisno tashlanadi g k 9m ` : ^ k k k Xato xabarini oldini olish uchun uchinchi parametr atribut mavjud bo'lmasa qaytariladigan qiymat bo'lishi mumkin. Funktsiyadan foydalanishga misol 12.3 ro'yxatda ko'rsatilgan.

## Nazorat savollari

1.Pythonda nomlar fazosini tavsifini tushuntiring.

2. Pythonda boshqa fayldan fungsiyani import qilib ishlatish usulini tushuntiring?

3.Import ko’rsatmasi nima?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд.,

перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**21-mavzu: Ob'ektga yo'naltirilgan dasturlash**

**Mavzu rejasi:**

1.Sinfni aniqlash va sinfni yaratish.

2.__init__() va __del__() usullari.

3.Meros olish. Ko'plikda meros olish.

4.Maxsus usullar.

5.Operatorning haddan tashqari yuklanishi.

Python dasturlash tilida obyektga yo’naltirilgan dasturlash tamoyilini ham ishlatish mumkin bo’lib, bu o’z navbatida dastur kompanentlarini sinflar ko’rinishida ifodalash imkonini beradi.

Sinf obyektning shabloni yoki formal tavsifi hisoblanadi. Obyekt esa ushbu sinfning haqiqiy shaklangan nusxasi (экземпляри) hisoblanadi. Quyidagicha tatbiq qilish mumkin: hammada inson to’g’risida qandaydir tassavvur mavjud – ikkita qo’li, ikkita oyog’i, boshi, oshqozoni, asab tizimi va boshqalari mavjud. Demak, shablon bor – bu shabloni sinf deb atash mumkin. Haqiqatdan ham, mavjud bo’lgan odamni bu sinfning obyekti deyish mumkin.

Kod nuqtai nazaridan sinf – funksiyalar va o’zgaruvchilar to’plamini ma’lum bir vazifalarni bajarish uchun o’zida birlashtiradi. Sinfning funksiyalari odatda metodlar deb ataladi. Ular sinfning xususiyatlarini ifodalaydi. Sinfning o’zgaruvchilari esa atributlar deb nomlanadi – ular singning holatini saqlaydi.

Sinf *class* kalit so’zi bilan aniqlanadi:

```
class sinf_nomi:
sinf_metodilari
```

6.1.1-rasm

Sinf obyektini yaratish uchun quydagi sintaksis ishlatiladi:

```
obyekt_nomi = sinf_nomi([parametrlar])
```

6.1.2-rasm

Masalan, insonni ta’riflaydigan *Person* sinfini aniqlaymiz:

```
class Person:
name = "Ali"
  def display_info(self):
        print("Salom, mening ismim", self.name)

person1 = Person()
person1.display_info()  # Salom, mening ismim Ali
person2 = Person() person2.name = "Salim"
person2.display_info()  # Salom, mening ismim Salim
```

6.1.3-rasm

*Person* sinfida insonning ismini saqlovchi *name* atributi va inson haqida ma’lumot chiqaradigan *display_info* metodlari aniqlangan.

Ixtiyoriy sinfning metodi aniqlayotganda hamma metodlarning birinchi parametri sifatida obyektning o’ziga ko’rsatgich bo’lgan *self* (bir qator dasturlash tillarida parametrning anologi sifatida *this* kalit so’zi ishlatiladi) parametri

bo’lishini hisobga olish kerak. Ushbu ko’rsatgich yordamida sinfning atriburlariga va metodlariga murojaat qilishimiz mumkin. Xususiy holda, *self.name* ifodasi orqali foydalanuvchi ismini olish mumkin.

*Person* sinfini aniqlagandan so’ng, *person1* va *person2* uning ikkita obyetkini yaratamiz. Obyekt nomlaridan foydalanish orqali uning metodlariga va atributlariga murojaat qilishimiz mumkin. Ushbu holda, qora oynaga sartni chiqaruvchi *display_info()* metodini har bir obyektdan chaqiramiz va ikkinchi obyektning *name* atributini o’zgartiramiz. *display_info* metodiga murojaatda *self* parametiriga hech qanday qiymat berish shart emas.

Vorislik avvaldan mavjud sinf asosida yangi sinf imkonini beradi. Vorislik inkapsulyatsiya bilan bir qatorda obyektga yo’naltirilgan dasturlashning asosini tashkil qiladi. Vorislikning asosiy tushunchalari bu: super_sinf va sinf_osti lar hisoblanib, sinf_osti sinfi  super_sinfdan hamma ochiq atributlar va metodlarni voris qilib oladi. Super_sinf ko’pincha asos yoki ota sinf deb ham nomlanadi, sinf ostisi esa hosilaviy sinf, voris sinf yoki bola sinf ham deb nomlanadi.

Sinflar vorisliklari uchun sintaksis quyidagi ko’rinishga ega bo’ladi:

```
class sinfostisi (supersinf):
    sinfostisi_azolari
```

6.6.1-rasm

O’tgan darslarda insonni tasvirlaydigan *Person* sinfini yaratgandik.

Korxonada ishlovchi ishchini tasvirlaydigan sinfni yaratish zarur deb qaraylik.

Buning uchun yangi *Employee* sinfni noldan yaratishimiz mumkin. Biroq,

*Employee* sinfining atributlari va metodlari ham *Person* sinfidagilar kabi bo'ladi, chunki ishchi ham inson. Shuning uchun ham *Employee* sinfini qaytadan yaratishda ma'no qolmaydi. Ushbu holda vorislikni ishlatish qulay hisoblanadi. *Person* sinfidan *Employye* sinfiga vorislik olamiz:

```
class Person:
    def __init__(self, name, age):
        self.__name = name  # ismni o'rnatamiz
        self.__age = age  # yoshni o'rnatamiz

    @property
    def age(self):
        return self.__age

    @age.setter
    def age(self, age):
        if age in range(1, 100):
            self.__age = age
        else:
            print("Mumkin bo'lmagan yosh")
    @property
    def name(self):
        return self.__name
    def display_info(self):
        print("Ism:", self.__name, "\tYosh:", self.__age)
class Employee(Person):
    def details(self, company):
        print(self.name, company, " kompaniyasida ishlaydi")

ali = Employee("Ali", 23)
ali.details("Google")
ali.age = 33
ali.display_info()
```

6.6.2-rasm

*Employee* sinfi to’lig’icha *Person* sinfning funksiollarini qabul qiladi va qo’shimcha ravishda *details()* metodi qo’shildi.

Shuni ta'kidlash kerakki, *Employee* sinfi uchun *Person* sinfidagi *__name* yoki *__age* turidagi yopiq atributlardan tashqari barcha metodlar va atributlar *self* kalit so'z orqali murojaat mavjud. *Employee* obyektini yaratishda biz aslida *Person* sinfining konstruktoridan foydalanamiz. Bundan tashqari, ushbu obyektda *Person* sinfining barcha metodlarini chaqirishimiz mumkin.

**Nazorat savollari**

1.Obyektga asoslangan dasturlash nima?
2.Obyektga asoslangan dasturlashning asosiy obyekti nima?
3.Klasslar qanday e’lon qilinishi va foydalanishini tushuntiring?
4.Python dasturlash tilida familiya,ism,fakultet va kursi kabi ma’lumotlarni o’z ichiga oluvchi klass e’lon qiling.
5.Pythonda vorislik mexanizmini izohlab bering?
6.Pythonda vorislik mexanizmidan foydalanishga misollar keltiring?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**22-mavzu: Sinf va uning xususiyatlari**

**Mavzu rejasi:**

1.Sinf ichidagi identifikatorlarga kirishni cheklash.

2.Sinf xususiyatlari.

3.Sinf dekoratorlari.

Qoidaga ko’ra, sinflar alohida modullarda joylashadi va asosiy skriptda ular import qilinadi. Aytaylik bitta loyihada ikkita fayli mavjud: *main.py* (dasturning asosiy skripti) va *classes.py* (sinflar aniqlangan skript).

*classes.py* faylida ikkita sinf aniqlaymiz:

```
class Person:     # konstruktor
  def __init__(self, name):
        self.name = name  # nomni o'rnatamiz
  def display_info(self):
        print("Salom, ismim", self.name)
class Auto:
  def __init__(self, name):
        self.name = name
  def move(self, speed):
        print(self.name, "tezlik bilan harakatlanayapti ", speed, "km/s")
```

6.4.1-rasm

*Person* sinfiga qo’shimcha ravishda avtomobilni harakterlovchi *move* va *name* atributi bor *Auto* sinfi aniqlangan. Bu sinflarga bog’lanamiz va *main.py* asosiy dastur skriptida foydalanamiz:

```
from classes import Person, Auto

ali = Person("Ali") ali.display_info()

bmw = Auto("Malibu") bmw.move(65)
```

6.4.2-rasm

Sinflarga bog’lanish ham moduldan funksiyani importi kabi amalga oshiriladi.

Yoki to’liq modulga ham bog’lanish mumkin:

```
Import classes
```

6.4.3-rasm

Natijada quyidagini olamiz:

```
#Salom, ismim Ali
#Malibu tezlik bilan harakatlanayapti  65 km/s
```

6.4.4-rasm

6.5. Inkapsulyatsiya

Kelishuv bo’yicha sinflardagi atributlar umumiy ruxsatga ega bo’ladi, ya’ni, dasturning ixtiyoriy joyidan obyektning atributlariga ruxsat olish va ularni o’zgartirish mumkin. Masalan:

```
class Person:
  def __init__(self, name):
        self.name = name  # ismini o'rnatish
        self.age = 1  # yoshni o'rnatish
  def display_info(self):
        print("Ism:", self.name, "\tYosh:", self.age)

ali = Person("Ali")
ali.name = "O'rgamchak odam"  # name atributini o'zgartirish
ali.age = -129  # age atributini o'zgartirish
ali.display_info()  # Ism: O'rgamchik odam     Yosh: -129
```

6.5.1-rasm

Lekin ushbu holda, misol uchun, yosh yoki odamning ismini noto’g’ri o’zgartirish mumkin, masalan, yuqoridagi kabi yoshiga manifiy son yozish. Ushbu holda obyekt atributiga murojaatni nazorat qilish haqida savol paydo bo’ladi.

Bunday muammo ikapsulyatsiya tushunchasi bilan chambarchas bog’liq. *Iknapsulyatsiya* obyektga yo’naltirilgan dasturlashning fundamental tushunchalaridan biri hisoblanadi. Kodnig murojaat joyidan obyekt atirbutiga to’g’ridan-to’g’ri murojaat qilishni chegaralaydi.

Python dasturlash tilida sinf atributlarini inkapsulyatsiya yordamida ko’rinmas yoki yopiq va chegaralangan murojaatni maxsus metodlar orqali o’rnatish mumkin. Odatda ular, xususiyat deb ham ataladi.

Yuqoridagi aniqlangan sinfga quyidagicha xususiyatlar qo’shamiz:

```
class Person:
  def __init__(self, name):
      self.__name = name  # ismini o'rnatish
      self.__age = 1  # yoshni o'rnatish
  def set_age(self, age):
    if age in range(1, 100):
          self.__age = age
    else:
          print("Mumkin bo'lmagan yosh")
  def get_age(self):
    return self.__age
  def get_name(self):
    return self.__name
  def display_info(self):
      print("Ism:", self.__name, "\tYosh:", self.__age)

ali = Person("Ali")

ali.__age = 43  # age atributi o'zgamaydi
ali.display_info()  # Ism: Ali  Yosh: 1
ali.set_age(-3486)  # Mumkin bo'lmagan yosh
ali.set_age(25)
ali.display_info()  # Ism: Ali  Yosh: 25
```

6.5.2-rasm

Yopiq atribut yaratish uchun atribut nomi oldidan ikkita tag chiziq qo’yish lozim: *self.__name*. Bunday atributlarga faqat ushbu sinfning ichidagina murojaat qilish mumkin, lekin, sinfdan tashqarida murojaat qilib bo’lmaydi. Masalan, atributga qiymat yozib bo’lmaydi:

```
ali.__age = 43
```

6.5.3-rasm

Uning qiymatini olishga urinsak xatolik yuz beradi:

```
print(ali.__age)
```

6.5.4-rasm

Lekin *__age* atributinga sinf tashqarisida qiymat berish kerak bo'lishi mumkin Buning uchun xususiyat yaratiladi va u orqali atribur qiymatiniga murojaat amalga oshiriladi. Masalan quyida *get_age()* metodi orqali *__age* atributi qiymatini sinf tashqarisida olish mumkin:

```
def get_age(self):
  return self.__age
```

6.5.5-rasm

Ushbu metod getter yoki aksessor deb ham nomlanadi.

Qiymatni o'zgartirish uchun esa yana boshqa xususiyati aniqlanadi:

```
def set_age(self, value):
  if value in range(1, 100):
        self.__age = value
  else:
        print("Mumkin bo'lmagan yosh")
```

6.5.6-rasm

Bu yerda esa biz kelayotgan qiymatni shartga tekshirish mumkin bo’ladi. Bu metod setter yoki myuteytor deb ham nomlanadi.

Har bir yopiq atribut uchun bir juft xususiyat aniqlash shart emas. Yuqoridagi misolda faqat konstruktor yordamida qiymatlarni berish mumkin. Qiymatni olish uchun *get_name* metodidan foydalniladi.

### Xususiyatlar annotatsiyasi

Yuqorida qanday qilib xususiyatlarni yaratishni ko’rib chiqdik. Python dasturlash tilida yanada qulayroq usul mavjud. “*@”*- belgisi bilan boshlanuvchi annotatsiyadan foydalanish orqali bu usul amalga oshiriladi.

Getter-xususiyat yaratish uchun *@property* annototsiyasini qo’yish lozim. Setter-xususiyat qo’yish uchun esa *@setter_xususiyatnomi.setter* annotosiyasni o’rnatish lozim.

*Person* sinfini annototsiyadan foydalanib qayta yozamiz:

```python
class Person:
  def __init__(self, name):
        self.__name = name  # ismni o'rnatamiz
        self.__age = 1  # yoshni o'rnatamiz
    @property
  def age(self):
        return self.__age

    @age.setter
  def age(self, age):
    if age in range(1, 100):
            self.__age = age
    else:
            print("Mumkin bo'lmagan yosh")

    @property
  def name(self):
        return self.__name
  def display_info(self):
        print("Ism:", self.__name, "\tYosh:", self.__age)

ali = Person("Ali")
ali.display_info()  # Ism: Ali  Yosh: 1
ali.age = -3486  # Mumkin bo'lmagan yosh
print(ali.age)  # 1
ali.age = 36
ali.display_info()  # Ism: Ali  Yosh: 36
```

6.5.7-rasm

Birinchi navbatda, setter-xususiyati getter-xususiyatidan so’ng aniqlanishiga e’tibor qaratish zarur. Ikkinchidan, setter va getter *age* bir xil nomlangan. Shuning uchun ham, getter *age* nomlangan, shu sababdan ham setterga *@age.setter* annotatsiyasi o’rnatilgan.

Natijada, getter va setter *ali.age* ifodasi bilan murojaat qilish mumkin bo’ladi.

## Nazorat savollari

1.Pythonda inkapsulyatsiya nima va u qanday vazifani bajaradi?

2.Pythonda xususiyatlar annotatsiyasidan foydalanishning mohiyatini tushuntiring?
3.Getter va Setter xusuiyatlarini tushuntirib bering.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**23-mavzu:Istisnolarni qayta ishlash**

**Mavzu rejasi:**

1.Try . . . except . . . else . . . finally ko’rsatmasi.
2.With . . . as ko’rsatmasi.
3.O'rnatilgan istisno sinflari.
4.Foydalanuvchi istisnolar.

Python dasturlash tilida kod yozilganda ikki turdagi xatoliklarni uchratish mumkin: Birinchi turi bu **sistaksis xatoligi (syntax error)** bo’lib, bunday turdagi xatoliklar odatda til sintaksisidan noto’g’ri foydalanganda yuzaga keladi. Agarda dasturda sintaksis xatoligi bo’lsa, uni kompilyatsiya qilishning imkoni bo’lmaydi. Kod yozishda Pycharm yoki shunga o’xshash boshqa IDElardan foydalanganda, odatda muhitning o’zi bunday turdagi xatoliklarni aniqlab beradi va ajratib ko’rsatadi; Ikkinchi turdagi xatoliklar bu **bajarilish davridagi xatolik (Runtime error)** bo’lib, odatda bunday turdagi xatoliklar kompilyatsiya bosqichida emas balki dastur bajarilishi jarayonida yuzaga keladigan xatoliklar hisoblanadi. Bunday xatoliklarga istisno holatlar ham deyiladi. Masalan:

```
str = "hello"
number = int(str)
print(number)
```

1.22.1-rasm

Bu erda sistaksis xatolik kuzatilmaydi, lekin kompilyatsiya jarayonida ***ValueError*** deb nomlanuvchi xatolik konsolga chiqariladi. Chunki *str* o'zgaruvchisida satr qiymat mavjud va uni 2-qatorda butun turga o'girilmoqda, bu esa mumkin emas. Bunday turdagi istisno holatlar ayniqsa foydalanuvchi tomonidan qiymatlar konsol ekranidan kiritilganda ro'y berish ehtimolligi kattaroq. Masalan:

```
str = input("Satr kiriting:")
number = int(str)
print(number)
```

1.22.2-rasm

Agar foydalanuvchi tomonidan konsoldan sondan farqli bo'lgan qiymatlar kiritilsa xatolik ro'y beradi.

Istisno holatlar ro'y berganda, dastur ishlashdan to'xtatiladi. Bunday holatlarni boshqarish uchun Python dasturlash tilida maxsus ***try except*** tuzilmasi mavjud bo'lib, u quyidagicha aniqlanadi: ***try***:

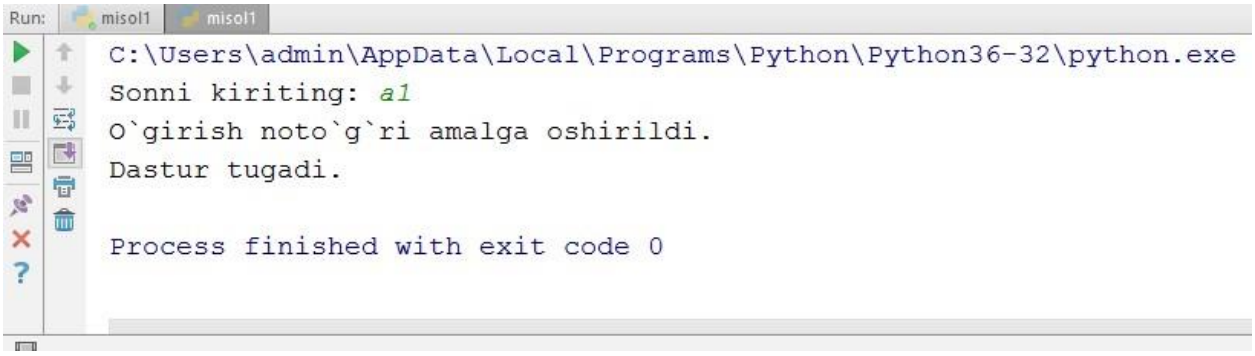
*instruktsiyalar*

***except*** [*Istisno_turi*]:

Ushbu dasturda *son* o'zgaruvchisiga qiymat sifatida konsoldan satr kiritilsa konsol ekraniga quyidagi ma`lumotlar chiqariladi:

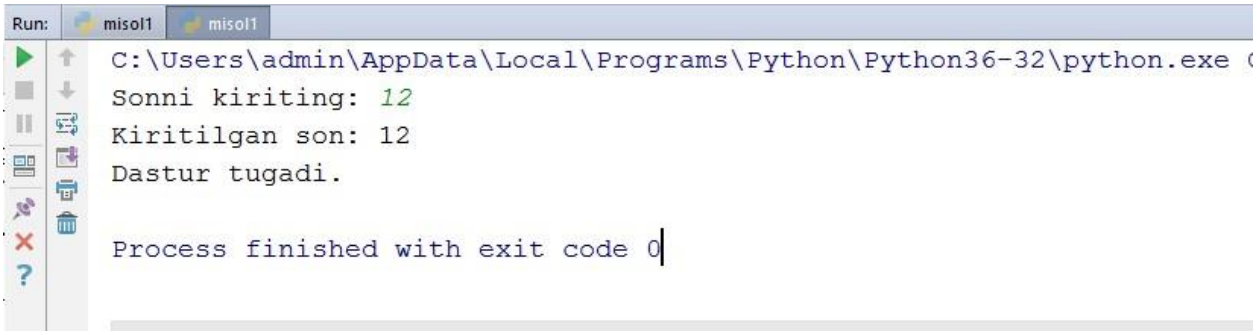
```
try:
    son = int(input("Sonni kiriting: "))
    print("Kiritilgan son:", son)
except:
    print("O`girish noto`g`ri amalga oshirildi.")
    print("Dastur tugadi.")
```

1.22.3-rasm




1.22.4-rasm

1.22.4-rasmda konsol ekranida istisno ro'y bergan holatdagi ma`lumotlar. Agar *son* o'zgaruvchisiga qiymat sifatida konsoldan son kiritilsa konsol ekraniga quyidagi ma`lumotlar chiqariladi:




1.22.5-rasm

1.22.5-rasmda konsol ekranida qiymat to'g'ri kiritilgandagi holat.

Yuqorida, dastur ishga tushirilganda qiymat xato kiritilgandagi (1.22.4-rasm

) va to’g’ri kiritilgan paytdagi (1.22.5-rasmda) holatlarlarda konsol ekraniga chiqarilgan ma`lumotlarni ko’rish mumkin. Shuni alohida ta`kidlash kerakki, agar *try – except* tuzilmasi qamrovida istisno holati yuzaga kelgan taqdirda dastur to’xtab (qotib) qolmaydi, balki boshqaruv istisno holatini qayta ishlovchi qismga (*except blokiga*) uzatiladi.

Yuqoridagi misolda kodda yuzaga kelishi mumkin bo’lgan barcha istisno holatlarni qayta ishlovchi holat ko’rildi. Ammo qayta ishlanuvchi istisno turini konkretlashtirish ham mumkin. Buning uchun *except* kalit so’zidan keyin istisno turi yoziladi:

```
try:
    son = int(input("Sonni kiriting: "))
    print("Kiritilgan son:", son)
except ValueError:
    print("O`girish noto`g`ri amalga oshirildi.")
print("Dastur tugadi.")
```

1.22.6-rasm

Agar kodda bir necha xil istisno holatlari ro’y berishi mumkin bo’lsa, qo’shimcha *except* ifodalardan foydalangan xolda, ularning har birini alohida qayta ishlashimiz mumkin bo’ladi, masalan:

```
try:
    son1 = int(input("Birinchi sonni kiriting: "))
    son2 = int(input("Ikkinchi sonni kiriting: "))
    print("Bo`lish natijasi:", son1/son2)
except ValueError:
    print("O`girish muaffaqiyatsiz amalga oshirildi.")
except ZeroDivisionError:
    print("Nolga bo`lish yuzaga keldi.")
except Exception:
    print("Umumiy istisno holati.")
print("Dastur tugadi.")
```

1.22.7-rasm

Agar kiritilgan satrni songa o'girishda istisno ro'y bersa *ValueError* bloki, kiritilgan ikkinchi qiymat 0 soniga teng bo'lsa *ZeroDivisionError* bloki orqali qayta ishlanadi. Shu ikki istisno turidan boshqa ixtiyoriy istisno ro'y bersa *Exception* blokida qayta ishlanadi.

***Finally* bloki**. Istisno holatlar bilan ishlaganda *finally* blogini ham ishlatish mumkin. Bu blok ixtiyoriy hisoblanib, uning o'ziga xosligi shundan iboratki, ushbu blokda yozilgan kodlar istisno holati ro'y berish bermasligidan qat`iy nazar har doim bajariladi.

```
try:
    son = int(input("Birinchi sonni kiriting: "))
except ValueError:
    print("O`girish muaffaqiyatsiz amalga oshirildi.")
finally:
    print("Try bloki ishi tugadi")
print("Dastur tugadi.")
```

1.22.8-rasm

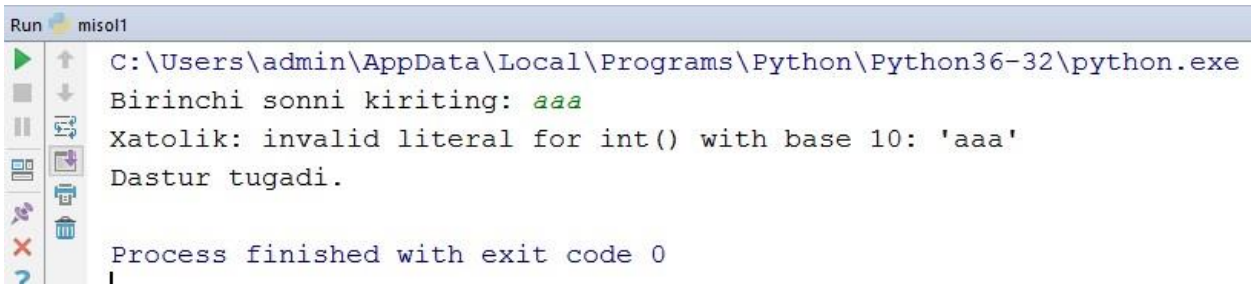
Odatda *finally* blokida dasturda foydalanilgan resurslarni bo'shatish, masalan, faylni yopish kabi qodlar yoziladi.

**Istisno to'g'risida ma`lumot olish**. ***as*** operatori yordamida *except* blokida foydalanilishi mumkin bo'lgan istisno to'g'risidagi barcha ma`lumotlarni o'zgaruvchiga uzatish mumkin:

```
try:
    son = int(input("Sonni kiriting: "))
except ValueError as e:
    print("Xatolik:",e)
print("Dastur tugadi.")
```

1.22.9-rasm

Konsoldan noto'g'ri qiymat kiritilgandagi holati quyidagicha:




1.22.10-rasm

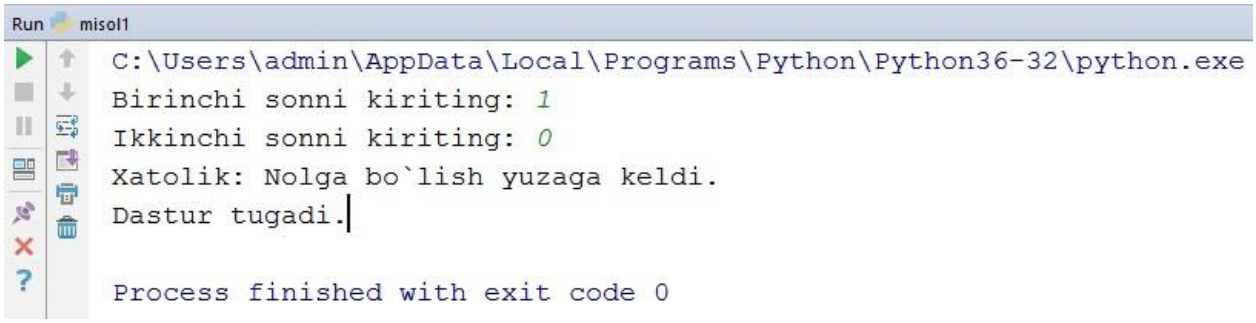
1.22.10-rasmda konsol ekranida qiymat noto'g'ri kiritilgandagi holat.

**Istisno holatini yuzaga keltirish.** Ba`zida istisno holatlarini foydalanuvchining o'zi hosil qilishga to'g'ri keladi. Buning uchun ***raise*** operatoridan foydalaniladi:

```
try:
    son1 = int(input("Birinchi sonni kiriting: "))
    son2 = int(input("Ikkinchi sonni kiriting: "))
    if son2 == 0:
        raise Exception("Nolga bo`lish yuzaga keldi.")
        print("Bo`lish natijasi:", son1 / son2)
except ValueError:
    print("O`girish muaffaqiyatsiz amalga oshirildi.")
except Exception as e:
    print("Xatolik:", e)
print("Dastur tugadi.")
```

1.22.11-rasm

Bu erda ikkinchi kiritiladgan son nolga teng bo’lsa, istisno hosil qilinadi. Istisno hosil qilinganda konsol ekraniga chiqarilishi kerak bo’lgan ma`lumotlar parametr sifatida beriladi. Yuqoridagi dasturda bu *Exception("**Nolga bo`lish yuzaga keldi.**")* tarzda yozilgan. Quyida istisno holati ro’y berganda konsol ekraniga chiqariladigan ma`lumotlarga misol sifatida keltirilgan:




1.22.12-rasm

Foydalanuvchi tomonidan hosil qilingan istisnoning ro’y bergandagi holati.

## Nazorat savollari

1.Pythonda xatoliklarni ushlashni tushuntiring.

2. Pythonda 2 ta sonni o’qib oling va birinchi sonni ikkinchi songa bo’ling. Bunda ikkinchi son 0 bo’lsa “Xatolik” deb aks holda natijani ekranga chiqaruvchi dastur tuzing.

3.Pythonda *finally* operatorini tushuntiring.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

## 24-mavzu:To’plamalar va uning metodlari

**Mavzu rejasi:**

1.To’plam tushunchasi.

2.To’plam ustida amallar.

3.Metodlar

To’plamlar elementlar majmuini ifodalashning yana bir ko’rinishi hisoblanadi. To’plamlarni aniqlash uchun figurali qavs (*‘{‘, ‘}’*) dan foydalanilib, elementlar unda ketma-ket sanaladi:

```
talabalar = {"Bobur", "Zafar", "Alisher"}
print(talabalar)  # {'Bobur', 'Zafar', 'Alisher'}
```

2.4.1-rasm

To’plamni tashkil qiluvchi elementlar qiymatlari unikal bo’lishi kerak, agar elementlar qiymatlari ayni bir xil bo’lsa, ya’ni bir xil element takrorlansa, u holda barcha takrorlanuvchi qiymatlar bitta deb hisoblanadi:

```
son = {"1", "1", "2","2", "2"}
print(son) # {'2', '1'}
```

2.4.2-rasm

Bu yerda to’plam elementlari ikkita “1”  va uchta “2” qiymatlar orqali hosil qilingan. Lekin ekranga to’plam elementlari chop qilinganda to’plam faqatgina ikkita elementdan tashkil topganligini ko’rish mumkin. To’plamni yaratish uchun *set*() funksiyasidan ham foydalanish mumkin.

Ushbu funksiyadan foydalanib to’plam yaratilganda parametriga qiymat sifatida ro’yxat yoki kortej ham berilishi mumkin:

```
tubSonlar = [2,3,5,7,11]
tubSonlarTuplami = set(tubSonlar)
print(tubSonlarTuplami) # {2, 3, 5, 7, 11}
```

2.4.3-rasm

Ayniqsa set() funksiyasi bo’sh to’plam hosil qilish uchun juda qulay hisobladi:

```
son = set()
print(son) # set()
```

2.4.4-rasm

To’plam uzunligi (to’plam elementlari soni) ni toppish uchun *len*() funksiyasidan foydalaniladi:

```
son = {3,4,5,6}
print(len(son)) # 4
```

2.4.5-rasm

**To’plamga element qo’shish**. To’plamga element qo’shish uchun *add*() metodidan foydalaniladi:

```
son = set()
son.add(2)
son.add(4)
print(son) # {2, 4}
```

2.4.6-rasm

**To’plamdan elementni o’chirish**. To’plamdan elementni o’chirish uchun *remove*() metodi qo’llanilib, uning argumentiga o’chirilishi kerak bo’lgan element beriladi. Agar o’chirilishi kerak bo’lgan element to’plamda mavjud bo’lmasa, u holda *KeyError* istisno xatoligi ro’y beradi. Shuning uchun to’plamdan elementni o’chirishdan oldin *in* operatori orqali shu elementning lug’atda mavjud yoki yo’qligini tekshirish tavsiya qilinadi:

```
ismlar = {"Anvar", "Abbos", "Abror"}
ism = " Abror"
if ism in ismlar:
    ismlar.remove(ism)
print(ismlar)  # {'Anvar', 'Abbos'}
```

2.4.7-rasm

To’plamdan elementni o’chirishning boshqa usuli ham mavjud bo’lib, *discard*() metodi orqali amalga oshiriladi. Usbu usulda element to’plamdan o’chirilganda, agar o’chirilayotgan element to’plamda mavjud bo’lmasa ham istisno xatoligi ro’y bermaydi:

```
ismlar = {"Anvar", "Abbos", "Abror"}
ism = "Abbos"
ismlar.discard(ism)
print(ismlar)  # {'Anvar', 'Abror'}
```

2.4.8-rasm

To’plamning barcha elementlarini birdaniga o’chirish uchun ya`ni to’plamni tozalash uchun *clear*() metodidan foydalaniladi:

```
ismlar = {"Anvar", "Abbos", "Abror"}
ismlar.clear()
print(ismlar)  # set()
```

2.4.9-rasm

To’plam elementlariga *for* operatori orqali murojaatni (perebor) amalga oshirish mumkin:

```
ismlar = {"Anvar", "Abbos", "Abror"}
for ism in ismlar:
    print(ism)
```

2.4.10-rasm

bu erda to’plamni har bir elementi *ism* o’zgaruvchisiga ketma-ket yuklanadi va keyingi hisoblashlarda ishlatilishi mumkin.

**To’plamlar ustuda amallar**.

To’plamlar ustuda turli xil amallarni bajarish mumkin bo’lib, ular metod va funksiyalar orqali amalga oshiriladi. Quyida ulardan eng ko’p qo’llaniladiganlarini qarab chiqamiz:

*copy*() metodi biror bir to’plamdan nusxa olish uchun ishlatiladi, masalan:

```
ismlar = {"Anvar", "Abbos", "Abror"}
ismlar2 = ismlar.copy()
print(ismlar)  # {'Abbos', 'Anvar', 'Abror'}
print(ismlar2)  # {'Abbos', 'Anvar', 'Abror'}
```

*2.4.11-rasm*

*union*() metodi ikkita to’plamni birlashtiradi va qiymat sifatida yangi to’plamni qaytaradi:

```
famil = {"Axmedov", "Niyazov"}
ism = {"Sardor", "Tohir"}
FIO = famil.union(ism)
print(FIO)  # {'Axmedov', 'Tohir', 'Sardor', 'Niyazov'}
```

2.4.12-rasm

*intersection*() metodi ikkita to’plamni kesishmasini olish uchun ishlatib, qiymat sifatida yangi to’plam qaytaradi. Ya`ni ikkita to’plam uchun umumiy bo’lgan elementlarni olish uchun *intersection()* metodi qo’llaniladi:

```
famil = {"Axmad", "Sardor", "Ikrom"}
ism = {"Sardor", "Tohir","Ikrom"}
ism2 = famil.intersection(ism)
print(ism2)  # {'Sardor', 'Ikrom'}
```

2.4.13-rasm

*intersection()* metodi o’rniga unga ekvivalent bo’lgan & (mantiqiy ko’paytirish) amalini ham qo’llash mumkin:

```
famil = {"Axmad", "Sardor", "Ikrom"}
ism = {"Sardor", "Tohir","Ikrom"}
ism2 = famil & ism
print(ism2)  # {'Sardor', 'Ikrom'}
```

2.4.14-rasm

*difference*() metodi to’plamlar ayirmasini topish uchun qo’llanilib, qiymat sifatida yangi to’plam qaytaradi. Ya`ni birinchi to’plamda mavjud va ikkinchi to’plamda yo’q bo’lgan elementlarni topishda ishlatish mumkin. *difference*() metodiga ekvivalent amal bu ‘-‘ amalidir:

```
famil = {"Axmad", "Sardor", "Ikrom"}
ism = {"Sardor", "Tohir", "Ikrom"}
ism2 = famil.difference(ism)
ism3 = famil - ism
print(ism2)  # {'Axmad'}
print(ism3)  # {'Axmad'}
```

2.4.15-rasm

*issubset*() metodi qaralayotgan to’plam boshqa to’plam (argumentida berilgan) ning qism to’plami yoki yo’qligini tekshirish uchun ishlatiladi:

```
famil = {"Axmad", "Sardor", "Ikrom"}
ism = {"Sardor", "Ikrom"}
print(ism.issubset(famil))  # True
print(famil.issubset(ism))  # False
```

2.4.16-rasm

*issuperset*() metodi qaralayotgan to’plam boshqa to’plamni (argumentida berilgan) o’z ichiga olishi yoki olmasligini tekshirish uchun ishlatiladi:

```
famil = {"Axmad", "Sardor", "Ikrom"}
ism = {"Sardor", "Ikrom"}
print(ism.issuperset(famil))  # False
print(famil.issuperset(ism))  # True
```

2.4.17-rasm

***frozenset.*** *frozenset* - o’zgartirib  bo’lmaydigan to’plamlarni yaratish uchun ishlatiladi. Ushbu turdagi to’plamga yangi element qo’shish, o’chirish yoki element qiymatini o’zgartirishga ruxsat berilmaydi. *frozenset* turidagi to’plam odatda ro’yxat, kortej yoki oddiy to’plam (*set*) orqali hosil qilinadi:

```
famil = {"Axmad", "Sardor", "Ikrom"}
fam = frozenset(famil)
print(fam)  # frozenset({'Sardor', 'Ikrom', 'Axmad'})
```

2.4.18-rasm

*frozenset* turidagi to’plamlar ustuda quyidagi amallarni bajarish mumkin:

*len*(s) – *s* to’plam uzunligi (elementlari soni)ni qaytaradi; *x in s* – *True* qiymat qaytaradi, agar *x* element s to’plamning tarkibida mavjud bo’lsa; *x not in s* – *True* qiymat qaytaradi, agar *x* element s to’plamning tarkibida mavjud bo’lmasa;

*s.issubset*(*t*) – *True* qiymat qaytaradi, agar *t* to’plam *s* to’plamni o’z ichiga olsa;

*s.issuperset*(*t*) – *True* qiymat qaytaradi, agar s to’plam t to’plamni o’z ichiga olsa;

*s.union*(*t*) – *s* va *t* to’plamlarning birlashmasidan tashkil topgan yangi to’plamni qaytaradi;

*s.intersection*(*t*) – *s* va *t* to’plamlarning kesishmasidan tashkil topgan yangi to’plamni qaytaradi;

*s.difference*(*t*) – *s* to’plamdan *t* to’plamni ayirishdan hosil bo’lgan yangi to’plamni qaytaradi;

*s.copy()* – *s* to’plamning nusxasini qaytaradi.

**Nazorat savollari**

1.Pythonda To’plamlardan qanday foydalaniladi? Misollar bilan tushuntirib bering.
2.Sizga A,B va C to’plamlar berilgan. A va B to’plamlarning kesishmasiga B va C ning ayirmasini birlashtiring.
3. Sizga A va B to’plamlar berilgan. Agar A to’plam B to’plamning qism to’plami bo’lsa A dan B ning ayirmasini, aks holda ularning kesishmasini ekranga chiqaruvchi dastur tuzing.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**25-mavzu: Fayllar va kataloglar bilan ishlash**

**Mavzu rejasi:**

1.Faylni ochish.

2.Fayllar bilan ishlash usullari.

3.OS moduli yordamida fayllarga kirish.

4.StringI() va BytesI() sinflari.

5.Fayllar va kataloglar uchun ruxsatlar

Python dasturlash tilida turli xil fayl turlari bilan ishlash imkoniyati mavjud bo’lib, shartli ravishda ularni ikki turga bo'lish mumkin: matn va binar fayllar. Matn fayllari, masalan, kengaytmasi cvs, txt, html, umuman, matn shaklida ma'lumot saqlaydigan barcha fayllarlarni o’z ichiga oladi. Binar fayllar tasvirlar, audio va video fayllar va boshqalardan iborat. Fayl turiga qarab u bilan ishlash biroz farq qilishi mumkin.

Fayllar bilan ishlaganda, quyidagi tartibdagi operatsiyalar ketma-ketligini amalga oshirish talab etiladi:

1. *open*() metodi yordamida faylni ochiladi;

2. *read*() metodi yordamida faylni o'qish yoki write() metodi yordamida faylga yozish amalga oshiriladi;
3. *close*() metodi faylni yopadi.

Fayllar bilan ishlash uchun avval faylni *open*() metodi yordamida ochish zarur. *open*() metodidan quyidagi ko’rinishda foydalaniladi**:**


Funksiyaning birinchi parametri faylning yo'lini ifodalaydi. Fayl yo'li absolyut bo'lishi mumkin, ya'ni disk harfidan boshlanadi, masalan, C://papkanomi/somefile.txt. Yoki nisbiy bo'lishi mumkin, masalan, papkanomi/somefile.txt - bu holda, fayl Python ishlaydigan skript joylashgan katologda hosil qilinadi. Ikkinchi argument *mode* - bu faylni ochish rejimi bo`lib, fayl bilan qanday ish bajarilishiga qarab, 4 turdagi fayllar bilan ishlash rejimidan birini qo’llash mumkin:

- *r* (*Read*) - Fayl o'qish uchun ochadi. Fayl topilmasa, *FileNotFoundError* xatolik qaytaradi;
- *w* (*Write*). Fayl yozish uchun ochadi. Agar fayl yo'q bo'lsa, u hosil bo'ladi. Bunday fayl allaqachon mavjud bo'lsa, u yangidan yaratiladi va shunga mos ravishda eski ma'lumotlar o'chiriladi.
- *a* (*Append*). Faylni qayta yozish uchun fayl ochiladi. Agar fayl yo'q bo'lsa, u hosil bo'ladi. Bunday fayl allaqachon mavjud bo'lsa, ma'lumotlar oxiridan yozish davom ettiriladi.
- *b* (*Binary*). Binar fayllar bilan ishlash uchun foydalaniladi. *w* va r kabi rejimlar kombinatsiyasi bilan birgalikda ishlatiladi.

Fayl bilan ishlashni tugatgandan so'ng uni *close()* metodi bilan yopish kerak bo’ladi. Ushbu metod fayl bilan bog'liq barcha resurslarni bo`shatadi.

Misol uchun, "salom.txt" matnli faylini yozish uchun ochamiz:

```
meningfaylim = open("salom.txt", "w")
meningfaylim.close()
```

Faylni ochishda yoki u bilan ishlashda turli xil istisno holatlarga duch kelish mumkin, masalan, unga ruxsat yo'q bo'lishi mumkin. Bunday holatlarda, dastur ishlash jarayonida xatolik yuz beradi va dastur bajarilishi *close()* metodi chaqirilishiga yetib bormaydi va shunga muvofiq fayl yopilmaydi. Bu kabi holatlarni oldini olish uchun istisnolardan foydalaniladi:

```
try:
    somefile = open("salom.txt", "w")
try:
    somefile.write("Salom olam")
except Exception as e:
    print(e)
finally:
    somefile.close()
except Exception as ex:
    print(ex)
```

Bu yerda, fayl bilan bajariladigan barcha amallar ketma-ketligi *try* blokida yoziladi. Agar biror bir istisno to'satdan kelib chiqsa, u holda *finally* blokida fayl blokirovka qilinadi.

Fayllar bilan ishlashning yanada qulayroq *with* konstruktsiyasi mavjud:

```
with open(file, mode) as file_obj:
    #buyruqlar
```

Bu konstruktsiya  ochiq fayl uchun *file_obj* o’zgaruvchi aniqlanadi va buyruqlar ketma-ketligi bajariladi. Ular bajarilgandan so'ng, fayl avtomatik ravishda yopiladi. Blokda amallar ketma-ketligini bajarishda istisnolar yuzaga kelsa ham, fayl avtomatik ravishda yopiladi.

*with* konstruktsiyasi yordamida, yuqoridagi misolni quyidagicha qayta yozish mumkin:

```
with open("salom.txt", "w") as somefile:
    somefile.write("Salom Python")
```

**Nazorat savollari**

1.Pythonda fayllar bilan ishlash uchun qanday metodlardan foydalaniladi?
2.Pythonda qanday fayllar bilan ishlash rejimlarini bilasiz? Ularning vazifalarini aytib bering.
3. Madhiya.txt fayliga O’zbekiston respublikasi Madhiyasining 1-bandini yozing.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

26-mavzu: **Kataloglar bilan ishlash funksiyalari**

**Mavzu rejasi:**

1.Fayllarni manipulyatsiya qilish funksiyalari.

2.Ob'ektlarni faylga saqlash

3.Kataloglar bilan ishlash funktsiyalari.

4.Scandir() funktsiyasi.

5.Fayl operatsiyalari tomonidan chiqarilgan istisnolar

Matn faylini yozish uchun ochishda *w* (qayta yozish) yoki *a* (yozuv qo’shish) rejimini qo'llaniladi. So'ngra, faylga yozish uchun *write(str)* metodidan foydalanilib, *str* parametriga yozilishi kerak bo’lgan satr uzatiladi. Shuni eslatib o'tish joizki, bu parametr satr bo’lishi shart, shuning uchun raqamlar yoki boshqa turdagi ma`lumotlarni yozish zarur bo’lsa, dastlab ularni satr turiga keltirish talab qilinadi.

"salom.txt" fayliga ba'zi ma'lumotlarni yozamiz:

```
with open("salom.txt", "w") as fayl:
    fayl.write("salom olam")
```

Joriy Python skriptining joylashgan papkasini ochsak, u yerda salom.txt faylini ko'rish mumkin. Ushbu fayl har qanday matn muharriridan ochilishi mumkin va agar kerak bo'lsa o'zgartirilishi ham mumkin. Keling, ushbu faylga yana bitta qator qo'shamiz:

```
with open("salom.txt", "a") as fayl:
    fayl.write("\nHayr, olam")
```

Agar satrni yangi qatorga yozish zarur bo’lsa, u holda, yuqoridagi kabi, yozilayotgan satr oldidan "\n"(yangi satrga o’tish belgisi) qo’yish yetarli bo’ladi. Yukunida salom.txt faylida quyidagi matn hosil bo’ladi:

*salom olam*

*Hayr, olam*

Faylga yozishning yana bir usuli - ma'lumotlarni konsolga chiqarish uchun ishlatiladigan standart *print()* metodan foydalanish orqali amalga oshiriladi:

```
with open("salom.txt", "a") as salom_fayl:
    print("Salom, olam", file=salom_fayl)
```

*print* metodi yordamida  ma'lumotlarni faylga chiqarish uchun faylning nomi *file* parametri orqali beriladi va birinchi parametr faylga yoziladigan ma‟lumotni ifodalaydi.

Fayldan o'qish. Fayldan o'qish uchun r (Read) rejimida ochiladi va uning mazmunini turli usullar bilan o'qish mumkin:

- *readline()* - fayldan bir qator o'qiydi;
- *read()* - faylning butun tarkibini bir qatorga o'qiydi;
- *readlines()* - faylning barcha satrlarini ro'yxatga oladi.

Masalan, biz yuqorida yozilgan fayllarni satrlar bo'yicha ko'rib chiqamiz:

```
with open("salom.txt", "r") as fayl:
  for satr in fayl:
        print(satr, end="")
```

Biz, albatta, har bir qatorni o'qish uchun *readline()* metidini ishlatmasak ham, har bir yangi satrni olish uchun ushbu metod avtomatik ravishda chaqiriladi. Shuning uchun ham, *readline()* metodini siklda chaqirishdan ma‟no yo’q va satrlar yangi satr "\n" belgisi bilan ajratilganligi uchun, yangi satrga chop qilish zaruryati qolmaydi va *end=""* qiymati *print* metodining ikkinchi parametri sifatida uzatiladi.

Endi satrlarni alohida o'qish uchun *readline()* metodini to’g’iridan-to’g’ri chaqiramiz:

```
with open("salom.txt", "r") as fayl:
    str1 = fayl.readline()
    print(str1, end="")
    str2 = fayl.readline()
    print(str2)
```

Konsol ekraniga quyidagi natijalar chiqariladi:

salom olam hayr, olam *readline()* metodini alohida qatordagi satrlarni o’qish uchun *while* siklida ham foydalanish mumkin:

```
with open("salom.txt", "r") as fayl:
    satr = fayl.readline()
    while satr:
        print(satr, end="")
        satr = fayl.readline()
```

Fayl kichik bo'lsa, *read*() metodidan foydalanib, uni birdan o'qishingiz mumkin:

```
with open("salom.txt", "r") as fayl:
    mazmun = fayl.read()
    print(mazmun)
```

Hamda, *readlines()* metodi yordamida fayldagi barcha satrlar ro’yxatga o’qib olinadi, ya`ni elementlari fayldagi satrlardan tashkil topgan ro’yxat hosil qilinadi:

```
with open("salom.txt", "r") as faly:
    mazmun = fayl.readlines()
    str1 = mazmun [0]
    str2 = mazmun [1]
    print(str1, end="")
    print(str2)
```

Ba’zida fayldagi ma’lumotlar ASCIIdagi belgilardan farqlanishi mumkin. Ushbu holatda fayldan berilganlarni o`qish to’g’ri bo’lishi uchun kodlash parametrini ishlatib kodlashni aniq belgilab olishimiz mumkin:

```
faylnomi = "salom.txt"
with open(faylnomi, encoding="utf8") as file:
    matn = file.read()
```

Quyidagi dastur orqali foydalanuvchi tomonidan kiritilgan satrlar massivi dastlab faylga yozish amalga oshirilgan, so’ngra ularni fayldan konsolga qayta o`qib, chop qilish amalga oshirilgan:

```
# fayl nomi
FILENAME = "habarlar.txt"
# bo’sh ro’yxat aniqlaymiz
xabarlar = list()
 for i in range(4):
    xabar = input("Satrni kiriting " + str(i + 1) + ": ")
    xabarlar.append(xabar + "\n")

# ro’yxatni faylga yozish
with open(FILENAME, "a") as fayl:
  for xabar in xabarlars:
    fayl.write(xabar)

# xabarlarni fayldan o’qiymiz
print("Xabarlarni o’qish") \
with open(FILENAME, "r") as fayl:
  for xabar in fayl:
        print(xabar, end="")
```

Dastur ishlashining namunasi:

```
Satrni kiriting 1: salom
Satrni kiriting 2: tinchlik so'zi
Satrni kiriting 3: buyuk ish
Satrni kiriting 4: Python
Xabarlarni
o'qish Salom
tinchlik so'zi
buyuk ish
Python
```

**Nazorat savollari**

1.Pythonda matnli fayllar qanday o'qiladi? Misollar bilan tushuntirib bering.

2.read() va readline() metodlarini farqi nima?. Pythondagi dasturda misollar keltiring.

3. Madhiya.txt faylidagi ma'lumotlarni o'qib, unga 2-bandni ham qo'shadigan va Madhiya2.txt fayliga yozib qo'yadigan dastur tuzing.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. тит. англ.

**27-mavzu: SQLite asoslari**

**Mavzu rejasi:**

1.Ma'lumotlar bazasini yaratish.

2.Jadval yaratish.

3.Yozuvlarni kiritish.

4.Jadval tuzilishini o'zgartirish

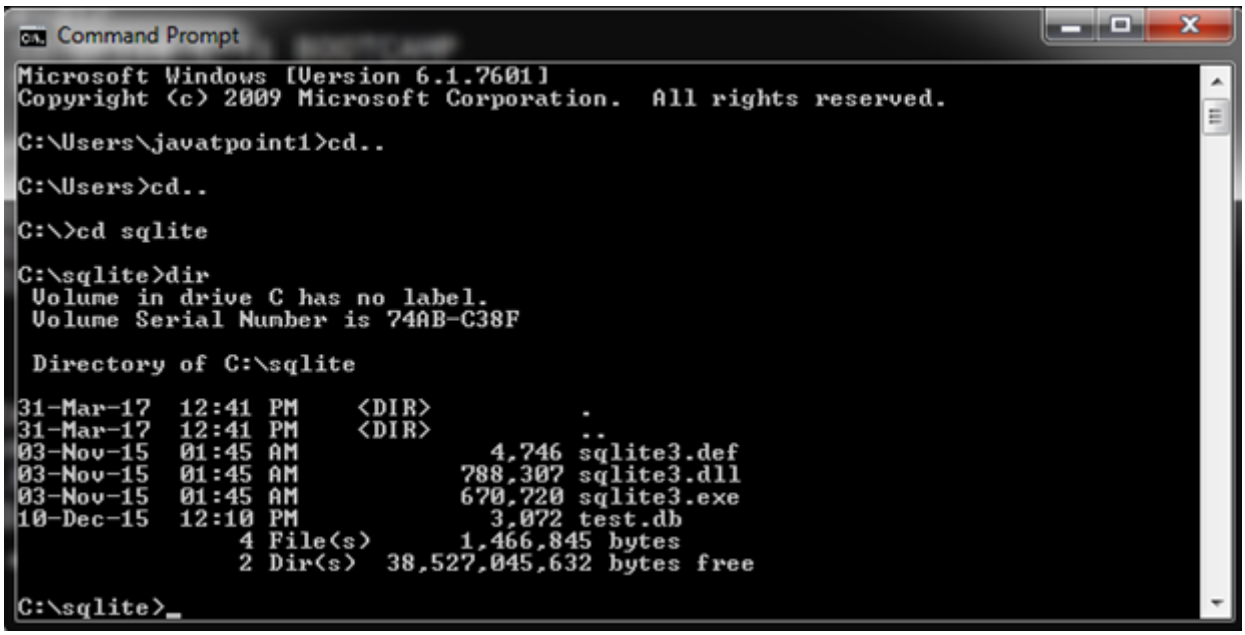
SQLite - bu kichik, tezkor, mustaqil, yuqori ishonchlilik, to'liq xususiyatli, SQL ma'lumotlar bazasi mexanizmini amalga oshiradigan C tilidagi kutubxona. SQLite - dunyodagi eng ko'p ishlatiladigan ma'lumotlar bazasi mexanizmi. SQLite barcha mobil telefonlar va ko'pgina kompyuterlarga o'rnatilgan va odamlar har kuni foydalanadigan son-sanoqsiz boshqa ilovalar ichida to'plangan. Batafsil ma'lumot...

SQLite fayl formati barqaror, o‘zaro platformali va orqaga qarab mos keladi va ishlab chiquvchilar uni 2050-yilgacha saqlab qolishga va’da berishadi. SQLite ma’lumotlar bazasi fayllari odatda tizimlar o‘rtasida boy kontentni uzatish uchun konteyner sifatida ishlatiladi va ma'lumotlar uchun uzoq muddatli arxiv formati sifatida . Faol foydalanishda 1 trilliondan ortiq (1e12) SQLite ma'lumotlar bazalari mavjud.

SQLite manba kodi ommaviy domenda bo'lib, har kim xohlagan maqsadda foydalanishi mumkin.

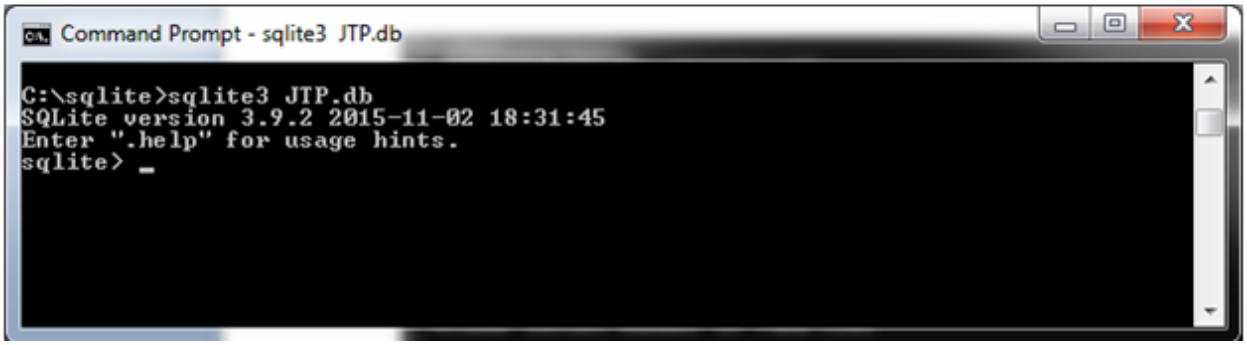
Malumotlar bazasi yaratish:
SQLite-da sqlite3 buyrug'i yangi ma'lumotlar bazasini yaratish uchun ishlatiladi.

Avval buyruq satrini oching va yo'lni belgilang. Shundan so'ng siz "dir" buyrug'i yordamida ushbu sqlite katalogini tekshirishingiz mumkin.



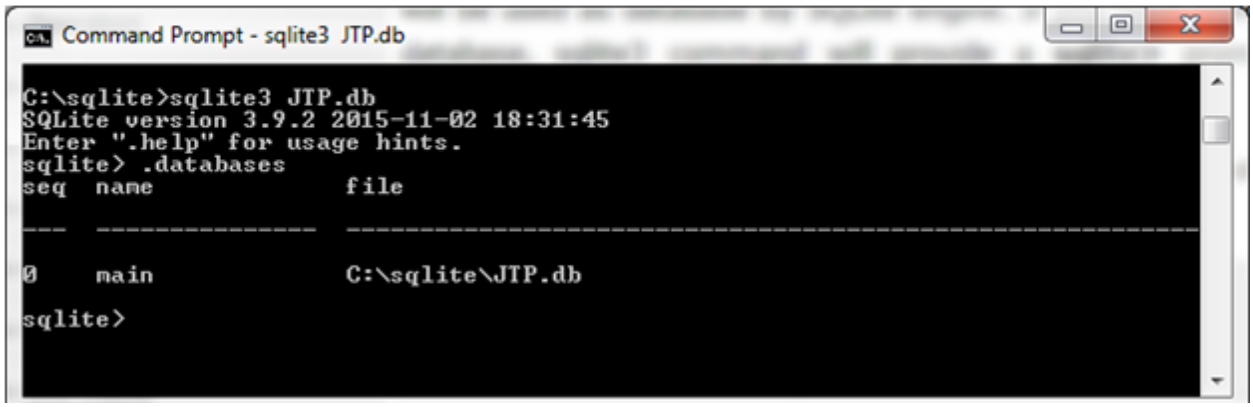

Misol uchun JTP degan database yarataylik:

```
sqlite3 JTP.db
```




Malumotlar bazasi yaraltilganini ko’rishimiz uchun :

```
.databases
```



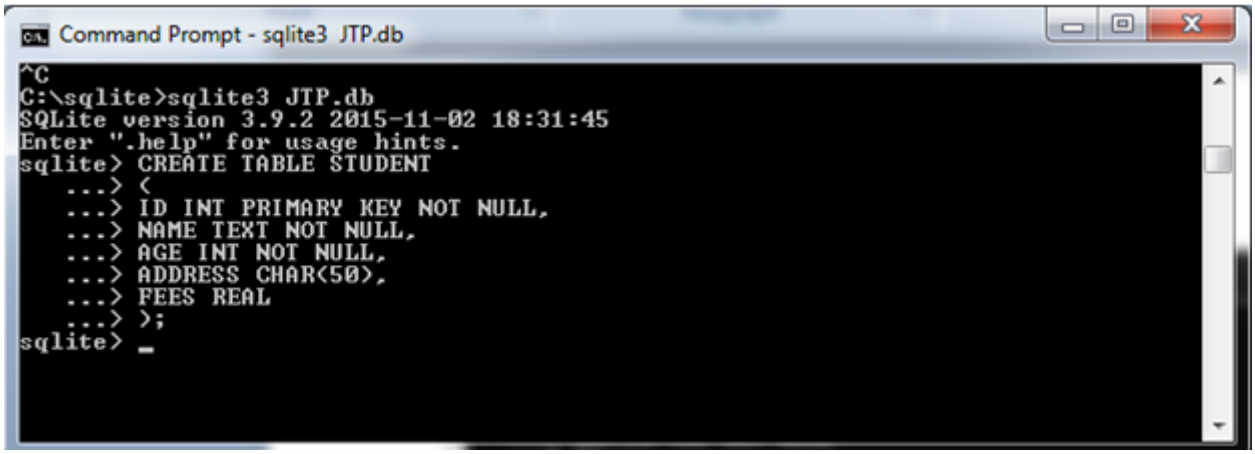

**Jadval yaratish:**

SQLite-da CREATE TABLE iborasi yangi jadval yaratish uchun ishlatiladi. Jadvalni yaratishda biz ushbu jadvalga nom beramiz va uning ustunini va har bir ustunning ma'lumotlar turlarini aniqlaymiz.

Sintaksis esa quyidagicha:

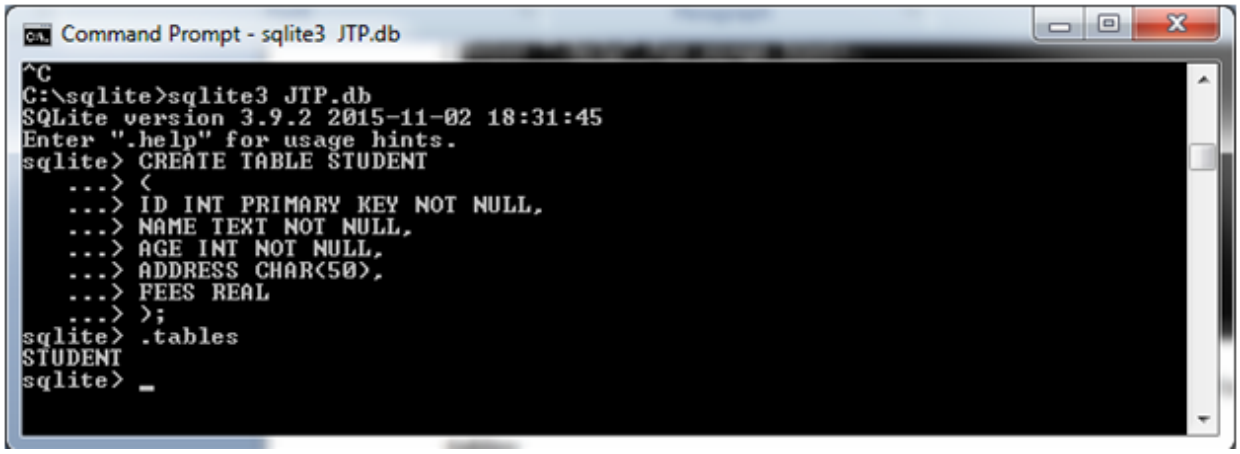
```
CREATE TABLE database_name.table_name(
   column1 datatype  PRIMARY KEY(one or more columns),
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
);
```

Keling Talaba jadvalini yaratib ko’ramiz:




Biz yaratilgan jadvallar ro’yhatini ko’rishimiz uchun:

**.tables**




**Jadvalga yozuv kiritish:**

SQLite-da INSERT INTO iborasi jadvalga yangi ma'lumotlar qatorlarini qo'shish uchun ishlatiladi. Jadval tuzilgandan so'ng ushbu buyruq jadvalga ma'lumotlarni kiritish uchun ishlatiladi.

INSERT INTO bayonoti uchun ikkita asosiy sintaksis mavjud:

**1-sintaksis:**

```
INSERT INTO TABLE_NAME [(column1, column2, column3,...columnN)]
VALUES (value1, value2, value3,...valueN);
```

**2-sintaksis:**

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

Talaba jadvaliga ma’lumot kiritish uchun:

```
INSERT INTO STUDENT (ID,NAME,AGE,ADDRESS,FEES)
VALUES (1, 'Ajeet', 27, 'Delhi', 20000.00);
INSERT INTO STUDENT (ID,NAME,AGE,ADDRESS,FEES)
VALUES (2, 'Akash', 25, 'Patna', 15000.00 );
INSERT INTO STUDENT (ID,NAME,AGE,ADDRESS,FEES)
VALUES (3, 'Mark', 23, 'USA', 2000.00 );
INSERT INTO STUDENT (ID,NAME,AGE,ADDRESS,FEES)
VALUES (4, 'Chandan', 25, 'Banglore', 65000.00 );
INSERT INTO STUDENT (ID,NAME,AGE,ADDRESS,FEES)
VALUES (5, 'Kunwar', 26, 'Agra', 25000.00 );
```

**Nazorat savollari**

1.SQLite haqida ma’lumot bering.

2..tables komandasi qanday vazifa bajaradi?

3. INSERT INTO qanday buyruq?.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**28-mavzu: Where va Having ko’rsatmalaridagi shartlar**

**Mavzu rejasi:**

1.Where va Having ko’rsatmalaridagi shartlar.

2.Indekslar.

3.O’rnatilgan so‘rovlar

4.Tranzaksiyalar(Bitimlar).

5.Jadval va ma'lumotlar bazasini o'chirish

SQLite WHERE bandi odatda SELECT, UPDATE va DELETE iboralari bilan bitta jadval yoki bir nechta jadvallardan ma’lumotlarni olishda shartni belgilash uchun ishlatiladi.

Agar shart bajarilsa yoki rost bo'lsa, u jadvaldan ma'lum qiymatni qaytaradi. Yozuvlarni filtrlash va faqat kerakli yozuvlarni olish uchun WHERE bandidan foydalanasiz.

WHERE bandi yozuvlarni filtrlash va faqat ma'lum ma'lumotlarni olish uchun ham ishlatiladi.

**Sintaksis**:

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition]
```

Misol uchun:

Yoshi 25 dan katta yoki unga teng bo‘lgan va to‘lovlar 10000.00 dan katta yoki teng bo‘lgan talabalarni tanlang.


SQLite HAVING bandi yakuniy natijalarda qaysi guruh natijalari paydo bo'lishini filtrlaydigan shartlarni belgilash uchun ishlatiladi. WHERE bandi tanlangan ustunlarga shartlar qo'yadi, HAVING bandi esa GROUP BY bandi tomonidan yaratilgan guruhlarga shartlar qo'yadi.

Misol: Nomlar soni 2 dan kam bo'lgan barcha yozuvlarni ko'rsatish


**Indekslar:**

1.Primary key:

SQLite asosiy kaliti oddiy maydon yoki yozuvni yagona aniqlash uchun ishlatiladigan maydonlar birikmasidir. Jadvalda faqat bitta asosiy kalit bo'lishi mumkin.

Birlamchi kalit NULL qiymat bo'lmasligi kerak.

Sintaksis:

```
CREATE TABLE table_name
(
column1 datatype [ NULL | NOT NULL ],
column2 datatype [ NULL | NOT NULL ],
.....
CONSTRAINT constraint_name PRIMARY KEY (pk_col1, pk_col2, ... pk_col_n)
);
```

2.Foreign  key:

SQLite Tashqi kaliti bir jadvaldagi qiymatlar boshqa jadvalda ham paydo bo'lishini belgilash uchun ishlatiladi. U SQLite ma'lumotlar bazasida ma'lumotlarning yaxlitligini ta'minlaydi. Yo'naltirilgan jadval ota-ona jadvali, tashqi kaliti bo'lgan jadval esa bolalar jadvali deb nomlanadi. Bolalar jadvalidagi tashqi kalit odatda ota-jadvaldagi asosiy kalitga murojaat qiladi.

Siz tashqi kalitni faqat CREATE TABLE iborasida belgilashingiz mumkin.

Sintaksis:

```
CREATE TABLE table_name
(
column1 datatype [ NULL | NOT NULL ],
column2 datatype [ NULL | NOT NULL ],
...
CONSTRAINT fk_column
FOREIGN KEY (column1, column2, ... column_n)
REFERENCES parent_table (column1, column2, ... column_n)
);
```

O’chirish:

SQLite'da DROP TABLE iborasi jadval ta'rifini va ushbu jadval bilan bog'liq bo'lgan barcha bog'liq ma'lumotlar, indekslar, triggerlar, cheklovlar va ruxsat spetsifikatsiyalarini olib tashlash uchun ishlatiladi.

Sintaksis:

```
DROP TABLE database_name.table_name;
```

**Nazorat savollari**

1.SQLite da where va havingni farqini tushuntiring.

2. Tranzaksiya  qanday vazifa bajaradi?

3. Jadval malumotlarini o’chirishni va jadval malumotlarini butunlay yo’qotadigan komandani mustaqil o’rganing.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**29-mavzu: Pythondan SQLite ma'lumotlar bazalariga kirish**

**Mavzu rejasi:**

1.Ma'lumotlar bazasini yaratish va ochish.

2.So'rovlarni bajarish.

3.So'rovlar natijasi bilan ishlash.

4.Tranzaksiyalarni boshqarish.

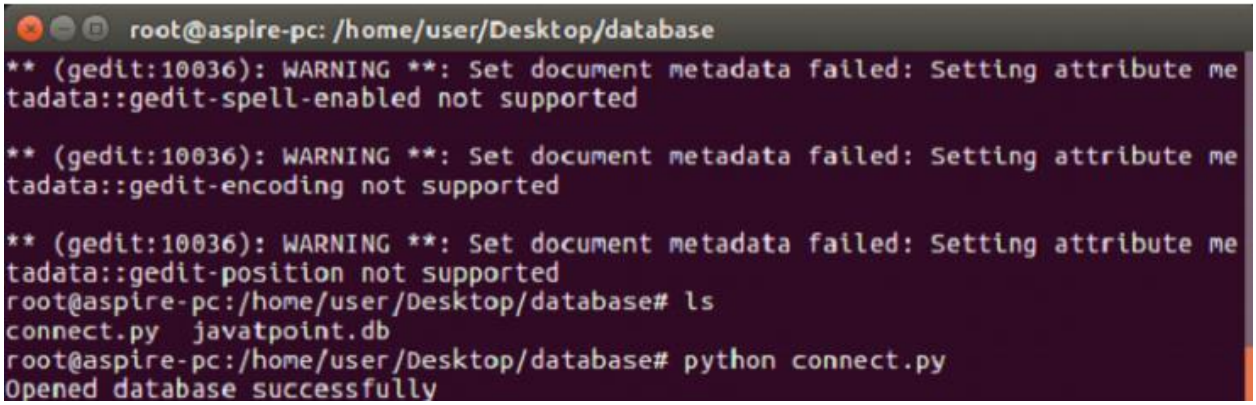
**Bog’lash:**

Quyidagi kodga ega bo'lgan "connect.py" python faylini yarating:

```
#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('javatpoint.db')

print "Opened database successfully";
```

Ushbu faylni ishlatganimizda:

```
python connect.py
```



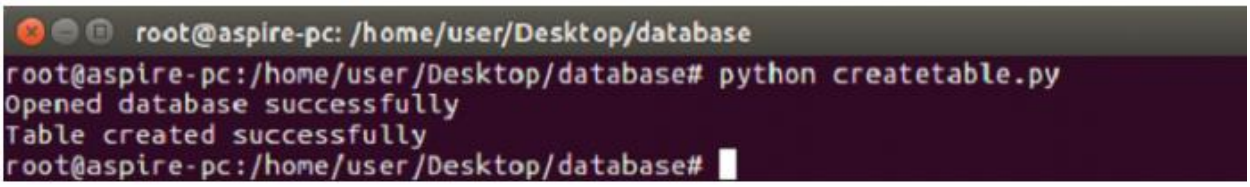

**Jadval yaratish:**

Quyidagi kodga ega bo'lgan " createtable.py" python faylini yaratamiz:

```
#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('javatpoint.db')
print "Opened database successfully";

conn.execute('''CREATE TABLE Employees
      (ID INT PRIMARY KEY     NOT NULL,
      NAME           TEXT    NOT NULL,
      AGE            INT     NOT NULL,
      ADDRESS        CHAR(50),
      SALARY         REAL);''')
print "Table created successfully";

conn.close()
```

Ushbu faylni ishlatganimizda:

```
python createtable.py
```




**Ma’lumot kiritish:**

Quyidagi kodga ega bo'lgan " connection.py" python faylini yaratamiz:

```
import sqlite3

conn = sqlite3.connect('javatpoint.db')
print "Opened database successfully";

conn.execute("'INSERT INTO Employees (ID,NAME,AGE,ADDRESS,SALARY) \
    VALUES (1, 'Ajeet', 27, 'Delhi', 20000.00 )");

conn.execute("'INSERT INTO Employees (ID,NAME,AGE,ADDRESS,SALARY) \
    VALUES (2, 'Allen', 22, 'London', 25000.00 )");

conn.execute("'INSERT INTO Employees (ID,NAME,AGE,ADDRESS,SALARY) \
    VALUES (3, 'Mark', 29, 'CA', 200000.00 )");

conn.execute("'INSERT INTO Employees (ID,NAME,AGE,ADDRESS,SALARY) \
    VALUES (4, 'Kanchan', 22, 'Ghaziabad ', 65000.00 )");

conn.commit()
print "Records inserted successfully";
conn.close()
```

Ushbu faylni ishlatganimizda:

```
python connection.py
```

```
root@aspire-pc: /home/user/Desktop/database
root@aspire-pc:/home/user/Desktop/database# python connection.py
Opened database successfully
Table created successfully
Records inserted successfully
```

## Nazorat savollari

1.Phyton va SQLite ni bog’lashni tushuntiring.

2.Phyton yordamida qanday qilib jadval yaratish mumkin?

3. Ekranga jadvaldan qaytadigan ma’lumotlarni olib kelishni ko’rsating.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд.,

перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**30-mavzu:Agregat funksiyalar**

**Mavzu rejasi:**

1.Katta-kichik harflarni hisobga olmay qidirish.

2.Agregat funktsiyalarni yaratish

3.Ma'lumotlar turini (almashtirish) konvertatsiya qilish.

4.Sana va vaqt jadvalida saqlash. Istisnolarni qayta ishlash.

5.Bajariladigan so‘rovlar (trassirovkasi) kuzatilishi.

SQLite LIKE operatori belgilar yordamida matn qiymatlarini qidirib moslashtirish uchun ishlatiladi. Qidiruv ifodasi belgi ifodasiga mos kelsa, LIKE operatori haqiqatni qaytaradi, bu 1 ga teng.

LIKE operatori bilan birgalikda ikkita belgilar qo'llaniladi:

Foiz belgisi (%)

Pastki chiziq (_)

Foiz belgisi nol, bitta yoki bir nechta raqam yoki belgilarni ifodalaydi. Pastki chiziq bitta raqam yoki belgini bildiradi.

```
SELECT FROM table_name
WHERE column LIKE 'XXXX%'
```

Yoki

```
SELECT FROM table_name
WHERE column LIKE '%XXXX%'
```

Yoki

```
SELECT FROM table_name
WHERE column LIKE '_XXXX_'
```

Agregat funksiyalar:

SQLite Agregat funktsiyalari bir nechta satrlarning qiymatlari ma'lum mezonlar bo'yicha kirish sifatida guruhlangan va chiqish sifatida bitta qiymatni tashkil etadigan funktsiyalardir. Quyida ba'zi SQLite Agregat funksiyalarining ro'yxati keltirilgan:

SQLite MIN funktsiyasi
SQLite MAX funktsiyasi
SQLite AVG funktsiyasi
SQLite COUNT funktsiyasi
SQLite SUM funktsiyasi
SQLite RANDOM funktsiyasi
SQLite ABS funktsiyasi
SQLite UPPER funktsiyasi
SQLite LOWER funktsiyasi
SQLite LENGTH funktsiyasi
SQLite sqlite_version funktsiyasi

Misol tariqasida Sum funksiyasi:

SQLite SUM funksiyasi ifodaning umumiy qiymatini qaytarish uchun ishlatiladi.

**Sintaksis**:

```
SELECT expression1, expression2, ... expression_n
SUM(aggregate_expression)
FROM tables
[WHERE conditions]
GROUP BY expression1, expression2, ... expression_n;
```

SQLite da sana va vaqt funksiyalari joriy sana va vaqtni olish uchun ishlatiladi. Sana va vaqt funksiyalari IS0-8601 sana va vaqt formatlarining quyi to'plamidan foydalanadi.

Turli formatlarda sana va vaqtni qaytaradigan va hisoblaydigan 6 xil sana va vaqt funksiyalari mavjud:

SQLite date() funktsiyasi
SQLite datetime() funktsiyasi
SQLite julianday() funktsiyasi
SQLite now() funktsiyasi
SQLite strftime() funktsiyasi
SQLite time() funksiyasi

SQLite "now" aslida funksiya emas, lekin "now" joriy sana va vaqtni olish uchun turli SQLite funksiyalarida foydalaniladigan vaqt qatori parametridir.
Sintaksis:

```
date('now')
```

Yoki

```
time('now')
```

Yoki

```
strftime(format, 'now')
i.e.
strftime('%Y-%m-%d','now')
strftime('%Y-%m-%d %H-%M','now')
strftime('%Y-%m-%d %H-%M-%S','now')
```

**Nazorat savollari**

1.SQLiteda Like operatorini tushuntiring.

2.Agregat fungsiya nima va qaysilar kiradi?

3. Hozirgi vaqtni oluvchi metod qaysi?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**31-mavzu: Python dasturlash tilidan MySQL ma'lumotlar bazalariga kirish**

**Mavzu rejasi:**

1.MySQLClient kutubxonasi.

2.So’rovlarni bajarish.

3.PyODBC kutubxonasi.

4.So’rovlarni bajarish.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое
2. необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

32-mavzu: **Python dasturlash tilida grafika bilan ishlash**

**Mavzu rejasi:**

1.Tayyor rasmlarni yuklash.

2.Yangi rasm yaratish.

3.Rasm haqida ma'lumot olish

4.Tasvirni manipulyatsiya qilish.

5.Chiziqlar va shakllarni chizish

6.Wand kutubxonasi bilan ishlash.

7.Matnni chiqarish.

8.Skrinshotlar yaratish

Grafik foydalanuvchi interfeysiga asoslangan dastur haqida gap ketganda, tasvir(lar) muhim rol o'ynaydi. Ilova belgisidan tortib animatsiyagacha foydali. Tasvirlarni teglar, tugmalar, kanvaslar va matn vidjetlarida ko'rsatish uchun tkinter paketida mavjud bo'lgan PhotoImage sinfidan foydalaniladi.

```
from tkinter import *
root = Tk()
canvas = Canvas(root, width = 300, height = 300)
canvas.pack()
img = PhotoImage(file="ball.ppm")
canvas.create_image(20,20, anchor=NW, image=img)
mainloop()
```

"PhotoImage()" funksiyasi rasm obyektini qaytaradi.


Python-da tasvirni ko'rsatish juda oddiy. Ammo muammo shundaki, PhotoImage klassi faqat GIF va PGM/PPM formatlarini qo'llab-quvvatlaydi.

Ko'proq umumlashtirilgan formatlar JPEG/JPG va PNG. Ushbu formatlarni ochish va ko'rsatish uchun bizga PIL (foto tasvirlash kutubxonasi) to'plamidan ImageTk va Image sinflari yordami kerak.

PIL (foto tasvirlash kutubxonasi) yordamida biz 30 dan ortiq formatdagi tasvirlarni yuklashimiz va ularni tasvir ob'yektlariga, hatto satrlardan base64-kodlangan GIF fayllariga aylantirishimiz mumkin!

```
from tkinter import *
from PIL import ImageTk,Image
root = Tk()
canvas = Canvas(root, width = 300, height = 300)
canvas.pack()
img = ImageTk.PhotoImage(Image.open("ball.png"))
canvas.create_image(20, 20, anchor=NW, image=img)
root.mainloop()
```

Tasvir klassi "open()" atributiga ega bo'lib, u *to'g'ridan-to'g'ri qo'llab-quvvatlanmaydigan* tasvir formatlarini ochadi va "ImageTk.PhotoImage()" bilan birga biz tasvir ob'ektini qaytarishimiz va undan foydalanishimiz mumkin.


Agar siz rasmni funktsiya ichida ko'rsatsangiz, uni global o'zgaruvchida saqlash yoki boshqa ob'ektga biriktirish orqali Python dasturingizdagi tasvir ob'ektiga havolani saqlashingiz maqsadga muvofiq.

```
global img
def foo(self):
    img = ImageTk.PhotoImage(Image.open("ball.png"))
```

```
def foo(self):
    self.img = ImageTk.PhotoImage(Image.open("ball.png"))
    self.canvas.create_image(20,20, anchor=NW, image=self.img)
    self.canvas.image = self.img
```

Bu juda muhim, chunki funktsiyadan qaytganingizda va agar tasvir ob'ekti ushbu funktsiya uchun mahalliy o'zgaruvchida saqlangan bo'lsa, tkinter tomonidan ko'rsatilgan bo'lsa ham, tasvir axlat yig'uvchi tomonidan o'chiriladi.

Tasvirni tahrirlash jarayoni tasvirni manipulyatsiya qilish deb ataladi. Tasvirlaringizni loyihalaringizda ishlatishdan oldin nima uchun ularga teginish kerak, deb hayron bo'lishingiz mumkin. Buning sabablari juda ko'p, ammo bir nechta asosiy sabablarni saqlash joyini tejash, o'qitish sifatini yaxshilash va ishlash vaqtini tezlashtirish kabi sanab o'tish mumkin. Ushbu postda ko'rib chiqiladigan manipulyatsiya usullari tasvir hajmini o'zgartirish, tasvir yorqinligi va nihoyat tasvir rangini kulrang rangga aylantirishdir. Biz ularning har birini sinab ko'rish orqali tasvir ustida amaliy mashg'ulotlar o'tkazamiz.

Rasm hajmini o‘zgartirish.Bitta rasm uchun bu unchalik katta farq qilmaydi, lekin dasturingizda minglab tasvirlarni qayta ishlash haqida o'ylaganingizda, o'lchamni ozgina o'zgartirish sizga ko'p vaqt va xotirani tejaydi.

Rasm yorqinligi. Tasvirga qaraganimizda nimani ko'rayotganimizni tushunish biz uchun oson, lekin mashinalar uchun bu biroz qiyin bo'lishi mumkin. Shunday qilib, mashinalarga tasvirni yaxshiroq qayta ishlashga yordam berish uchun biroz yorqinlik aniqlikni oshirishga yordam beradi.

Rasm kulrang shkalasi. Mashinani o'rgatishda kul rangdagi tasvirlar ancha yaxshi ishlaydi. Buning sababi, mashinalar tasvirlarni massivlar matritsasi sifatida ko'rganligi sababli, ko'p rangli bir nechta tasvir o'rniga qora va oq tasvirni saqlash osonroq.

Bular biz ushbu postda tasvirni manipulyatsiya qilish usullarining ba'zilari. Yana ko'p narsalar bor, lekin yuqorida aytib o'tilganidek, bu eng ko'p qo'llaniladigan texnikalar va har qanday tasvir formatiga qo'llanilishi mumkin. Agar tayyor bo'lsangiz, boshlaylik!

Rasm. Kodlashni boshlashdan oldin, kodlarimizni sinab ko'rmoqchi bo'lgan rasmni tanlaylik. Ushbu rasm har qanday formatda bo'lishi mumkin, lekin men "png" yoki "jpg" dan foydalanishni tavsiya qilaman. Mana men o'ynashni tanlagan rasm. Ushbu rasmning yaxshi tomoni shundaki, hech qanday filtr yoki effekt ishlatilmaydi.

Paketlar. Hammasi bo'lib faqat bitta paketdan foydalanamiz. Biz tasvirni manipulyatsiya qilish uchun foydalanadigan asosiy kutubxona PIL deb ataladi, bu tasvirni qayta ishlash kutubxonasi. PIL "yostiq" sifatida o'rnatiladi, adashmang, ular bir xil narsa.

Keyin paketni o'rnatishdan boshlaylik.

```
pip install pillow
```

Biz ulardan foydalanishimiz uchun paketni kutubxona sifatida import qilishimiz kerak.

```
from PIL import Image, ImageEnhance
```

## Tasvirni import qilish

Men rasmimni "testme.jpg" deb qayta nomladim va bu bosqichda uni "img" deb nomlangan o'zgaruvchi nomiga belgilash orqali uni import qilaman. Biz eng ko'p ishlatiladigan uchta manipulyatsiya texnikasini qilamiz. Ishlarni toza va sodda qilish uchun men har bir texnika uchun bir xil tasvirdan foydalanaman, lekin har bir manipulyatsiya texnikasi uchun turli xil tasvirlarni belgilashga qodirman.

```
img = Image.open("testme.jpg")
```

Tasvirni import qilish .Men rasmimni "testme.jpg" deb qayta nomladim va bu bosqichda uni "img" deb nomlangan o'zgaruvchi nomiga belgilash orqali uni import qilaman. Biz eng ko'p ishlatiladigan uchta manipulyatsiya texnikasini qilamiz. Ishlarni toza va sodda qilish uchun men har bir texnika uchun bir xil tasvirdan foydalanaman, lekin har bir manipulyatsiya texnikasi uchun turli xil tasvirlarni belgilashga qodirman.

```
print(img.size)
```

Rasm o’lchamini o’zgartirish.

```
newsize = (300, 300)
img_resized = img.resize(newsize)
print(img_resized.size)
```

Rasmni kerakli papkaga saqlaymiz.

```
img_resized.save("resized.jpg")
```

Asosiy Tkinter oynamizda chiziqlar yaratish uchun biz oynada chiziqni joylashtirish uchun koordinatalarni oladigan **create_line()** usulidan foydalanamiz. Ushbu koordinatalar chiziqning uzunligi va yo'nalishini belgilaydi.

Tkinterda har qanday turdagi chiziqni yaratish juda oson. To'g'ri chizish uchun create_line() usulidan foydalanamiz.

```
canvas.create_line(15, 25, 200, 25, width=5)
canvas.pack()
```

Nuqtali chiziqni yaratish tartibi to'g'ri chiziq bilan bir xil. Xuddi shunday, biz create_line() usulidan foydalanamiz va chiziq koordinatasini o'tkazamiz, yagona o'zgarish shundaki, biz yana boshqa parametr chizig'ini qo'shamiz.

```
canvas.create_line(300, 35, 300, 200, dash=(10), width=5)
canvas.pack()
```

Biz muhokama qilganimizdek, biz bir nechta chiziqlar yaratish orqali turli shakllarni chizishimizga imkon beradigan chiziqlar **yo'nalishini ham nazorat qilishimiz mumkin.** Berilgan kodda biz uchta chiziqning 3 ta koordinatasini uchburchak hosil qiladigan tarzda oldik.

```
canvas.create_line(55, 85, 155, 85, 105, 180, 55, 85, width=5)
canvas.pack()
```

**Wand** - bu python uchun Imagick kutubxonasi. U Python 2.6, 2.7, 3.3+ va PyPy versiyalarida Imagick API funksiyalarini qo‘llab-quvvatlaydi. Bu kutubxona nafaqat tasvirlarni qayta ishlashga yordam beradi, balki NumPy yordamida Machine Learning kodlari uchun qimmatli funksiyalarni ham taqdim etadi.

O’rnatilishi:

```
pip install Wand
```

Tasvirni yaratish:

```
from wand.image import Image

with Image(filename ='salom.png') as pic:
    print('Uzunlik:', pic.width)
    print('Balandlik:', pic.height)
```

Tasvirni xiralashtirish:

```
from wand.image import Image

with Image(filename ="salom.png") as pic:

    pic.blur(radius = 0, sigma = 3)

    pic.save(filename ="blur1.png")
```

Chizma(aylana) chizish:

Python keng qo'llaniladigan umumiy maqsadli tildir. Bu turli xil vazifalarni bajarishga imkon beradi. Ulardan biri ekran tasvirini olishi mumkin. U skrinshotni olish uchun ishlatilishi mumkin bo'lgan **pyautogui** nomli modulni taqdim etadi . Ushbu modul tasvirlarni manipulyatsiya qilish NumPyva OpenCVsaqlash usuli bilan birga taqdim etadi (bu holda skrinshot) Python keng qo'llaniladigan umumiy maqsadli tildir. Bu turli xil vazifalarni bajarishga imkon beradi. Ulardan biri ekran tasvirini olishi mumkin. U skrinshotni olish uchun ishlatilishi mumkin bo'lgan **pyautogui** nomli modulni taqdim etadi . Ushbu modul tasvirlarni manipulyatsiya qilish NumPyva OpenCVsaqlash usuli bilan birga taqdim etadi (bu

holda skrinshot)

```
from wand.image import Image
from wand.drawing import Drawing
from wand.color import Color

with Drawing() as draw:
    draw.stroke_color = Color('black')

    draw.stroke_width = 2

    draw.fill_color = Color('white')

    draw.circle((200, 200),
                (100, 100))
    with Image(width = 400, height = 400,
            background = Color('lightgreen')) as pic:
        draw(pic)
        pic.save(filename ='circle1.jpg')
```

Python keng qo'llaniladigan umumiy maqsadli tildir. Bu turli xil vazifalarni bajarishga imkon beradi. Ulardan biri ekran tasvirini olishi mumkin. U skrinshotni olish uchun ishlatilishi mumkin bo'lgan **pyautogui** nomli modulni taqdim etadi . Ushbu modul tasvirlarni manipulyatsiya qilish NumPyva OpenCVsaqlash usuli bilan birga taqdim etadi (bu holda skrinshot)

pyautoguirasmlarni PIL (python tasvir kutubxonasi) sifatida oladi, bu ko'plab turli xil tasvir fayl formatlarini ochish, boshqarish va saqlashni qo'llab-quvvatlaydi. Keyin rasmni RGB dan BGR ga aylantirish uchun NumPy massiviga aylantirishimiz kerak, chunki tasvir fayli OpenCV yordamida o'qilganda imread()ranglar tartibi BGR (ko'k, yashil, qizil) bo'lishi kerak.

Kutubxonalarni o’rnatamiz.

```
pip install numpy
pip install pyautogui
pip install opencv-python
```

```
import numpy as np
import cv2
import pyautogui

image = pyautogui.screenshot()

image = cv2.cvtColor(np.array(image),
                     cv2.COLOR_RGB2BGR)

cv2.imwrite("image1.png", image)
```

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**33-mavzu: Internet - dasturlash**

**Mavzu rejasi:**

1.URLmanzilni tahlil qilish.

2.So'rovlar qatorini kodlash va dekodlash.

3.Nisbiy URLni Absolutega aylantirish.

4.HTML ekvivalentlarini tahlil qilish.

5.HTTP  protakollari bo’yicha ma’lumotlar almashinuvi

6.JSON formatidagi ma'lumotlari bilan ishlash

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

34-mavzu: **Tkinter kutubxonasi Oyna ilovalarini ishlab chiqish asoslari**

**Mavzu rejasi:**

1.Tkinterda birinchi dasturni yaratish  va tahlil qilish.

2.Komponentlarni ma'lumotlar bilan bog'lash.

3.Meta o'zgaruvchilar.

4.Hodisalarni qayta ishlash va boshqarish.

6.Hodisa yaratish

Hozirgi kunda ko'plab dasturlarda konsolga qaraganda intuitiv va foydalanuvchilar uchun qulay bo'lgan grafik interfeys ishlatiladi va Python dasturlash tili bilan grafik dasturlarni ham yaratish mumkin. Buning uchun Python maxsus vositalar ─ tkinter deb nomlangan komponentlar to'plamidan foydalanadi.

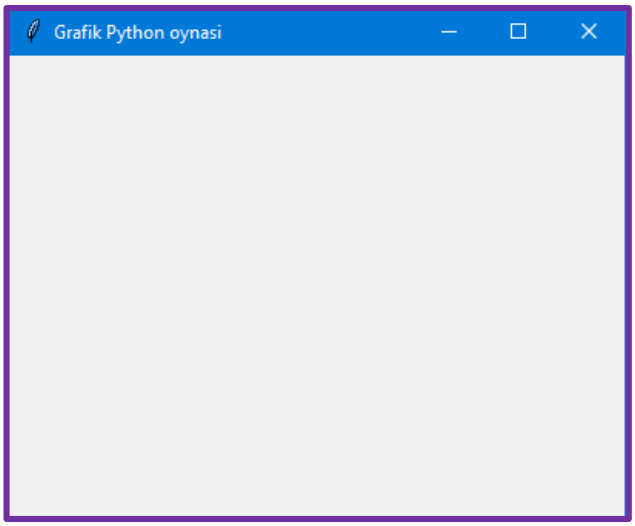
Tkinter ─ Python uchun standart GUI(Graphical User Interface ─ Grafik foydalanuvchi interfeysi) kutubxonasi. Pythonni o‘rnatganda kutubxona dasturning ichida birga taqdim etiladi. Python Tkinter bilan birgalikda GUI dasturlarini yaratishning tez va oson usulini taqdim etadi.

## Oyna yaratish va uning parametrlari

Grafik dasturlarni tuzishda asosiy nuqta ─ bu oyna yaratishdir. Keyin boshqa barcha GUI komponentlari oynaga qo'shiladi. Keling, avval oddiy oyna yaratish kodini ko'rib chiqaylik:

```
from tkinter import* # tkinter modulining chaqirilishi

root = Tk()

root.title("Grafik Python oynasi") #oyna sarlavhasi kiritish

root.geometry("400x300") #oyna o’lchamini belgilash
```

Kodni tahlil qilamiz. Dastlabki satr izohda aytilganidek tkinter modulini chaqiradi. Biz grafik interfeys hosil qilishimiz uchun doimo tkinter modulini chaqirib olishimiz kerak. Aks holda biz yozgan kodimizdan hech qanday natija ololmaymiz. Grafik oynani yaratish uchun tkinter modulida aniqlangan **Tk()** konstruktoridan foydalaniladi. Yaratilayotgan oyna **ildiz o'zgaruvchi**siga beriladi va shu o'zgaruvchi orqali biz oynaning atributlarini boshqarishimiz mumkin. Yuqoridagi kodda o’zgaruvchi ‘root’ deb nomlangan. Oyna sarlavhasini o'rnatish uchun **title()** buyrug’idan foydalanamiz. **geometry()** buyrug’i yordamida oynaning o'lchamini o'rnatish mumkin. Agar dastur oynasi yaratilganda geometry() buyrug’i chaqirilmasa, u holda ichki tarkibni joylashtirish uchun zarur bo'lgan joy oynani egallaydi. Oynani aks ettirish uchun **mainloop()** buyrug’i ishlatiladi. Yuqoridagi kodni ishga tushirganimizda ekranda 400x300 o’lchamga ega bo’lgan, “Grafik Python oynasi” nomli bo’sh oyna hosil bo’ladi:




Tkinter asboblar to'plami barcha kerakli grafik komponentlar: tugmalar, matn maydonlari va shu kabi turli xil boshqaruv elementlarini o’z ichiga olgan modul hisoblanadi. Ushbu boshqaruv elementlari odatda vijetlar deb nomlanadi. Bugungi kunda tkinter modulida 21 ta vijet mavjud:

| |
|---|
| Bitmap Class vijeti |
| Button vijeti |
| Canvas vijeti |
| Checkbutton vijeti |
| Entry vijeti |
| Frame vijeti |
| Image Class vijeti |
| Label vijeti |

| |
|---|
| LabelFrame vijeti |
| Listbox vijeti |
| Menu vijeti |
| Menubutton vijeti |
| Message vijeti |
| OptionMenu vijeti |
| PanedWindow vijeti |
| Radiobutton vijeti |

| |
|---|
| Scale vijeti |
| Scrollbar vijeti |
| Spinbox vijeti |
| Text vijeti |
| Toplevel vijeti |

Ularga tegishli parametr va xususiyatlarni birma bir izohlashga va misollar orqali ko’rsatishga harakat qilamiz.

Hodisa foydalanuvchi tomonidan sichqoncha bilan bajariladigan operatsiya yoki “ishlab chiqaruvchi” **funksiyasi hodisa** obyekti bilan chaqiriladi, deyishimiz mumkin . U ular bilan bog'liq barcha funktsiyalarni bajara oladi.

**Quyidagi kodda biz = “Chiqishni** bosganingizda” matnini berib, tugma yaratamiz . Tugmani bosish orqali ular Python qo'llanmalarini chop etadilar va chiqish buyrug'i ham bajariladi.

- **Button()** buyruqni bajarish uchun ishlatiladi
- **Button.focus()** ma'lum bir hodisaga diqqatni jalb qilish uchun ishlatiladi.

```
from tkinter import *

def task(event):
    print("Python yo'lboshchi")

ws=Tk()
ws.geometry("200×200")

button=Button(ws,text="Bosilganda chiqish",command= ws.quit)

button.focus()
button.pack(expand=True)
button.bind('<Button-1>', task)

ws.mainloop()
```

Yuqoridagi kodni ishga tushirgandan so'ng biz tugma o'rnatilgan oyna paydo bo'lishini ko'rishimiz mumkin. Foydalanuvchi tugmani bosgandan so'ng Python qo'llanmasini chop etadi va avtomatik ravishda o'chiriladi. Voqea sodir bo'lganda, ajratilgan funksiya avtomatik ravishda chaqiriladi.

Ushbu bo'limda biz Python Tkinter-da hodisani qanday bog'lash haqida bilib olamiz.

Funktsiyani hodisaga ajratish hodisani bog'lash deb nomlanadi. Voqea sodir bo'lganda, ajratilgan funksiya avtomatik ravishda chaqiriladi.

bind() usulining sintaksisi

```
widget.bind(event,handler,add=None)
```

Quyidagi kodda biz voqea sodir bo'lishi mumkin bo'lgan vidjet yaratishimiz mumkin. Agar foydalanuvchi tugmani bir marta bosgan bo'lsa, biz vidjetda tugma yaratamiz, agar " **Salom** " tugmachasini bosgandan so'ng matn ko'rsatiladi. Agar foydalanuvchi tugmani ikki marta bosgan bo'lsa, dasturning bajarilishi to'xtatiladi.

```
from tkinter import *
def task(event):
    print("Salom")
def quit(event):
    print("To'xtash uchun 2 marta bosing")
    import sys; sys.exit()

ws=Tk()
ws.title("Python yo'lboshchi")
ws.geometry("200×200")

button = Button(ws, text='Bosish')
button.pack(pady=10)
button.bind('<Button-1>', task)
button.bind('<Double-1>', quit)
button.mainloop()
```

ws.title() vidjetga sarlavha berish uchun ishlatiladi.

sys mavjud Python moduliga nom beradigan funktsiyadir

Natija:

Yuqoridagi kodni ishga tushirgandan so'ng biz vidjetni ko'rishimiz mumkin bo'lgan quyidagi natijani olamiz. Vidjetda Salom chop etilgandan so'ng tugmani bosgandan so'ng tugma joylashtiriladi. shundan so'ng yana ikki marta bosish orqali ular barcha dasturlardan chiqishadi.

Ushbu bo'limda biz Python Tkinter-da event_generate tomonidan qanday qilib hodisa yaratishimiz mumkinligini bilib olamiz .

Oddiy so'zlar bilan aytganda, biz Generate so'zidan nimanidir vujudga keltirishni tushunamiz. Hatto biz ishlab chiqarishni ishlab chiqarish yoki yaratish uchun ishlatilishini ham tushunamiz.

Mana, bu event_generate da biz ba'zi harakatlarni ko'rsatadigan va mavjudlikka nimadir olib keladigan hodisani yaratdik.event_generate hodisani qayta chaqiruvchi standart jarayondir.

Quyidagilarning sintaksisi:

```
widget.event_generate(sequence,when='tail')
```

Quyidagi kodda biz vaqt doimiy ravishda o'zgarib turadigan vidjet yaratamiz. **event_getate when = 'tail'** atributi bilan ishlab chiqaradi . **tail** har qanday hodisalar qayta ishlanganidan keyin uni voqea navbatiga qo'shishdir.

```
import threading
import time
import queue
from tkinter import *

ws = Tk()

ws.geometry("200×200")

comque= queue.Queue()

def timeThread():
    prestime = 0
    while 1:

        comque.put(prestime)

        try:
            ws.event_generate('<<TimeChanged>>', when='tail')

        except TclError:
            break

        time.sleep(1)
        prestime += 1


clcvar = IntVar()
Label(ws, textvariable=clcvar, width=9).pack(pady=10)


def timeChanged(event):
    clcvar.set(comque.get())

ws.bind('<<TimeChanged>>', timeChanged)


Thr=threading.Thread(target=timeThread)
Thr.start()

ws.mainloop()
```

Ushbu keyingi chiqishda biz ko'rib turganimizdek soniyalarda doimiy ravishda o'zgarib turadigan vaqt hosil bo'ladi.

**Nazorat savollari**

1.Tkinter nima va u qanday maqsadlarda ishlatiladi?

2.*Vijet* nima va qanday maqsadlarda foydalaniladi.

3.450x380 o'lchamli va *"Assalomu Alaykum Ona Vatan"* sarlavhali GUI oynasini yaratuvchi dastur tuzing.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 720 с. : ил. — Парал. тит. англ.

35-mavzu: **Komponentlardan foydalanish**

**Mavzu rejasi:**

1.Komponentlar uchun opsiyalarni belgilash.

2.Komponentlarni konteynerlarga joylashtirish.

3.Oynalar bilan ishlash.

4.Ilovaning hayot tsiklini boshqarish

5.Qayta ishlashda xatolik

Tkinterda har bir *komponent* sinf bo'lib, komponentni yaratish aslida sinfni ishga tushiradi. Instantiatsiya jarayonida siz konstruktor orqali komponent uchun ba'zi xususiyatlarni o'rnatishingiz mumkin, shuningdek, komponentga *ota-konteynerni* ko'rsatishingiz kerak , ya'ni komponent qaerda joylashganligini bildiradi.

Nihoyat, shuningdek, uni qaerga qo'yish masalasini hal qiladigan komponentlar uchun geometriya menejerini (tartib boshqaruvchisi) o'rnatish kerak, shuningdek, uni qanday qo'yish masalasini hal qilish kerak. Joylashtirish menejeri uni qanday qo'yish, ya'ni asosiy konteynerda bola komponentlarini joylashtirish masalasini hal qiladi.

Foydalanuvchining bosganiga javob berish uchun oddiy tugma. Python funktsiyasi bilan bog'lanishi mumkin, bu tugma bosilganda avtomatik ravishda chaqiriladi.

```
B = Button ( master, option=value, ... )
```

Ushbu parametrlar haqida ko'proq ma'lumot olish uchun:

- master: tugmani olib yuradigan ota-konteyner.
- variantlar: ixtiyoriy, ya'ni tugmaning o'rnatiladigan xususiyatlari.

```
from tkinter import *

def onclick():
    print("Bosildi !!!")

root = Tk()

Button = Button(root,text = 'This is a button ',fg = 'red',command = onclick)

Button.pack()

root.mainloop()
```

Tez-tez ishlatiladigan tugma atributlari:

- Komponent joylashtirilganda matn satri tugmachasining **matn** tarkibi faol bo'ladi
- Atributlar **bg** tugmaning fon rangi faol, **fg** old fon rangi
- **bd** tugma chegarasining kengligi pikseldir. Standart qiymat 2 pikselli chegara o'lchamidir. Standart qiymat tugma chegarasining bg (fon) o'lchamidir.
- tugma bosilganda tugma bilan bog'langan funktsiyaga **buyruq bering**

*Yorliq*

Yorliq Assambleyasi. Asosan displey funktsiyasini amalga oshirish uchun ishlatiladi, matn va rasmlarni ko'rsatishi mumkin.

```
l = Label ( master, option=value ... )
```

Uni yaratish usuli va tugmalarni bosish bir xil. Aytish mumkinki, barcha komponentlar shu tarzda yaratilgan. *** Label atributi bevosita tugmaga murojaat qilishi mumkin. Aslida, tugma maxsus Labeldir, lekin tugma javobni bosish funktsiyasiga ega.

```
from tkinter import *

root = Tk()

label_1 = Label(root,text="Salom")
label_2 = Label(root,bitmap="error")
label_1.pack()
label_2.pack()
root.mainloop()
```

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

36-mavzu: **Tkinter kutubxonasi. Frame, Button va Entry komponentlari**

**Mavzu rejasi:**

1.Komponentlar.

2.Frame komponenti.

3.Button komponenti.

4.Entry komponenti

**Nazorat savollari**

1. *Checkbutton()* elementining ishlashini misollar bilan birgalikda tushuntirib bering.
2. *Entry* vidjetining maqsad va vazifalarini misollar yordamida tushuntirib bering.
3. *Checkbutton()* dan foydalangan holda Badiiy asarlar, Fantastik asalar, Ilmiy asarlar .... kabilardan iborat inputlarni turli ranglarda chiqaruvchi va har bir tugma bosilganda fon rangini sariq, tugma matni rangini qizil qilib ko’rsatuvchi dastur tuzing.
4.Faqat qo’shish va ayrish amali uchun ishlaydigan kalkulyator dasturini tuzing.

## 8.7.Frame va Label vijeti

Frame vijeti boshqa vijetlarni qandaydir do'stona tarzda guruhlash va tartibga solish jarayonida juda muhimdir. U boshqa vijetlarning joylashishini tartibga solish uchun javob beradigan konteyner kabi ishlaydi. U tartibni tashkil qilish va ushbu vijetlarning to'ldirilishini ta'minlash uchun ekrandagi to'rtburchak maydonlardan foydalanadi. Kadr, shuningdek, murakkab vijetlarni amalga oshirish uchun poydevor sinfi sifatida ham foydalanish mumkin. Ushbu vijetni yaratish uchun oddiy sintaksis :

*Frame (master, xususiyat=qiymat, ...)*

Jadvalda vijet uchun eng ko'p ishlatiladigan xossalarining ro'yxati keltirilgan. Ushbu parametrlar vergul bilan ajratilgan *xususiyat=qiymat* juftlari sifatida ishlatilishi mumkin.

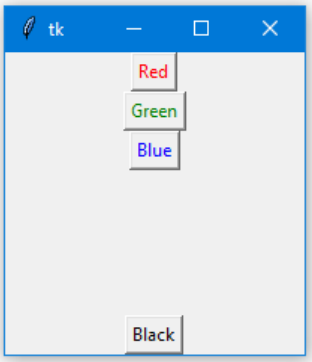
| **Parametrlar** | **Tavsif** | **Qiymat** |
|---|---|---|
| bg | Oddiy fon rangi yorliq va indikator orqasida ko'rsatiladi | ="red"<br>="#555" |

| | | |
|---|---|---|
| bd | Ko'rsatkich atrofidagi chegara kengligi | =2<br>=15 |
| cursor | Belgilangan sohada sichqoncha kursorining ko’rinishini o’zgartirish | ="arrow"<br>="dot"<br>=”plus” |
| height | Yangi ramkaning vertikal o'lchamlari | =20<br>=15 |
| highlightbackground | Fokus bo'lmasa, fokusning rangi ko’rsatiladi | ="red"<br>="#555" |
| highlightcolor | Agar ramka fokusga ega bo'lsa, u fokusda ko'rsatilgan rang | ="red"<br>="#555" |
| highlightthickness | Fokusning qalinligi | =2<br>=5 |
| width | Yangi ramkaning gorizontal o'lchamlari | =20<br>=15 |

Frame vijetining ishlatilish o’rnini quyidagi misolda ko’rishimiz mumkin:

main.py

```
from tkinter import *
root = Tk()
root.geometry("200x200")
frame = Frame(root, cursor="dot")
frame.pack()
bottomframe = Frame(root, cursor="plus")
bottomframe.pack(side=BOTTOM)
redbutton = Button(frame, text="Red", fg="red").pack()
greenbutton = Button(frame, text="Green", fg="green").pack()
bluebutton = Button(frame, text="Blue", fg="blue").pack()
blackbutton = Button(bottomframe, text="Black", fg="black").pack()
root.mainloop()

```




Dastur kodini ishlatish natijasida ekaranda 4ta tugma aks etadi. Frame metodning dasturdagi o'rni sichqoncha kursori tugmalar egallagan sohada boshqa shaklga o'zgaradi.

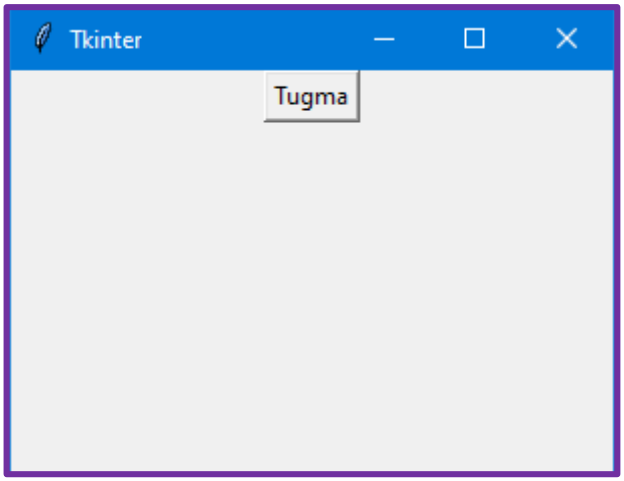
Button ─ tugma elementi hisoblanadi. Bu orqali biz yaratayotgan oynamizga turli xildagi tugmalarni joylashtirishimiz mumkin bo'ladi. Uning hodisa va xususiyatlarini o'zimiz belgilashimiz mumkin. Masalan tugma faollashtirilganda avtomatik ravishda biror funksiya yoki metod chaqirilishi mumkin. Oynaga tugma qo'shish sintaksisi quyidagicha:

*Button (master, xususiyat=qiymat, ...)*

**Button()** konstruktori tugma yaratish uchun ishlatiladi. **Master** sifatida bizga kerakli oyna uchun e'lon qilingan o'zgaruvchi nomi olinadi. Agar kodda bitta oyna yaratilgan bo'lsa, birinchi parametrni ko'rsatmasligimiz mumkin. Agar kodimizda bir nechta oyna ifodalangan bo'lsa, biz kerakli oynaga havolani Button konstruktorida belgilab qo'yishimiz kerak. **Xususiyat**ga yaratilayotgan tugmaning parametrlari: rangi, o'lchamlari, nomlanishi va hokazolar belgilanadi va qiymat beriladi. Soddaroq misol orqali ko'raylik:

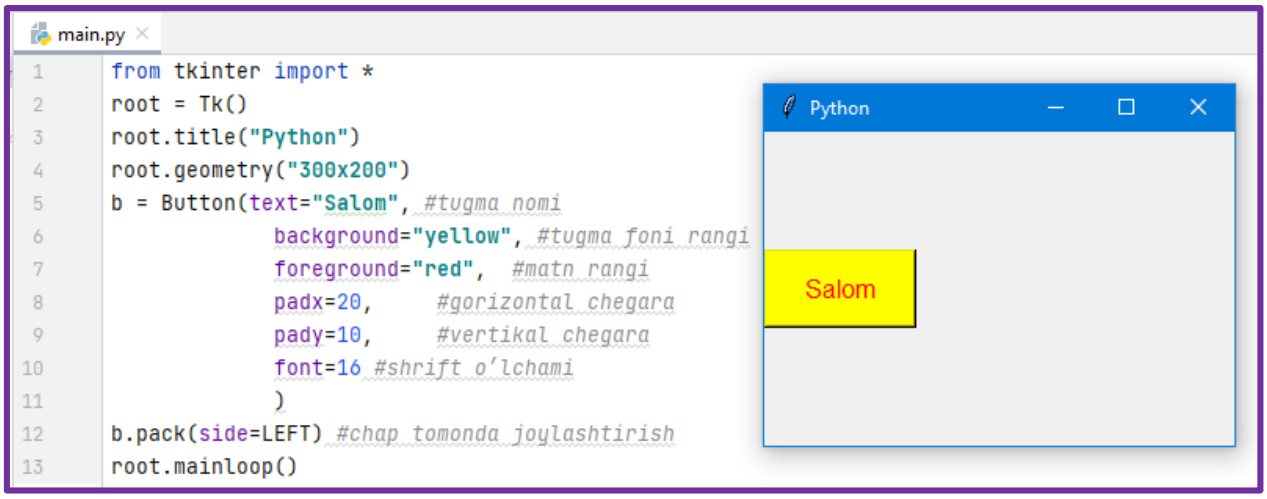
```
from tkinter import *

top = Tk()

top.title("Tkinter")

top.geometry("300x200")

b=Button(text="Tugma")

b.pack()
```

Yuqoridagi kodni ishlatganimizda 300x200 o’lchamli “Tkinter” deb nomlangan oyna va uning yuqori qismida “Tugma” so’zi aks etgan tugma hosil bo’ladi.




Oyna parametrlari bilan yuqorida tanishgandik. Endi tugmaning xususiyatlari bilan yaqindan tanishib chiqamiz. Avvalo, tugmani ixtiyoriy bir o’zgaruvchi (“b”)ga qiymat qilib berdik. **Button()** konstruktorida biz birinchi parametr(ya’ni bizning misolimizda “top”)ni ko’rsatmadik, sababi kodimizda faqat bitta oyna mavjud. Biz bu parametrni yozsak ham, yozmasak ham konstruktor joriy oynani tushunadi. **text** parametridan foydalanib tugma matnini o'rnatdik. Elementni ko’rsatish uchun **pack()** metodi chaqirildi. Bu metod vijetlarning oynada joylashish o’rnini belgilash imkoniyatini beradi. Metod parametrsiz holatda tugma ekranning yuqori markazida joylashadi. Qo’llanma oxirida shu kabi metodlar haqida kengroq ma’lumotlarni keltiramiz.

Keyingi misolda tugma uchun bir necha parametrlarni qo’llaymiz:




Quyidagi jadvalda tugma xususiyatlarini tavsiflab, qabul qilishi mumkin bo'lgan bir necha qiymatlarini keltiramiz:

| **Parametr** | **Tavsif** | **Qiymat** |
|---|---|---|
| activebackground | Tugma bosilganda orqa fon rangi | ="red", =”#000” |
| activeforeground | Tugma bosilganda matn rangi | =”red”,=”#555”, |
| bd | Chegaraning kengligi(piksellarda) | =2, =15 |
| bg | Oddiy fon rangi | =”blue”, =”#456” |
| command | Tugma bosilganda chaqiriladigan funksiya yoki protsedura | =myFunction |
| fg | Matn rangi | =”blue”, =”#456” |
| font | Tugma yorlig'i uchun ishlatiladigan matn shrifti | =”Arial 15”,=”14” =”Calibri 10” |
| height | Tugmachaning balandligi | =25,=5 |

| | | |
|---|---|---|
| highlightcolor | Vijet fokusga ega bo'lganda fokusning rangi | =”blue”, =”#456” |
| image | Tugmachada ko'rsatiladigan rasm (matn o'rniga) | |
| justify | Matn satrlarining tugmada joylashish holati | =LEFT,<br>=CENTER,<br>=RIGHT |
| padx | Matnning chap va o'ng tomonlariga qo'shimcha to'ldirish | =20, =15 |
| pady | Matnning yuqorisi va ostiga qo'shimcha to’ldirish | =5, =10 |
| relief | Chegara turini belgilaydi | = SUNKEN,<br>=RAISED,<br>=GROOVE,<br>=RIDGE |
| state | Tugmani aktiv holatda ishlashni boshlashi va unga bo’lgan murojaatni cheklash uchun ishlatiladi | = DISABLED,<br>= ACTIVE,<br>=NORMAL |
| underline | Tegishli matn belgisi ostiga chizish. Agar manfiy bo'lsa, tugmachadagi matnning biron bir belgisi ostiga chizilmaydi | =4,<br>=0 |
| width | Tugmaning kengligi | =25,=5 |

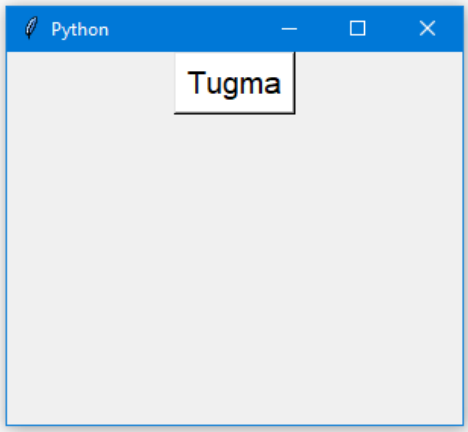
| | | |
|---|---|---|
| wraplength | Agar bu parametrga musbat qiymat berilsa, matn satrlari vertikal holda yoziladi, | =0, =1 |

Tugmani aktiv holatga o’tkazish uchun buyruq parametrini konstruktorga o'rnatishimiz kerak. Quyidagi dasturni ko’rib chiqaylik:

Bu yerda "Tugma" funksiyasi bosish moslamasi sifatida o'rnatiladi. Ushbu funksiya global o'zgaruvchining qiymatini o'zgartiradi va uni oyna sarlavhasida aks ettiradi. Har safar tugma bosilganda "Tugma" funksiyasi ishga tushadi va bosishlar soni bittaga ko'payadi. Dastur kodini Pycharm muhitida yozib quyidagicha natija olamiz:

main.py

```
from tkinter import *
Aktiv = 0
def Tugma():
    global Aktiv
    Aktiv += 1
    root.title("Bosishlar soni={}".format(Aktiv))
root = Tk()
root.title("Python")
root.geometry("300x250")
btn = Button(text="Tugma",bg="white", fg="black",
    activebackground="black",activeforeground="white",
    font="Arial 16", command=Tugma)
btn.pack()
root.mainloop()
```




Biz button elementini oynaga joylashtirish va unga turli vazifalar yuklash usullari haqida yaqindan tanishib o’tdik. Bu element grafik interfeysda eng ko’p qo’llaniladigan elementlardan biri hisoblanadi. Undan kerakli o’rinlarda samarali foydalana olish dasturchining dasturlash qobiliyatiga bog’liq.

Entry vijeti foydalanuvchidan bitta qatorli matn satrlarini qabul qilish uchun ishlatiladi. Agar tahrirlash mumkin bo'lgan bir nechta matn satrlarini namoyish qilmoqchi bo'lsak, u holda Text vijetidan foydalanamiz. Agar foydalanuvchi tomonidan o'zgartirilishi mumkin bo'lmagan bir yoki bir nechta matn satrini

ko'rsatish kerak bo’lsa, u holda Label vijetidan foydalanishimiz maqsadga muvofiq. Entry vijeti sintaksisi quyidagicha:

*Entry (master, xususiyat=qiymat, ...)*

Entry vijeti quyidagi parametrlardan iborat.

| **Parametr** | **Tavsif** | **Qiymat** |
|---|---|---|
| bg | Oddiy fon rangi | =”blue”, =”#456” |
| bd | Ko'rsatkich atrofidagi chegara kengligi piksellarda | =2, ="15” |
| command | Foydalanuvchi har safar ushbu tugmachaning holatini o'zgartirganda chaqiriladigan protsedura | =myfunction |
| cursor | Agar siz ushbu parametrni kursor nomiga o'rnatgan bo'lsangiz sichqoncha kursori tugma tugagandan so'ng shu naqshga o'zgaradi | =”arrow”(o’q)<br>=”dot”(nuqta) |
| font | Shrift o’lchami | =13, ="15",<br>="Arial 20" |
| exportselection | Agar siz Entry vijeti ichidagi matnni tanlasangiz, u avtomatik ravishda buferga eksport qilinadi | =0 |

| | | |
|---|---|---|
| fg | Matn rangi | ="#555",="red", ="yellow" |
| highlightcolor | Tugma fokusga ega bo'lganda fokusning rangi ta'kidlanadi | ="#555",="red", ="yellow" |
| justify | Agar matnda bir nechta satr mavjud bo'lsa, ushbu parametr matnning qanday asoslanishini boshqaradi | =CENTER, =LEFT =RIGHT |
| relief | Chegara turini belgilaydi | = SUNKEN, =RAISED, =GROOVE, =RIDGE |
| selectbackground | Tanlangan matnni aks ettirish uchun fon rangi | ="#555",="red", ="yellow" |
| selectborderwidth | Tanlangan matn atrofida foydalaniladigan chegara kengligi | =12, =20 |
| selectforeground | Tanlangan matnning oldingi (matn) rangi | ="#555",="red", ="yellow" |
| show | Odatda foydalanuvchi kiritgan belgilar yozuvda paydo bo'ladi. Siz bu belgilarni password bilan yashirishingiz mumkin | ="*", ="#" |

| | | |
|---|---|---|
| state | Ushbu parametr vijetni aktiv holatda ishlashni boshlashi va unga bo’lgan murojaatni cheklash uchun ishlatiladi | = NORMAL,<br>= DISABLED,<br>= ACTIVE |
| textvariable | Entry vijetidan joriy matnni olish uchun ushbu parametrni StringVar sinfining misoliga o'rnatishingiz kerak | =variable)o’zgaruvchi nomi) |
| width | Tekshirish tugmachasining standart kengligi ko'rsatilgan rasm yoki matnning o'lchamiga qarab belgilanadi | =15, =20 |
| xscrollcommand | Agar foydalanuvchilar tez-tez vijetning ekrandagi hajmidan ko'proq matn kiritadilar deb o'ylasangiz, kirish vijetingizni aylantirish paneliga bog'lashingiz mumkin | =myfunction |

**Metodlari:**

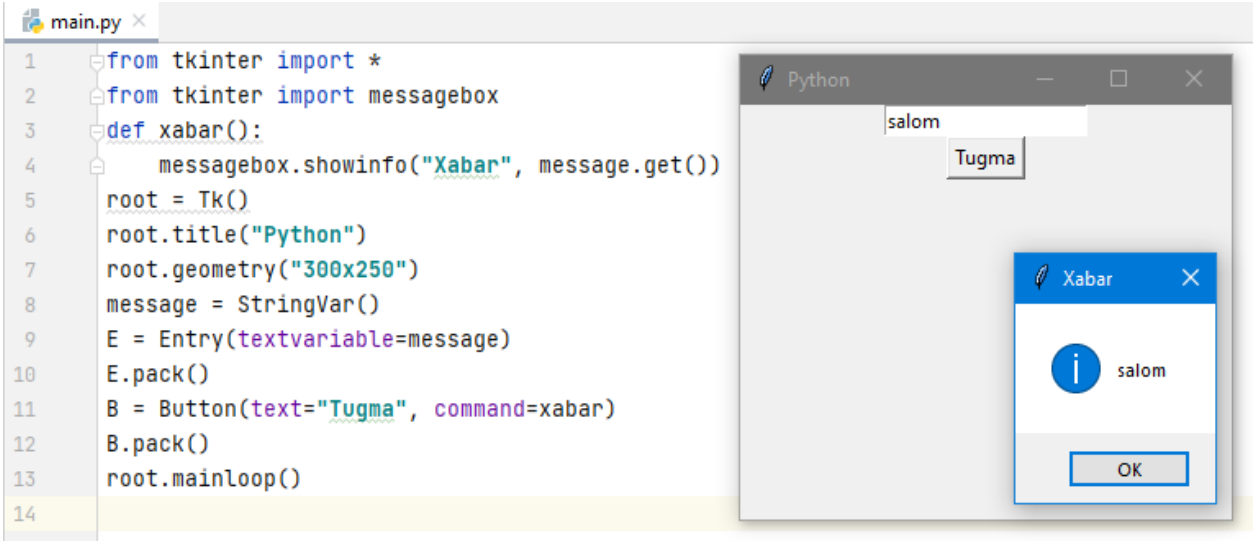
| **Metodlar** | **Tavsif** |
|---|---|
| delete (first, last=None) | Vijetdagi belgilar birinchi indeksdagi belgidan boshlanib, oxirgi o’rindagi belgini qo'shmasdan o'chiriladi. Agar ikkinchi argument tashlansa, faqat birinchi o’rindagi bitta belgi o'chiriladi |

| | |
|---|---|
| get() (“olish”) | Yozuvning joriy matnini satr sifatida qaytaradi. |
| icunsor (index) | Kursorni berilgan indeksdagi belgi oldidan o'rnatadi |
| index | Belgilangan indeksdagi belgi chap tomondagi ko'rinadigan belgi bo'lishi uchun yozuv tarkibini o'zgartiradi. Agar matn to'liq yozuvga to'g'ri keladigan bo'lsa, ta'sir qilmaydi |
| insert (index, s) | Belgidan oldin s qatorini berilgan indeksga kiritadi. |
| select_clear() | Belgilangan sohani tozalaydi. Agar belgilangan soha bo'lmasa, ta'sir qilmaydi |
| select_form (index) | ANCHOR indeks o'rnini indeks bo'yicha tanlangan belgiga o'rnatadi va shu belgini tanlaydi |
| select_present () | Agar belgilangan soha bo'lsa, true qiymatini qaytaradi, aks holda false qiymatini qaytaradi. |
| select_to (index) | Barcha matnni ANCHOR o’rnidan shu indeksdagi belgini qo'shmasdan tanlaydi |
| xview (index) | Ushbu metod Entry vijetini gorizontal aylantirish paneliga bog'lashda ishlatiladi |
| xview_scroll (number, what) | Entry ni gorizontal ravishda aylantirish uchun foydalaniladi. Belgilar kengligi bo'yicha o'tish uchun UNITS yoki PAGES, Entry vijetining kattaligi bo'yicha qismlarga o'tish uchun nima argument bo'lishi kerak. Chapdan o'ngga siljitish uchun |

| | |
|---|---|
| | raqam ijobiy(musbat), o'ngdan chapga o'tish uchun<br>salbiy(manfiy) |

Quyidagi misolda Entry elementini yaratamiz va tugmachani bosib kiritilgan jumlani alohida oynada ko’rsatamiz

Natija:

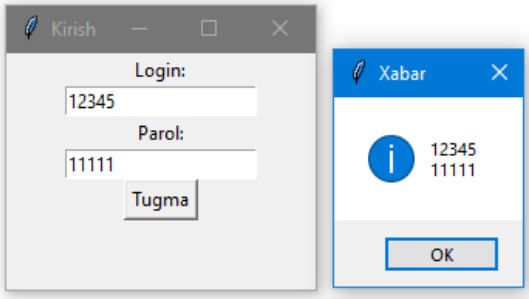



Xabarni ko'rsatish uchun bu yerda **showinfo()** funksiyasini o'z ichiga olgan qo'shimcha modulli xabar qutisi ishlatiladi, bu matn maydoniga kiritilgan matnni aks ettiradi. Kiritilgan matnni olish uchun oldingi mavzulardan birida tasvirlanganidek, **StringVar** komponentasidan foydalanildi.

Boshqa misol ko’raylik:

main.py

```
from tkinter import *
from tkinter import messagebox
def xabar():
    messagebox.showinfo("Xabar", login.get() + "\n" + parol.get())
root=Tk()
root.title("Kirish")
root.geometry("200x150")
login = StringVar()
parol = StringVar()
Label(text="Login:").pack()
Entry(textvariable=login).pack()
Label(text="Parol:").pack()
Entry(textvariable=parol).pack()
Button(text="Tugma", command=xabar).pack()
root.mainloop()
```



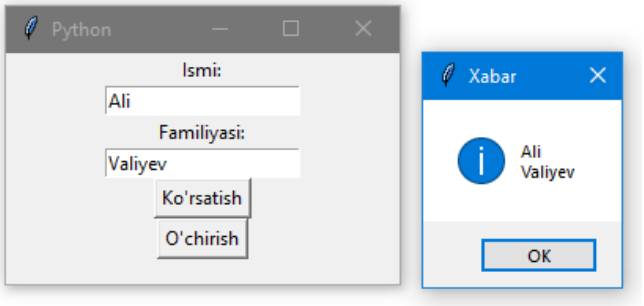

Bu misolda biz hali ko’rib o’tmagan Label vijetidan foydalanildi. Dastur kodi ishga tushirilganda Kirish nomli oynada login hamda parolni kiritish so’raldi va natija Xabar oynasida ko’rsatildi. Shu va shunga o’xshash ko’plab misollarda Entry vijetining foydalanish o’rni va ahamiyatini ko’rishingiz mumkin.

Entry vijeti bir qator metodlarga ega. Ulardan asosiylari:

| Metod | Tavsif |
|---|---|
| insert(index, str) | ma'lum bir indeksdagi qatorni matn maydoniga qo'shib qo'yadi |
| get() | matn maydoniga kiritilgan matnni qaytaradi |
| delete(first, last=None) | avval indeksdagi belgini olib tashlaydi. Agar oxirgi parametr ko'rsatilgan bo'lsa, o'chirish oxirgi indeksdan oldin amalga oshiriladi. Oxirigacha olib tashlash uchun END ni ikkinchi parametr sifatida ishlatish mumkin |

Quyidagi dasturda metodlardan foydalanamiz

```
main.py
def Xabar():
    messagebox.showinfo("Xabar", ism.get() + "\n" + familiya.get())
root = Tk()
root.title("Python")
root.geometry("250x150")
Label(text="Ismi:").pack()
ism = Entry()
ism.pack()
ism.insert(0, "Ali")
Label(text="Familiyasi:").pack()
familiya = Entry()
familiya.pack()
familiya.insert(0, "Valiyev")
Button(text="Ko'rsatish", command=Xabar).pack()
Button(text="O'chirish", command=Tozalash).pack()
root.mainloop()
```




Dastur ishga tushirilganda, standart matn bir vaqtning o'zida ikkala matn maydoniga qo'shiladi:

*ism.insert(0, "Ali")*

*familiya.insert(0, "Valiyev")*

Tozalash tugmasi o'chirish funksiyasini chaqirish orqali ikkala maydonni ham tozalaydi:

*def Tozalash():*

*ism.delete(0, END)*

*familiya.delete(0, END)*

**Get**() metodi yordamida ikkinchi tugma kiritilgan matnni oladi:

*def Xabar():*

*messagebox.showinfo("Xabar", ism.get() + "\n " + familiya.get())*

Bundan tashqari, misoldan ko'rinib turibdiki, biz StringVar turidagi o'zgaruvchilar orqali yozuvdagi matnga murojaat qilishimiz shart emas, biz buni to'g'ridan-to'g'ri get metodi orqali amalga oshirishimiz mumkin.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

## 37-mavzu: Label, Checkbutton, Radiobutton va *Combobox* komponenti komponentlari

**Mavzu rejasi:**

1.Label komponenti.

2.Checkutton komponenti.

3.Radiobutton komponenti.

4.Scale komponenti

Label vijeti yaratilayotgan oynada matn yoki rasmlarni joylashtirish mumkin bo’lgan sohani belgilaydi. Ushbu vijet ko'rsatadigan matnni xohlagan vaqtda yangilash mumkin.

*Label(master, xususiyat=qiymat,...)*

Label vijetining ham bir qator parametrlari bor:

| Parametr | Tavsif | Qiymat |
|---|---|---|
| anchor | Agar vijetda matn ehtiyojidan ko'proq joy bo'lsa, matn qayerda joylashishini boshqaradi | =CENTER, |

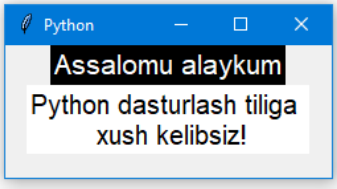
| | | |
|---|---|---|
| | | =RIGHT |
| bg | Oddiy fon rangi, yorliq va indikator orqasida ko'rsatiladi | =”red”,<br>=”#555” |
| bd | Ko'rsatkich atrofidagi chegara kattaligi | =2, =5 |
| cursor | Agar ushbu parametr kursor nomiga o'rnatilgan bo'lsa, sichqoncha kursori tugma tugagandan so'ng shu naqshga o'zgaradi | =”arrow”(o’q)<br>=”dot”(nuqta) |
| font | Agar ushbu yorliqda matn namoyish qilinsa, matn yoki matn o'zgaruvchan variant bilan birga, shrift opsiyasi ushbu matn qaysi shriftda ko'rsatilishini belgilaydi | =”Arial 14”,<br>=”Calibri 15” |
| fg | Agar ushbu yorliqda matn yoki bitmap ko'rsatilgan bo’lsa, ushbu parametr matn rangini belgilaydi | =”red”, =”#333” |
| height | Yangi ramkaning vertikal o'lchamlari | =15, =20 |
| image | Yorliq vijetida statik tasvirni ko'rsatish uchun ushbu parametrni rasm ob'ektiga o'rnating. | |
| justify | Matnning bir nechta satrlari bir-biriga nisbatan qanday tekislanishini belgilaydi | =CENTER,<br>=LEFT,<br>=RIGHT |
| padx | Vijet ichidagi matnning chap va o'ng qismiga qo'shimcha joy qo'shildi | =1, =5 |

| | | |
|---|---|---|
| pady | Vijet ichidagi matnning yuqorisida va ostiga qo'shimcha joy qo'shildi | =3, =6 |
| relief | Label atrofidagi dekorativ chegara ko'rinishini belgilaydi | =FLAT |
| text | Label vijetida bir yoki bir nechta satrlarni ko'rsatish uchun ushbu parametrni matn o'z ichiga olgan qatorga o'rnatadi | |
| textvariable | Label vijetida ko'rsatilgan matnni StringVar sinfidagi boshqaruv o'zgaruvchisiga saqlash uchun ushbu parametrni ushbu o'zgaruvchiga o'rnatadi | |
| underline | Ushbu parametrni n ga o'rnatib, matnning n harfi ostida 0 dan boshlab, pastki chizig'ini (_) ko'rsatishi mumkin. Sukut bo'yicha chiziq underline= -1, ya'ni pastki chiziq chizilmaydi. | =2, =5 |
| width | Belgilardagi Label ning kengligi (piksel emas!). Agar ushbu parametr o'rnatilmagan bo'lsa, Label uning tarkibiga mos keladigan hajmga ega bo'ladi. | =15, =20 |

Ilova oynasida eng sodda matnni ko'rsatamiz:

main.py

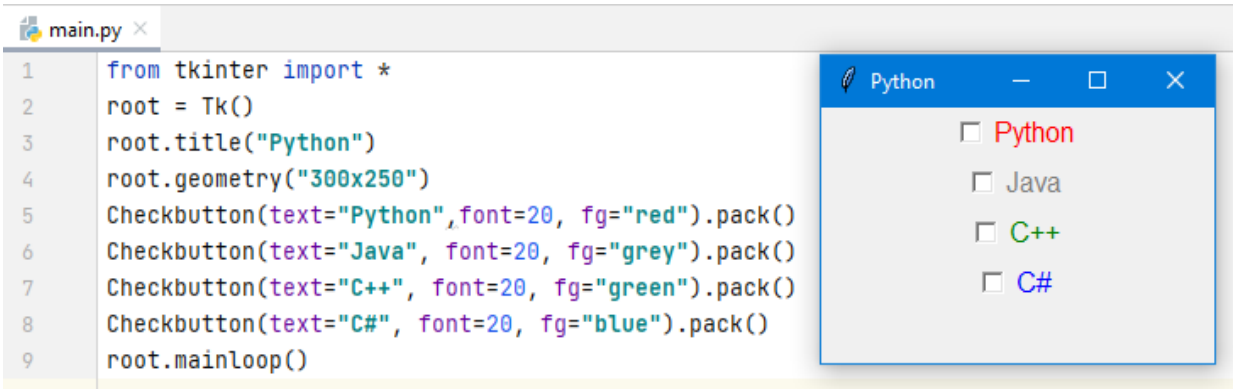
```
from tkinter import *
root = Tk()
root.title("Python")
root.geometry("300x100")
Label(text="Assalomu alaykum",fg="white", font="Arial 16", bg="black").pack()
matn = "Python dasturlash tiliga \n xush kelibsiz!"
Label(text=matn, font="Arial 15", fg="black", bg="white").pack()
root.mainloop()
```



Checkbutton vijeti foydalanuvchiga o'tish tugmachalari sifatida bir qator variantlarni ko'rsatish uchun ishlatiladi. Keyin foydalanuvchi har bir parametrga mos keladigan tugmani bosish orqali bir yoki bir nechta variantni tanlashi mumkin. Matn o'rniga rasmlarni ham ko'rsatishimiz mumkin. Checkbutton elementi quyida ko’rinishda e’lon qilinadi:

*Checkbutton(master, xususiyat=qiymat,…)*

Quyidagi misol orqali checkbutton nima ekanligi va dastur kodida qay tarzda ifodalanishini ko’rib chiqaylik:




Ko’rib turganingizdek dastur kodi qisqa va tushunarli: "Python" deb nomlangan bitta oyna va 4 ta checkbutton elementidan tashkil topgan. Eslatib o’tamiz Checkbutton vijeti ham boshqa vijetlar kabi ko’plab parametrlarga ega. Quyidagi jadvalda bir qator parametrlar va ularning qabul qilishi mumkin bo’lgan qiymatlarini keltiramiz:

| **Parametr** | **Tavsif** | **Qiymat** |
|---|---|---|
| activebackground | Tugma bosilganda orqa fon rangi | ="red", =”#000” |
| activeforeground | Tugma bosilganda matn rangi | =”red”,=”#555”, |
| bg | Oddiy fon rangi | =2, =15 |

| | | |
|---|---|---|
| bd | Ko'rsatkich atrofidagi chegara kattaligi | =”blue”,=”#456” |
| command | Foydalanuvchi ushbu tugmachaning holatini o'zgartirganda chaqiriladigan funksiya yoki protsedura | =myFunction |
| cursor | Agar siz ushbu parametrni kursor nomiga o'rnatgan bo'lsangiz (o'q, nuqta va boshqalar), sichqoncha kursori tugma tugagandan so'ng shu naqshga o'zgaradi | =”arrow”(o’q)<br>=”dot”(nuqta) |
| disabledforeground | O'chirilgan tugmachani ko'rsatish uchun ishlatiladigan oldingi rang | =”red”, =”#256” |
| font | Matn shrift | =”blue”, =”#456” |
| fg | Matn rangi | =”Arial 15”,=”14”<br>=”Calibri 10” |
| height | Tekshirish tugmachasidagi satrlar soni | =25,=5 |
| highlightor | Tugma fokusga ega bo'lganda fokusning rangi | =”blue”, =”#456” |
| image | Tugmachada grafik tasvirni ko'rsatish uchun ishlatiladi | |
| justify | Agar matnda bir nechta satrdan iborat bo’lsa, uning joylashish o’rnini belgilaydi | CENTER, =LEFT =RIGHT |

| | | |
|---|---|---|
| offvalue | Tekshiruv tugmasi bilan bog'liq boshqaruv o'zgaruvchisi o'chirilganda 0 ga o'rnatiladi. | =0 |
| onvalue | Tasdiqlash tugmachasi bilan bog'liq boshqaruv o'zgaruvchisi yoqilganda1 ga o'rnatiladi. | =1 |
| padx | Tekshirish tugmasi va matnning chap va o'ng tomonidan ajratilishi kerak bo’lgan joyni belgilaydi | =20, =15 |
| pady | Tekshirish tugmasi va matnning yuqorisida va past tomonidan ajratilishi kerak bo’lgan joyni belgilaydi | =5, =10 |
| relief | Chegara turini belgilaydi | = SUNKEN,<br>=RAISED,<br>=GROOVE,<br>=RIDGE |
| selectcolor | Belgilangan tugmachaning rangi | =”red” |
| selectimage | Agar ushbu parametr rasmga o'rnatilgan bo'lsa, rasm tasdiqlash tugmachasida paydo bo'ladi | |
| state | Tasdiqlash tugmasini aktiv holatda ishlashni boshlashi va unga bo’lgan murojaatni cheklash uchun ishlatiladi | = DISABLED,<br>= ACTIVE,<br>=NORMAL |

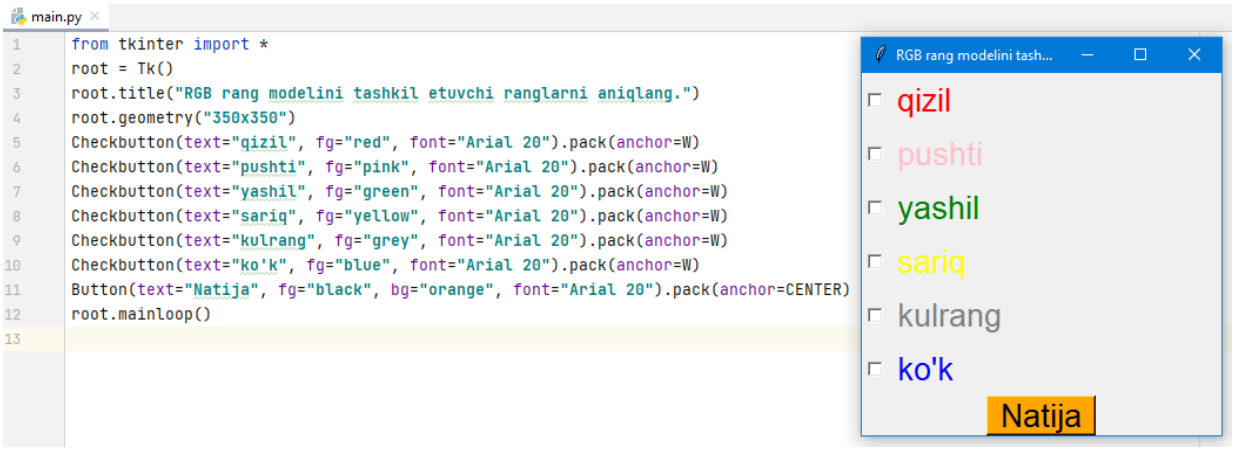
| | | |
|---|---|---|
| text | Belgilash tugmasi yonida ko'rsatiadigan yozuv. Matn satrlarini ko'rsatish uchun yangi qatorlardan ("\ n") foydalaniladi | =”salom” |
| underline | Agar manfiy qiymat olmasa, tegishli matn belgisi ostiga chiziladi | =-1, =5, =10 |
| variable | Tekshirish tugmachasining joriy holatini kuzatadigan boshqaruv o'zgaruvchisi | =0,<br>=1 |
| width | Tekshirish tugmachasining standart kengligi, ko'rsatilgan rasm yoki matnning o'lchamiga qarab belgilanadi | =10, =15 |
| wraplength | Parametr musbat qiymat qabul qilganda berilgan matn vertikal ko’rinishda yoziladi | =2, =5 |

**Ushbu vijet uchun quyidagi keng tarqalgan metodlar qo'llaniladi:**

| Metod | Tavsif |
|---|---|
| deselect() (“bekor qilish”) | Tekshirish tugmachasini tozalaydi (o'chiradi) |
| flash() (“chaqmoq”) | Faol va normal ranglar o'rtasida bir necha marta yonib-o'chib turadi, lekin uni qanday boshlagan bo'lsa, shunday qoldiradi |
| invoke() (“chaqirish”) | Xuddi shu amallarni bajarish uchun ushbu usulni chaqirishingiz mumkin agar foydalanuvchi o'z holatini o'zgartirish uchun tugmani bosgan bo'lsa paydo bo'ladi |

| | |
|---|---|
| select() (“tanlash”) | Tekshirish tugmachasini o'rnatadi (yoqadi). |
| toggle()<br><br>(“almashtirish”) | O'rnatilgan bo'lsa, tugmachani tozalaydi, agar o'chirilgan bo'lsa, uni o'rnatadi. |

Checkbutton vijetini qo’llashga doir yana boshqa bir misolni tahlil qilamiz:




Checkbutton vijetidan hozirda ko’p dasturlarda foydalanilmoqda. Misol sifatida google tarmoqdagi “I’m not robot” sahifasini keltirishimiz mumkin. Biz ilovamizda biror jumlani tasdiqlashda yoki kerakli xususiyat(obyekt)larni ketma-ket belgilab chiqishda foydalanishimiz mumkin.

Radiobutton - bu ko'p tanlovlardan birini amalga oshirish uchun ishlatiladigan standart Tkinter **vidjeti . Radio tugmalari** matn yoki tasvirni o'z ichiga olishi mumkin va siz har bir tugma bilan Python funktsiyasi yoki usulini bog'lashingiz mumkin. Tugma bosilganda, Tkinter avtomatik ravishda ushbu funktsiya yoki usulni chaqiradi. **Sintaksis:**

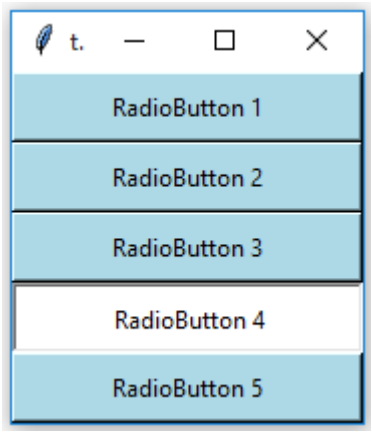
```
button = Radiobutton(master, text="Tugmadagi nom", variable = “birgalikda o`zgaruvchi"
                     value = “har bir tugma qiymatlari", options = values, …)
```

umumiy o'zgaruvchi = Barcha Radio tugmalari o'rtasida taqsimlangan Tkinter o'zgaruvchisi
qiymati = har biri radiotugma boshqa qiymatga ega bo'lishi kerak, aks holda 1 dan ortiq radiotugma tanlanadi.

Radio tugmalari, lekin tugmalar shaklida emas, tugmalar **qutisi** shaklida . Tugma oynasini ko'rsatish uchun *indikator/indikator* opsiyasi 0 ga o'rnatilishi kerak.

```
from tkinter import *

master = Tk()
master.geometry("175×175")

v = StringVar(master, "1")

values = {"RadioButton 1" : "1",
        "RadioButton 2" : "2",
        "RadioButton 3" : "3",
        "RadioButton 4" : "4",
        "RadioButton 5" : "5"}


for (text, value) in values.items():
    Radiobutton(master, text = text, variable = v,
                value = value, indicator = 0,
                background = "light blue").pack(fill = X, ipady = 5)

mainloop()
```


Ushbu tugma qutilarining foni ochiq ko'k rangda. Oq fonga ega bo'lgan tugma qutilari, shuningdek, cho'kib ketganlar tanlangan.




Scale vijeti ma'lum miqyosdagi qiymatlarni tanlashga imkon beradigan grafik slayder ob'ektini taqdim etadi. Ushbu vijetni yaratish uchun oddiy sintaksis:

*w = Scale (master, xossa=qiymat, ....)*

| **Parametr** | **Tavsif** |
|---|---|
| activebackground | Sichqoncha tarozidan oshganda fon rangi. |
| bg | Vijetning truba tashqarisidagi qismlarining fon rangi. |

| | |
|---|---|
| bd | Chuqurcha va slayder atrofidagi uchburchakning kengligi. Standart - 2 piksel. |
| command | Har safar slayder harakatlantirilganda chaqiriladigan protsedura. Ushbu protsedura bitta argumentdan, yangi o'lchov qiymatidan o'tadi. Agar slayder tezlik bilan ko'chirilsa, siz har qanday pozitsiya uchun qayta qo'ng'iroqni qabul qilmasligingiz mumkin, ammo bu aniqlanganda siz qayta qo'ng'iroq qilishingiz mumkin. |
| cursor | Agar siz ushbu parametrni kursor nomiga o'rnatgan bo'lsangiz (o'q, nuqta va boshqalar), sichqoncha kursori shkaladan oshib ketgach, o'sha naqshga o'zgaradi. |
| digits | Dasturingiz o'lchov vijetida ko'rsatilgan joriy qiymatni o'qish usuli o'zgaruvchan o'zgaruvchidir. Shkala uchun boshqaruv o'zgaruvchisi IntVar, DoubleVar (float) yoki StringVar bo'lishi mumkin. Agar u satr o'zgaruvchisi bo'lsa, raqamlar opsiyasi raqamli shkala qiymati satrga aylantirilganda qancha raqam ishlatilishini boshqaradi. |
| font | Yorliq va izohlar uchun ishlatiladigan shrift. |
| fg | Yorliq va izohlar uchun ishlatiladigan matnning rangi. |

| | |
|---|---|
| from_ | Shkala diapazonining bir uchini belgilaydigan float yoki integer qiymati. |
| highlightbackground | Shkalada fokus bo'lmasa, fokusning rangi ta'kidlanadi. |
| highlightcolor | Shkala fokusga ega bo'lganda fokusning rangi ta'kidlanadi. |
| label | Ushbu parametrni label matniga o'rnatib, o'lchov vijetida label ni ko'rsatishingiz mumkin. Label gorizontal bo'lsa chap yuqori burchakda, vertikal bo'lsa o'ng yuqori burchakda ko'rinadi. Odatiy label emas |
| length | O'lchov vijetining uzunligi. Agar o'lchov gorizontal bo'lsa x o'lchovi yoki vertikal bo'lsa y o'lchovdir. Odatiy qiymati 100 piksel. |
| orient | Agar o'lchov x o'lchovi bo'ylab harakatlanishini istasangiz, orient = HORIZONTAL ni o'rnating yoki orient = VERTICAL ni y o'qiga parallel ravishda bajaring. Odatiy holatda gorizontal bo’ladi |
| relief | Yorliq atrofidagi dekorativ chegara ko'rinishini belgilaydi. Odatiy qiymati FLAT |

| | |
|---|---|
| repeatdelay | Ushbu parametr slayder ushbu yo'nalishda bir necha bor harakatlana boshlaguncha, 1 tugmachasini truba ichida qancha vaqt ushlab turish kerakligini nazorat qiladi. Odatiy - repeatdelay= 300, birliklar esa millisekundlarda bo’ladi. |
| resolution | Odatda, foydalanuvchi o'lchovni faqat butun birliklarda o'zgartira oladi. Shkala qiymatining eng kichik o'sishini o'zgartirish uchun ushbu parametrni boshqa qiymatga o'rnating. Masalan, agar from_= -1,0 dan from_= 1,0 gacha, va siz resolution = 0,5 ni o'rnatgan |
| | bo'lsangiz, shkala 5 ta mumkin bo'lgan qiymatga ega bo'ladi: -1.0, 0.5, 0.0, +0.5 va +1.0. |
| showvalue | Odatda, masshtabning joriy qiymati slayder tomonidan matn shaklida ko'rsatiladi (gorizontal tarozi uchun uning ustida, vertikal tarozi uchun chapda). Ushbu yorliqni bostirish uchun ushbu parametrni 0 ga sozlang |
| sliderlength | Odatda slayder shkalaning uzunligi bo'yicha 30 pikselga teng. Slayder uzunligi parametrini kerakli uzunlikka o'rnatish orqali siz ushbu uzunlikni o'zgartirishingiz mumkin |
| state | Odatda, miqyosdagi vijetlar sichqoncha hodisalariga javob beradi Ya’ni ular diqqat markazida bo'lganda, |

| | |
|---|---|
| | shuningdek klaviatura hodisalarida. Vijetga javob bermaslik uchun state = DISABLED bo’ladi |
| takefocus | Odatda, diqqat markazidagi vijetlar bo'ylab aylanadi. Ushbu harakatni xohlamasangiz, ushbu parametrni 0 ga sozlang |
| tickinterval | Vaqti-vaqti bilan o'lchov qiymatlarini ko'rsatish uchun ushbu parametrni raqamga qo'ying va shu qiymatning ko'paytmalarida belgilar paydo bo'ladi. Masalan, from_ = 0.0, 1.0 gacha va tickinterval = 0.25 bo'lsa, yorliqlar shkala bo'yicha 0.0, 0.25, 0.50, 0.75 va 1.00 qiymatlarida ko'rsatiladi. Ushbu yorliqlar gorizontal bo'lsa, shkaladan pastda, vertikal bo'lsa chap tomonda ko'rinadi. Odatiy qiymati 0, bu shomil ko'rsatilishini bostiradi |
| to | O'lchovning bitta uchini belgilaydigan float yoki integer qiymati ; boshqa uchlari yuqorida muhokama qilingan from_ varianti bilan belgilanadi. To qiymati from_ qiymatidan katta yoki kichik bo'lishi mumkin. Vertikal tarozilar uchun to qiymati shkalaning pastki qismini belgilaydi; gorizontal tarozilar uchun o'ng uchi. |
| troughcolor | Olukning rangi. |
| variable | Agar mavjud bo'lsa, ushbu o'lchov uchun boshqaruv o'zgaruvchisi. Boshqaruv o'zgaruvchilari IntVar, DoubleVar (float) yoki StringVar sinfidan bo'lishi |

| | |
|---|---|
| | mumkin. Ikkinchi holda, raqamli qiymat mag'lubiyatga aylantiriladi. |
| width | Vijetning pastki qismi kengligi. Bu vertikal tarozilar uchun x o'lchov va agar shkalasi orient = HORIZONTAL bo'lsa, y o'lchovidir. Standart 15 piksel. |

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

## 38-mavzu: Notebook, Progressbar, Sizegrip, va Treeview komponentlari

**Mavzu rejasi:**

1.Notebook komponenti.

2.Progressbar komponenti.

3.Sizegrip komponenti

4.Treeview komponenti

Notebook komponenti yorliqlarni bosish orqali tarkib sahifalarini tanlash imkonini beradi :


Ushbu yorliqlardan birini bosganingizda, Notebook komponenti tanlangan yorliq bilan bog'langan bolalar panelini ko'rsatadi. Odatda, bolalar paneli Frame komponenti hisoblanadi.

Notebook komponenti yaratish uchun siz ttk.Notebook sinfdan quyidagi tarzda foydalanasiz:

```
notebook = ttk.Notebook(container,**options)
```

Bu sintaksisda konteyner daftarning ota-onasi hisoblanadi. Odatda, bu ildiz oynasi.

Notebookda foydali variantlar mavjud. Masalan, siz komponentga ajratilgan piksellardagi balandlik va kenglikni belgilash uchun balandlik va  kenglik parametrlaridan foydalanasiz.

Bundan tashqari, **padding** opsiyadan foydalanib, komponentning tashqi tomoniga biroz bo‘sh joy qo‘shishingiz mumkin .

## Notebook usullari

Sinf ttk.Notebook sizga yorliqlarni samarali boshqarish imkonini beruvchi ko'plab qulay usullarni taqdim etadi.

Progressbar komponenti foydalanuvchiga uzoq davom etgan vazifaning borishi haqida fikr bildirish imkonini beradi. Progressbar komponentini yaratish uchun siz ttk.Progressbar sinfdan foydalanasiz:

```
ttk.Progressbar(container, **options)
```

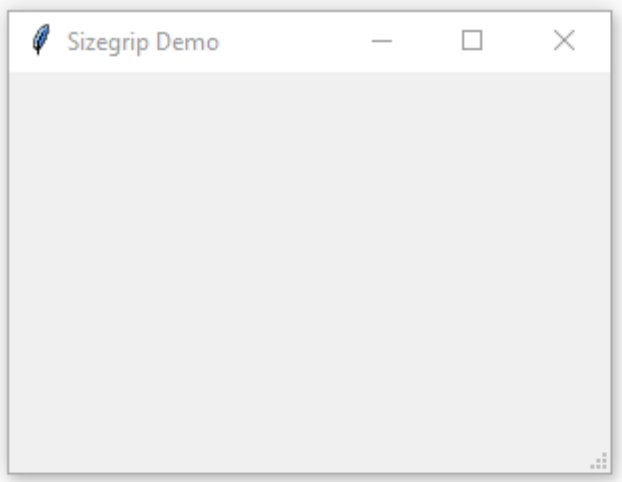
Quyida Progressbar komponentini yaratish uchun odatiy parametrlar ko'rsatilgan:

```
ttk.Progressbar(container, orient, length, mode)
```

Ushbu sintaksisda:

- Bu container progressbarning asosiy komponentidir.
- Gorizontal length progress satrining kengligi yoki vertikal progressbar balandligini ifodalaydi.

Vidjet odatda oynaning Sizegrip pastki o'ng burchagida joylashgan. U ilova oynasini kiritish hajmini o‘zgartirish imkonini beradi:




Vidjet yaratish uchun Sizegripsiz quyidagi sintaksisdan foydalanasiz:

```
ttk.Sizegrip(parent, **option)
```

Vidjetning to'g'ri ishlashiga ishonch hosil qilish uchun Sizegripsiz ildiz oynasining o'lchamini o'zgartirishingiz kerak.

Agar siz grid geometriya menejeridan foydalansangiz, ustun va satr o'lchamlarini sozlashingiz kerak.

Quyidagi dastur ildiz oynasining pastki o'ng qismida Sizegrip-ni ko'rsatadi:

```
import tkinter as tk
from tkinter import ttk

root = tk.Tk()
root.title('Sizegrip Demo')
root.geometry('300x200')
root.resizable(True, True)

# grid layout
root.columnconfigure(0, weight=1)
root.rowconfigure(0, weight=1)

# create the sizegrip
sg = ttk.Sizegrip(root)
sg.grid(row=1, sticky=tk.SE)


root.mainloop()
```

Treeview komponenti ma'lumotlarni jadval va ierarxik tuzilmalarda ko'rsatish imkonini beradi. Treeview komponentini yaratish uchun siz ttk.Treeview sinfdan foydalanasiz:

```
tree = ttk.Treeview(container, **options)
```

Treeview komponentida elementlar ro'yxati mavjud. Har bir elementda bir yoki bir nechta ustunlar mavjud.

Birinchi ustunda matn va uni kengaytirish mumkinmi yoki yo'qligini ko'rsatadigan belgi bo'lishi mumkin. Qolgan ustunlar har bir satrning qiymatlarini o'z ichiga oladi.

Treeview ko'rinishining birinchi qatori har bir ustunni nom bilan belgilaydigan sarlavhalardan iborat.

## Nazorat savollari

1.Treeview komponentini tavsifini tushuntiring.
2. Sizegrip komponentini tavsifini tushuntiring.
3.Komponentlarni yana qanaqalarini bilasiz?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**39-mavzu: Listbox, Spinbox va PanedWindow komponentlari**

**Mavzu rejasi:**

1.Uslubsiz komponentlar.

2.Listbox komponenti.

3.Spinbox komponenti.

4.PanedWindow komponenti

Listbox vidjeti bir qatorli matn elementlari ro‘yxatini ko‘rsatadi. Ro'yxat qutisi sizga elementlarni ko'rib chiqish va bir vaqtning o'zida bir yoki bir nechta elementlarni tanlash imkonini beradi.

Ro'yxat qutisini yaratish uchun siz quyidagi tk.Listboxsinfdan foydalanasiz:

```
listbox = tk.Listbox(container, listvariable, height)
```

Ushbu sintaksisda:

- Ro'yxat container qutisining asosiy komponentidir.
- List variableStringVar ob'ektiga havolalar . Bu haqda keyinroq batafsil tushuntirish.
- Ro'yxat height oynasi aylantirmasdan ko'rsatadigan elementlar soni.

## Ro'yxat elementlarini boshqarish

Ro'yxat qutisiga elementlarni to'ldirish uchun siz avval StringVar elementlar ro'yxati bilan ishga tushirilgan ob'ektni yaratasiz. StringVar va keyin siz ushbu ob'ektni listvariable variantga quyidagicha tayinlaysiz :

```
list_items = StringVar(value=items)
listbox = tk.Listbox(
    container,
    height,
    listvariable=list_items
)
```

Spinbox vidjeti qiymatlar to‘plamidan qiymat tanlash imkonini beradi. Qiymatlar bir qator raqamlar bo'lishi mumkin.

Spinbox-da joriy qiymatni ko'rsatish uchun maydon va bir juft o'q uchlari mavjud.

Yuqoriga qaragan o'qni bosganingizda, Spinbox joriy qiymatni ketma-ketlikda keyingi yuqori qiymatga o'tkazadi. Agar joriy qiymat maksimal qiymatga yetsa, uni minimal qiymatga o'rnatishingiz mumkin.

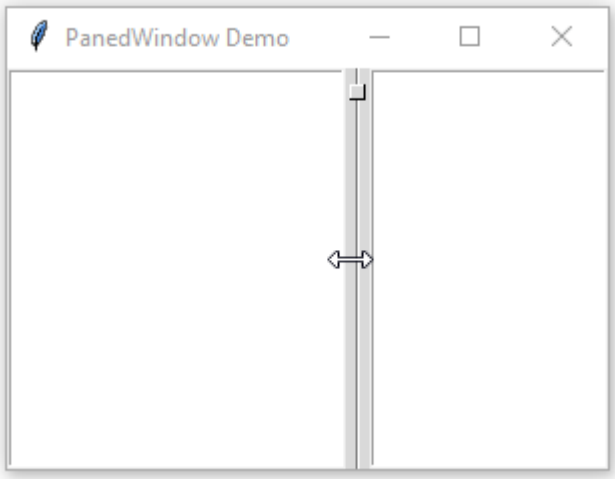
Boshqa tomondan, agar siz pastga yo'naltirilgan o'qni bossangiz, Spinbox joriy qiymatni ketma-ketlikdagi keyingi pastki qiymatga o'tkazadi. Agar joriy qiymat eng past qiymatga yetsa, uni maksimal qiymatga o'rnatishingiz mumkin.

Shuningdek, siz Spinbox vidjetiga xuddi Kirish vidjeti kabi qiymat kiritishingiz mumkin .

Spinbox vidjetini yaratish uchun siz ttk.Spinbox konstruktordan foydalanasiz. Bu yerda odatiy variantlar:

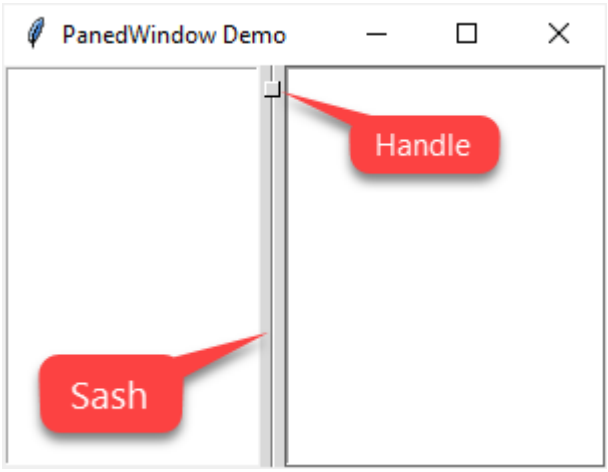
```
ttk.Spinbox(container, from_, to, textvariable, wrap)
```

Odatda, a PanedWindowbola vidjetlarining vertikal yoki gorizontal to'plamini o'z ichiga oladi:




A PanedWindowbola vidjetlarini ajratish uchun bardan foydalanadi. Ushbu novda *kanat* deb ataladi .

*To'shakda* sichqoncha bilan sudrab olib boradigan kichik kvadrat shaklidagi tutqich *bo'lishi* mumkin :




*Panel* - bitta kichik vidjet egallagan maydon.

Vidjet yaratish uchun PanedWindowsiz quyidagi sintaksisdan foydalanasiz:

```
ttk.PanedWindow(container, **options)
```

**Nazorat savollari**

1. Listbox komponentini tavsifini tushuntiring.
2. Spinbox komponentini tavsifini tushuntiring.
3. PanedWindow komponentini tavsifini tushuntiring.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**40-mavzu: Menu  komponenti va uning imkoniyatlari**

**Mavzu rejasi:**

*1.Menu* komponenti.
2.Menu komponentining imkoniyatlari.
3.Asosiy menyu yaratish.
4.Kontekst menyusini yaratish.
5.Menubutton komponenti
**6.**“Tezkor tugmalar” bilan ishlash.

## Oddiy menyu yaratish

Birinchidan, root oyna yarating va uning sarlavhasini quyidagicha o'rnating 'Menu Demo':

```
root = tk.Tk()
root.title('Menu Demo')
```

Ikkinchidan, menyu satrini yarating va uni oyna menuvariantiga tayinlang :root

```
menubar = Menu(root)
root.config(menu=menubar)
```

E'tibor bering, har bir yuqori darajadagi oynada faqat bitta menyu satri bo'lishi mumkin.

Uchinchidan, konteyneri bo'lgan **Fayl** menyusini yarating menubar:

```
file_menu = Menu(menubar)
```

To'rtinchidan, menyuga quyidagi band qo'shing file_menu:

```
file_menu.add_command(
    label='Exit',
    command=root.destroy,
)
```

Ushbu misolda menyu elementining yorlig'i Exit.

Menyu bandini bosganingizda , Python oynani yopish uchun avtomatik ravishda usulni Exitchaqiradi .root.destroy() root

Nihoyat, Filemenyuni menyu paneliga qo'shing:

```
menubar.add_cascade(
    label="File",
    menu=file_menu,
    underline=0
)
```

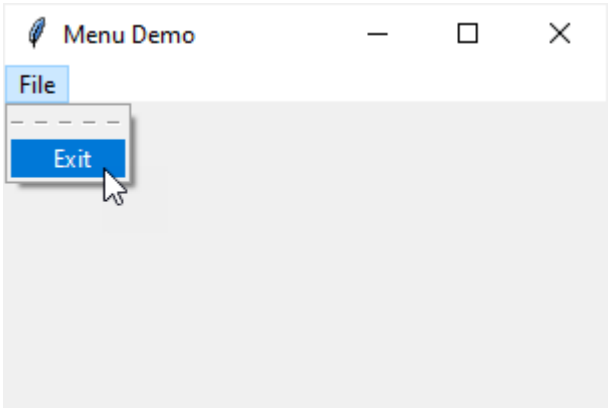
underlineVariant klaviatura yorlig'ini yaratishga imkon beradi . U tagiga chizilishi kerak bo'lgan belgi o'rnini belgilaydi.

E'tibor bering, pozitsiya noldan boshlanadi. Ushbu misolda biz uni birinchi belgi sifatida belgilaymiz F. **Alt+F** va siz uni klaviatura yorlig'i yordamida tanlashingiz mumkin .

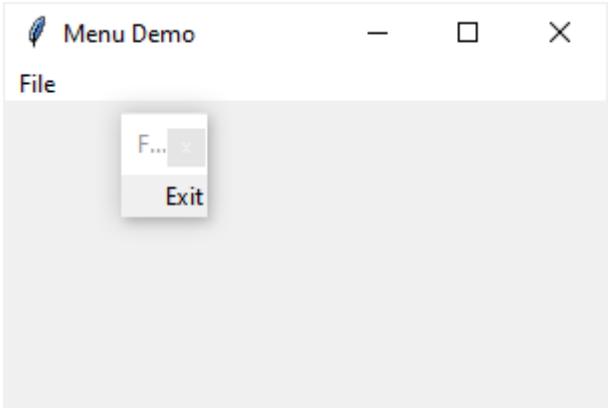
Hammasini bir joyga qo'ying:

```
import tkinter as tk
from tkinter import Menu

# root window
root = tk.Tk()
root.title('Menu Demo')

# create a menubar
menubar = Menu(root)
root.config(menu=menubar)

# create a menu
file_menu = Menu(menubar)

# add a menu item to the menu
file_menu.add_command(
    label='Exit',
    command=root.destroy
)


# add the File menu to the menubar
menubar.add_cascade(
    label="File",
    menu=file_menu
)

root.mainloop()
```

Chiqish:



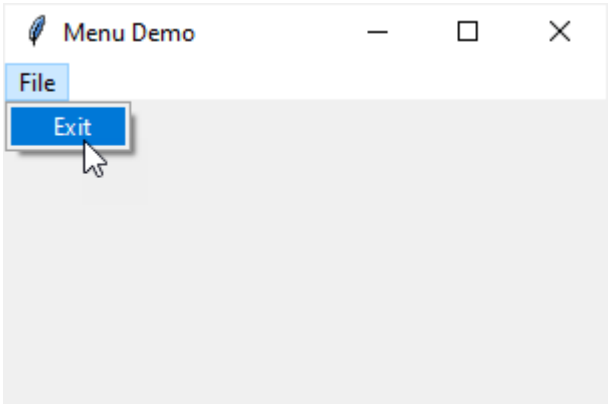

Odatiy bo'lib, Tkinter birinchi menyu elementi oldiga chiziqli chiziq qo'shadi. Kesilgan chiziqni bosganingizda, asosiy oyna undan menyuni quyidagicha ajratadi:




Kesilgan chiziqni olib tashlash uchun tearoffmenyuning xususiyatini quyidagicha o'rnatishingiz mumkin False:

```
file_menu = Menu(menubar, tearoff=False)
```

Chiqish:




**Nazorat savollari**

1. Menu komponentini tavsifini tushuntiring.
2. Menu komponentining imkoniyatlari qaysilar?.
3. Kontekst menyusini yaratish jarayoni haqida tushuncha bering.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**41-mavzu: Standart dialog oynalar bilan ishlash**

**Mavzu rejasi:**

1.Standart dialog oynalarini yaratish va undan foydalanish.

2.Faylni ochish va saqlash uchun dialog oynalarini ko'rsatish.

3.Yuqori darajadagi uskunalar.

Oldindan belgilangan GUI dialog oynalari yordamida bajarilishi mumkin bo'lgan ko'plab umumiy dasturlash vazifalari mavjud. Quyidagi muhokamada ushbu dialog oynalari tasvirlangan va bir necha oddiy misollar keltirilgan. Qo'shimcha ixtiyoriy parametrlar uchun Python hujjatlariga murojaat qilishingiz mumkin.

Xabarlar (messagebox)

**messagebox** foydalanuvchiga ma'lumotlarni ko'rsatishi mumkin. Siz ko'rsatmoqchi bo'lgan xabar turiga qarab, ushbu dialog oynalarida uchta variant mavjud. Funktsiyalarning birinchi parametri oynaning sarlavhasida ko'rsatiladigan dialog oynasiga nom beradi. Ikkinchi parametr - xabar matni. Funktsiyalar odatda e'tiborga olinmaydigan qatorni qaytaradi.

```
from tkinter import messagebox

messagebox.showinfo("Information","Informative message")
messagebox.showerror("Error", "Error message")
messagebox.showwarning("Warning","Warning message")
```

Tkinter messagebox obyekti shuningdek, foydalanuvchiga oddiy ha/yo‘q turdagi savollarni berish imkonini beradi va savol turiga qarab tugma nomlarini o‘zgartiradi. Bu funksiyalar:

```
from tkinter import messagebox

answer = messagebox.askokcancel("Question","Do you want to open this file?")
answer = messagebox.askretrycancel("Question", "Do you want to try that again?")
answer = messagebox.askyesno("Question","Do you like Python?")
answer = messagebox.askyesnocancel("Question", "Continue playing?")
```

Qaytish qiymati mantiqiy, to'g'ri yoki noto'g'ri, savolga javobdir. Agar "bekor qilish" variant bo'lsa va foydalanuvchi "bekor qilish" tugmasini tanlasa, None qaytariladi.

Yagona qiymatli ma'lumotlarni kiritish

Agar siz foydalanuvchidan bitta ma'lumot qiymatini so'ramoqchi bo'lsangiz, satr, butun yoki suzuvchi nuqta qiymati, siz simpledialog ob'ektdan foydalanishingiz mumkin. Foydalanuvchi so'ralgan qiymatni kiritishi va kiritilgan qiymatni

qaytaradigan "OK" tugmasini bosishi mumkin. Agar foydalanuvchi "Bekor qilish" tugmasini bossa, u None qaytariladi.

```
import tkinter as tk
from tkinter import simpledialog

application_window = tk.Tk()

answer = simpledialog.askstring("Input", "What is your first name?",
                                parent=application_window)
if answer is not None:
    print("Your first name is ", answer)
else:
    print("You don't have a first name?")

answer = simpledialog.askinteger("Input", "What is your age?",
                                 parent=application_window,
                                 minvalue=0, maxvalue=100)
if answer is not None:
    print("Your age is ", answer)
else:
    print("You don't have an age?")

answer = simpledialog.askfloat("Input", "What is your salary?",
                               parent=application_window,
                               minvalue=0.0, maxvalue=100000.0)
if answer is not None:
    print("Your salary is ", answer)
else:
    print("You don't have a salary?")
```

Fayl tanlash

Umumiy vazifa saqlash qurilmasidagi papkalar va fayllar nomlarini tanlashdir. Filedialog Buni ob'ekt yordamida amalga oshirish mumkin . E'tibor bering, bu buyruqlar faylni saqlamaydi yoki yuklamaydi. Ular shunchaki foydalanuvchiga faylni tanlash imkonini beradi. Fayl nomini olganingizdan so'ng, tegishli Python kodidan foydalanib faylni ochishingiz, qayta ishlashingiz va yopishingiz mumkin. Ushbu dialog oynalari har doim faylga to'liq yo'lni o'z ichiga olgan "to'liq malakali fayl nomini" qaytaradi. Shuni ham yodda tutingki, agar foydalanuvchiga bir nechta fayllarni tanlashga ruxsat berilsa, qaytariladigan qiymat

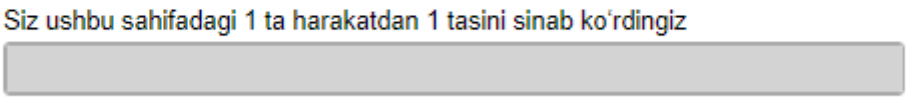
barcha tanlangan fayllarni o'z ichiga olgan kortejdir. Agar foydalanuvchi dialog oynasini bekor qilsa, qaytarilgan qiymat bo'sh qatordir.

```
import tkinter as tk
from tkinter import filedialog
import os

application_window = tk.Tk()

# Build a list of tuples for each file type the file dialog should display
my_filetypes = [('all files', '.*'), ('text files', '.txt')]

# Ask the user to select a folder.
answer = filedialog.askdirectory(parent=application_window,
                                 initialdir=os.getcwd(),
                                 title="Please select a folder:")

# Ask the user to select a single file name.
answer = filedialog.askopenfilename(parent=application_window,
                                    initialdir=os.getcwd(),
                                    title="Please select a file:",
                                    filetypes=my_filetypes)

# Ask the user to select a one or more file names.
answer = filedialog.askopenfilenames(parent=application_window,
                                     initialdir=os.getcwd(),
                                     title="Please select one or more files:",
                                     filetypes=my_filetypes)

# Ask the user to select a single file name for saving.
answer = filedialog.asksaveasfilename(parent=application_window,
                                      initialdir=os.getcwd(),
                                      title="Please select a file name for saving:",
                                      filetypes=my_filetypes)
```

Rang tanlash

Tkinter ranglarni tanlash uchun yoqimli dialog oynasini o'z ichiga oladi. Siz uni ota-oyna va boshlang'ich rang bilan ta'minlaysiz va u ikki xil spetsifikatsiyadagi rangni qaytaradi: 1) qizil rangni ifodalovchi kortej sifatidagi RGB qiymati va 2) veb-sahifalarda ishlatiladigan o'n oltilik qator, masalan bu ham qizil rangni ifodalaydi. Agar foydalanuvchi operatsiyani bekor qilsa, qaytariladigan qiymatlar va .(255, 0, 0)"#FF0000"None None

```
from tkinter import colorchooser

rgb_color, web_color = colorchooser.askcolor(parent=application_window,
                                             initialcolor=(255, 0, 0))
```




**Nazorat savollari**

1. Standart dialog oynalarini yaratish tavsifini tushuntiring.
2. Faylni saqlash uchun dialog oynalarini ko'rsatish qanday qilinadi?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**42-mavzu: Parallel dasturlash**

**Mavzu rejasi:**

1.Parallel vazifalarni bajarish.

2.Vazifalarni rejalashtiruvchi.

Multithreading va GIL

Python tilining dizaynerlari **global tarjimon blokirovkasi (GIL)** yordamida **jarayonda faqat bitta ip haqiqiy Python kodini ishga tushirishi mumkinligini** tanladilar . Bu shuni anglatadiki, boshqa tillarda (C, C++, Fortran) ishlashi mumkin bo'lgan yondashuvlar biroz ehtiyot bo'lmasdan Pythonda ishlamasligi mumkin. Bir qarashda, bu parallelizm uchun yomon. *Lekin hammasi yomon emas!:*

- C yoki boshqa tillarda yozilgan tashqi kutubxonalar (NumPy, SciPy, Pandas va boshqalar) qulfni bo'shatishi va ko'p tarmoqli ishlashi mumkin. Bundan tashqari, ko'pchilik kirish/chiqish GILni chiqaradi va kirish/chiqish sekin.
- Agar tezlik etarlicha muhim bo'lsa, sizga parallel narsalar kerak bo'lsa, siz odatda sof Python dan foydalanmaysiz.

Python bir nechta jarayonlardan foydalanadigan shunga o'xshash narsani qilishning oqilona usuliga ega: **multiprocessing** modul.

- Interfeys juda ko'p ish zarralariga o'xshaydi, lekin fonda global tarjimon blokirovkasini aylanib o'tish uchun yangi jarayonlarni yaratadi.
- dan foydalanishdagi kabi bir xil xavf va qiyinchiliklarga ega bo'lgan past darajadagi funktsiyalar mavjud **threading**.

Misol uchun, ajratish-qo'llash-birlashtirish yoki xaritani qisqartirish paradigmasi ko'plab ilmiy ish oqimlari uchun juda foydali. Sizda shunday deb hisoblang:

```
def square(x):
    return x*x
```

Funktsiyadan foydalanib, funktsiyani ro'yxatdagi har bir elementga qo'llashingiz mumkin **map()**:

```
>>> list(map(square, [1, 2, 3, 4, 5, 6]))
[1, 4, 9, 16, 25, 36]
```

Sinf **multiprocessing.pool.Pool** buni amalga oshirishning ekvivalent, ammo parallellashtirilgan (ko'p ishlov berish orqali) usulini taqdim etadi. Hovuz klassi, sukut bo'yicha, har bir CPU uchun bitta yangi jarayon yaratadi va ro'yxatda parallel hisob-kitoblarni amalga oshiradi:

```
>>> from multiprocessing import Pool
>>> with Pool() as pool:
...     pool.map(square, [1, 2, 3, 4, 5, 6])
[1, 4, 9, 16, 25, 36]
```

```
import random

def sample(n):
    """Make n trials of points in the square.  Return (n, number_in_circle)

    This is our basic function.  By design, it returns everything it\
    needs to compute the final answer: both n (even though it is an input
    argument) and n_inside_circle.  To compute our final answer, all we
    have to do is sum up the n:s and the n_inside_circle:s and do our
    computation"""
    n_inside_circle = 0
    for i in range(n):
        x = random.random()
        y = random.random()
        if x**2 + y**2 < 1.0:
            n_inside_circle += 1
    return n, n_inside_circle

%%timeit
n, n_inside_circle = sample(10**6)

pi = 4.0 * (n_inside_circle / n)
pi
```

Darsdagi **multiprocessing.pool.Pool** koddan foydalanib, sample funktsiyani har birida 10**5 faqat namunalar bilan 10 marta bajaring. Natijalarni birlashtiring va hisoblash vaqtini belgilang. Qabul qilingan vaqtning farqi nimada?

(ixtiyoriy, kengaytirilgan) Xuddi shunday qiling,
lekin **multiprocessing.pool.ThreadPool** uning o‘rniga. Bu bilan bir xil ishlaydi Pool, lekin turli jarayonlar o'rniga iplardan foydalanadi. Qabul qilingan vaqtni solishtiring.

## Nazorat savollari

1. Thread bilan ishlash mohiyati.
2. Paralel dasturlash orqali qanday afzalliklarga erishish mumkin?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**43-mavzu: Ko’p bosqichli  dasturlash**

Mavzu rejasi:

1.Ko'p bosqichli dasturlash.

2.Mahalliy ma'lumotlarni o'tkazish

Dastur - bu foydalanuvchi tomonidan yozilgan ko'rsatmalar ketma-ketligi bo'lib, u kompyuterga biron bir muammoni hal qilish vazifasini bajarishni buyuradi. Endi biz dasturni ishga tushirishdan oldin qanday bosqichlardan o'tishini ko'proq bilib olamiz. Dastur diskda ikkilik bajariladigan fayl sifatida joylashadi. Dasturni ishga tushirish uchun uni diskdan asosiy xotiraga olib kirish kerak (chunki protsessor to'g'ridan-to'g'ri kirishi mumkin bo'lgan yagona saqlash joylari - Asosiy xotira va registrlar - registrga bir yoki undan kamroq protsessor siklida kirishda. , Asosiy xotira ko'p tsikllarni olishi mumkin). Keling, oldingi vazifani bajarish uchun dastur qanday batafsil bosqichlarni bilib olaylik.

Bajarilishi kerak bo'lgan dastur diskdan asosiy xotiraga olib kelinganligi sababli, u protsessorda bajarish uchun mavjud bo'lgan jarayon kontekstiga (asosan bajarilayotgan dastur) joylashtiriladi. Amalga oshirish jarayonida u xotiradan ma'lumotlar va ko'rsatmalarga kiradi va bajarish tugallangach, jarayon tugatiladi va xotira boshqa jarayon tomonidan foydalanish uchun qaytariladi.

Manba kodidagi manzillar odatda ramziydir (o'zgaruvchining soni kabi).  Kompilyator bu manzillarni o'zgartiriladigan manzillarga bog'laydi va ular bog'lovchi yoki yuklovchi tomonidan mutlaq manzillarga bog'lanadi (Binding - bir manzil maydonidan boshqasiga xaritalash).

Ko'rsatmalar va ma'lumotlarni xotira manzillari bilan bog'lash dasturni bajarishning quyidagi bosqichlaridan birida amalga oshirilishi mumkin:

1. **Kompilyatsiya vaqti :**

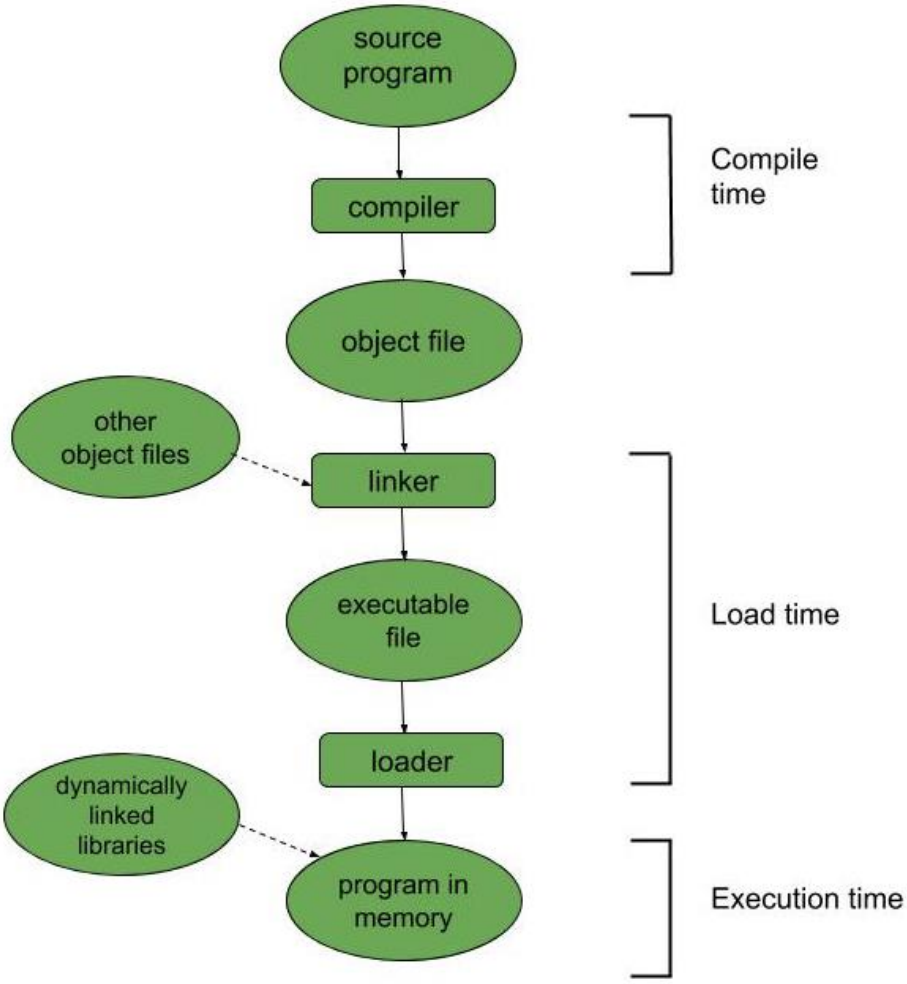
Agar biz dastlab kompilyatsiya vaqtida jarayon xotirada qayerda joylashganligini bilsak, u holda mutlaq kod yaratilishi mumkin. Agar boshlang'ich joylashuvi o'zgarsa, kodni qayta kompilyatsiya qilish kerak.

2. **Yuklash vaqti :**

Agar kompilyatsiya vaqtida jarayon joylashgan xotira manzili noma'lum bo'lsa, kompilyator o'zgartiriladigan kodni yaratishi kerak (ishlash uchun statik xotira manzili yo'q). Agar boshlang'ich manzil o'zgargan bo'lsa, ushbu qiymatni kiritish uchun dasturni qayta yuklash kerak.

3. **Bajarish_vaqti :**

Agar jarayonni bajarish vaqtida bir segmentdan ikkinchisiga o'tkazish mumkin bo'lsa, u holda bog'lash bajarilish vaqtigacha kechiktirilishi kerak. Ushbu turdagi bog'lash uchun maxsus apparat talab qilinadi (quyida muhokama qilinadi).




**Rasm -** Foydalanuvchi dasturini ko'p bosqichli qayta ishlash

**Mantiqiy va jismoniy manzil :**

Mantiqiy manzil protsessor tomonidan yaratilgan manzil, jismoniy manzil esa xotira bloki tomonidan ko'riladigan manzildir (bu manzil xotiraning xotira-manzil registriga yuklangan manzil). Foydalanuvchi dasturi mantiqiy manzillar bilan shug'ullanadi va hech qachon jismoniy manzillarni ko'rmaydi.

Kompilyatsiya vaqtida yoki yuklash vaqtida bog'lanish manzillari (yuqorida muhokama qilinganidek) bir xil mantiqiy va jismoniy manzillarni hosil qiladi, lekin ulanish manzillari bajarilish vaqtida bajarilsa, bunday emas. Ikkinchi holda, yaratilgan jismoniy va mantiqiy manzillar boshqacha. Mantiqiy manzil bu holda **virtual manzil** deb ataladi . **Ishlash vaqtida Xotirani boshqarish birligi (MMU)** deb nomlangan apparat qurilmasi virtual manzildan jismoniy manzilga xaritalashni amalga oshiradi.

**Dinamik yuklash** xotiradan yanada samarali foydalanish uchun ishlatiladi. Dinamik yuklashning afzalligi shundaki, ma'lum bir tartib faqat kerak bo'lganda yuklanadi. Dinamik bog'langan kutubxonalar (DLL) - bu dasturlar ishga tushirilganda dastur bilan bog'langan tizim kutubxonalari. Bu foydalanuvchi dasturining dastlabki kodi bosqichidan boshlab to bajarilishigacha bo'lgan ko'p bosqichli ishlov berishning umumiy qisqacha tavsifi. Ilova qilingan diagrammada dasturning ko'p bosqichli ishlov berish jarayoni aniq tasvirlangan.

## Nazorat savollari

1. Ko'p bosqichli dasturlash bilan ishlash mohiyati.
2.Mahalliy ma'lumotlarni o'tkazish orqali qanday afzalliklarga erishish mumkin?

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое

необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**44-mavzu:  Utilit funksiyalar**

**Mavzu rejasi:**

1.Mavzu voqealari.

2.To'siqlar. Oqim taymer.

3.Utility funktsiyalari

Thread Utility funksiyalari kerak

Jarayon - bu kompyuter dasturining ishlaydigan namunasidir .

Har bir Python dasturi jarayonda bajariladi, bu Python tarjimonining yangi namunasidir. Ushbu jarayon *MainProcess nomiga ega va MainThread* deb nomlangan dastur ko'rsatmalarini bajarish uchun foydalaniladigan bitta ipga ega . Har ikkala jarayon va oqimlar asosiy operatsion tizim tomonidan yaratiladi va boshqariladi.

Ba'zan kodni bir vaqtda bajarish uchun dasturimizda yangi bolalar jarayonlarini yaratishimiz kerak bo'lishi mumkin.

**Python multiprocessing.Process** sinfi orqali yangi jarayonlarni yaratish va boshqarish imkoniyatini beradi .

Ba'zi ilovalarda bizda juda ko'p jarayonlar bo'lishi mumkin va ishlayotgan jarayonlar soni haqida ma'lumotlarga kirishimiz
yoki **multiprocessing** xususiyatlariga kirishimiz kerak bo'lishi mumkin . Muayyan oqimlar uchun jarayon misollari.

**Python dasturidagi jarayonlarga qanday kirishimiz mumkin?**

**Process Utility funksiyalaridan qanday foydalanish**

Python jarayonlari bilan ishlashda foydalanishimiz mumkin bo'lgan bir qator yordamchi dasturlar mavjud.

**Ushbu yordam dasturlari ko'p ishlov** berish moduli funktsiyalari sifatida taqdim etiladi .

Ko'p ishlov berish modulining mashhur yordamchi funktsiyalariga misollar quyidagilar:

- **active_children()** : Joriy jarayon uchun barcha faol bolalar jarayonlari ro'yxatini qaytaradi.
- **cpu_count()** : Tizimdagi mantiqiy protsessor yadrolari sonini qaytaradi.
- **current_process()** : Joriy jarayon uchun jarayon misolini qaytaring.
- **parent_process()** : Joriy jarayonning ota-onasi uchun Jarayon misolini qaytaring.

Kamroq qo'llaniladigan ba'zi qo'shimcha multiprocessing moduli funktsiyalariga quyidagilar kiradi:

- **freeze_support()** : Windowsda muzlatilgan Python bajariladigan fayl uchun ko'p ishlov berish uchun qo'llab-quvvatlash qo'shing.
- **get_all_start_methods()** : Tizimda qo'llab-quvvatlanadigan jarayonni boshlash usullari ro'yxatini qaytaring.

- **get_context()** : Jarayon kontekst obyektini qaytaring.
- **get_start_method()** : Hozirda sozlangan jarayonni boshlash usulini qaytaring.
- **set_executable()** : Bolalar jarayonlari uchun Python bajariladigan faylga yo'lni o'rnating.
- **set_start_method()** : Yangi jarayonlar uchun ishlatiladigan boshlash usulini o'rnating.

Endi biz ko'p ishlov berish moduli funktsiyalari bilan tanishganimizdan so'ng, ba'zi ishlangan misollarni ko'rib chiqaylik.

## Joriy jarayonni qanday olish mumkin

Joriy kod bilan ishlaydigan jarayon uchun **multiprocessing.Process** misolini olishimiz mumkin .

**Bunga multiprocessing.current_process()** funksiyasi orqali erishish mumkin, bu **multiprocessing.Process** misolini qaytaradi .

```
# get the current process
thread = current_process()
```

*MainProcess* uchun **multiprocessing.Process** ga kirish uchun ushbu funksiyadan foydalanishimiz mumkin .

Buni quyida keltirilgan qisqa misol bilan ko'rsatish mumkin.

```
# SuperFastPython.com
# example of executing the current process
from multiprocessing import current_process
# get the current process
process = current_process()
# report details
print(process)
```

Misolni ishga tushirish joriy ishlayotgan jarayon uchun jarayon misolini oladi.

Keyin tafsilotlar haqida xabar beriladi, bu esa biz asosiy jarayonga ega bo'lmagan asosiy jarayonga kirganimizni ko'rsatadi.

**Adabiyotlar:**

1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)
2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.
3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.

**45-mavzu: Python dasturlash tilida arxivlar bilan ishlash**

**Mavzu rejasi:**

1.GZIP algoritmi yordamida siqish va ochish.
2.BZIP2 algoritmi yordamida siqish va dekompressiya.
3.LZMA algoritmi yordamida siqish va dekompressiya

**Ushbu modul GNU dasturlari gzip** va **gunzip** kabi fayllarni siqish va ochish uchun oddiy interfeysni taqdim etadi .

Ma'lumotlarni siqish zlib modul tomonidan ta'minlanadi.

Modul sinfni, shuningdek va qulaylik funksiyalarini gzip taqdim etadi. Sinf **gzip** formatidagi fayllarni o'qiydi va yozadi , ma'lumotlarni avtomatik ravishda siqib chiqaradi yoki oddiy fayl ob'ekti kabi ko'rinadi .GzipFileopen()compress()decompress()GzipFile

**Esda tutingki, gzip** va **gunzip** dasturlari tomonidan ochiladigan qo'shimcha fayl formatlari, masalan, **kompres** va **pack** tomonidan ishlab chiqarilganlar , bu modul tomonidan qo'llab-quvvatlanmaydi.

Siqilgan faylni o'qishga misol:

```
import gzip
with gzip.open('/home/joe/file.txt.gz', 'rb') as f:
    file_content = f.read()
```

Siqilgan GZIP faylini qanday yaratishga misol:

```
import gzip
content = b"Lots of content here"
with gzip.open('/home/joe/file.txt.gz', 'wb') as f:
    f.write(content)
```

Mavjud faylni GZIP siqish usuliga misol:

```
import gzip
import shutil
with open('/home/joe/file.txt', 'rb') as f_in:
    with gzip.open('/home/joe/file.txt.gz', 'wb') as f_out:
        shutil.copyfileobj(f_in, f_out)
```

Ikkilik qatorni GZIP siqish usuliga misol:

```
import gzip
s_in = b"Lots of content here"
s_out = gzip.compress(s_in)
```

Ikki tomonlama siqishni ko'rsatish uchun `compress()` va foydalanish :`decompress()`

```
>>> import bz2
>>> data = b"""\
... Donec rhoncus quis sapien sit amet molestie. Fusce scelerisque vel augue
... nec ullamcorper. Nam rutrum pretium placerat. Aliquam vel tristique lorem,
... sit amet cursus ante. In interdum laoreet mi, sit amet ultrices purus
... pulvinar a. Nam gravida euismod magna, non varius justo tincidunt feugiat.
... Aliquam pharetra lacus non risus vehicula rutrum. Maecenas aliquam leo
... felis. Pellentesque semper nunc sit amet nibh ullamcorper, ac elementum
... dolor luctus. Curabitur lacinia mi ornare consectetur vestibulum."""
>>> c = bz2.compress(data)
>>> len(data) / len(c)  # Data compression ratio
1.513595166163142
>>> d = bz2.decompress(c)
>>> data == d  # Check equality to original object after round-trip
True
```

`BZ2Compressor` Qo'shimcha siqish uchun foydalanish :

```
>>> import bz2
>>> def gen_data(chunks=10, chunksize=1000):
...     """Yield incremental blocks of chunksize bytes."""
...     for _ in range(chunks):
...         yield b"z" * chunksize
...
>>> comp = bz2.BZ2Compressor()
>>> out = b""
>>> for chunk in gen_data():
...     # Provide data to the compressor object
...     out = out + comp.compress(chunk)
...
>>> # Finish the compression process.  Call this once you have
>>> # finished providing data to the compressor.
>>> out = out + comp.flush()
```

Yuqoridagi misolda juda “tasodifiy” ma’lumotlar oqimi ( b”z” bo’laklar oqimi) qo’llaniladi. Tasodifiy ma’lumotlar yomon siqilishda moyil bo’ladi, tartibli, takrorlanuvchi ma’lumotlar odatda yuqori siqish nisbatini beradi.

## Foydalanilgan adabiyotlar


1. Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. -2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. -(Самое необходимое)

2. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил. — Парад, тит. англ.

3. Изучаем Python, том 2, 5-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил. — Парал. тит. англ.