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

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

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

**O‘quv savollari:**

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

**1. Arifmetik operatorlar.**

Python barcha turdagi arifmetik operatsiyalarni qo‘llab-quvvatlaydi :

**Qo‘shish (+):** print(6 + 2)   # 8

**Ayirish (-):** print(6 - 2)   # 4

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

**Bo‘linish (/):** print(6/2)   # 3.0

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

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

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

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

Operatsiya   yo‘nalishi:

2-jadval

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

Keling, quyidagi ifodani ko‘rib chiqamiz:

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

print(raqam)   # 110

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

 Qavslar amallar tartibini qayta aniqlash uchun ishlatilishi mumkin:

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

print(raqam)   # 224

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

**Arifmetik amallarning tenglik belgisi bilan ishlatilishi**

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

3-jadval

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

**Operatsiyalarni dasturda qo‘llanilishi:**

raqam = 10

raqam += 5

print(raqam)   # 10 + 5 = 15

raqam -= 3

print(raqam)   # 15 - 3 = 12

raqam \*= 4

print(raqam)   # 12 \* 4 = 48

raqam /= 2

print(raqam)   # 48 / 2 = 24.0

raqam //= 5

print(raqam)   # 24 // 5 = 4

raqam \*\*= 3

print(raqam)   # 4 \*\* 3 = 64

raqam %= 6

print(raqam)   # 64 % 6 = 4

**Yaxlitlash va “round” funksiyasi**

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

bir\_son = 2.0001

ikki\_son = 5

uch\_son = bir\_son / ikki\_son

print(uch\_son) # 0.40002000000000004

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

print(2.0001 + 0.2)   # 2.1002000000000003

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

bir\_son = 2.0001

ikki\_son = 0.1

uch\_son = bir\_son + ikki\_son

print(round(uch\_son))  # 2

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

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

bir\_son = 2.0001

ikki\_son = 0.1

uch\_son = bir\_son + ikki\_son

print(round(uch\_son, 4))  # 2.1001

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

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

Yaxlitlash misollari:

# butun songacha yaxlitlash

print(round(2.49))

print(round(2.51))

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

print(round(2.5))    # 2

print(round(3.5))    # 4

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

# verguldan keyingi 2 ta raqamgacha yaxlitlash

print(round (2.554, 2))       # 2.55

print(round (2.5551, 2))       # 2.56

print(round (2.554999, 2))    # 2.55

print(round (2.499, 2))       # 2.5

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

# verguldan keyingi 2 ta raqamgacha yaxlitlash

print(dumaloq (2.545, 2))    # 2.54

print(dumaloq(2.555, 2))  # 2.56

print(dumaloq (2.565, 2))    # 2.56

print(dumaloq (2.575, 2))    # 2.58

print( dumaloq(2.655, 2))    # 2.65

print(dumaloq (2.665, 2))    # 2.67

print(dumaloq (2.675, 2))    # 2.67

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

**O‘zgaruvchilar**

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

*False await else import pass*

*None break except in raise*

*True class finally is return*

*and continue for lambda try*

*as def from nonlocal while*

*assert del global not with*

*async elif if or yield*

O‘zgaruvchilarni e’lon qilish quyidagicha amalga oshiriladi:

foydalanuvchi = "Tomson"

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

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

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

userName = "Tomson"

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

user\_name = "Tomson"

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

# ikki xil oʻzgaruvchi

ism = "Jenifer"

Ism = "Loktfud"

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

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

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

print(name)     # print: Tomson

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

print(name)     # chiqish: Bobbi

**Ma’lumotlar turlari**

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

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

isMarried = False

print(isMarried)     # False

isAlive = True

print(isAlive)       # True

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

age = 21

print("Yosh:", age)     # Yosh: 21

count = 15

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

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

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

a = 0b11

b = 0b1011

c = 0b100001

print(a)     # 3 o‘nlik sanoq sistemasida

print(b)     # 11 o‘nlik sanoq sistemasida

print(c)     # 33 o‘nlik sanoq sistemasida

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

a = 007

b = 0011

c = 0017

print(a)     # 7 o‘nlik sanoq sistemasida

print(b)     # 9 o‘nlik sanoq sistemasida

print (c)    # 15 o‘nlik sanoq sistemasida

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

a = 0x0A

b = 0xFF

c = 0xA1

print(a)     # 10 o‘nlik sanoq sistemasida

print(b)     # 255 o‘nlik sanoq sistemasida

print(c)     # 161 o‘nlik sanoq sistemasida

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

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

uzunligi = 1.68

p = 3.14

vazn = 68.

print (uzunligi)    # 1.68

print(p)        # 3.14

print(vazn)    # 68.0

kasr sonini eksponensial belgida aniqlash mumkin:

x = 3.9e3

print(x)   # 3900.0

x = 3.9e-3

print(x)   # 0.0039

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

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

hi = "Salom dunyo!"

print (hi )   # Salom dunyo!

ism = "Tomson"

print (ism)   # Tomson

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

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

f\_Id = "abc"   # satr turiga mansub

print (f\_Id)

f\_Id = 234   # butun sonlar turiga mansub

print (f\_Id)

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

f\_Id = "abc"       # tip str

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

f\_Id = 234         # tip  int

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

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

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

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

4-jadval

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

 Taqqoslash operatsiyalariga misollar:

a = 6

b = 9

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

print(res)  # False

print(a > b)  # False

print(a != b)  # True

print(a < b)  # True

boolean1 = True

boolean2 = False

print(boolean1 == boolean2) # False

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

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

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

x and y

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

yosh = 20

vazn= 65

res = yosh > 19 and vazn == 65

print(res) # True

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

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

natija = 14 va "s"

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

#shuning uchun oxirgi operandning qiymati qaytariladi

natija = 0 va "s"

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

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

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

x or y

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

yosh = 22

natija = yosh > 21 or False

print(natija) # True

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

res = 4 or "f"

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

natija = 0 or "f"

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

**not (mantiqiy inkor - !)**

Ifoda False bo‘lsa, True qiymatini qaytaradi

yosh = 25

isSingle = False

print(not yosh > 21)   # False

print(not isSingle)   # True

print(not 45)   # False

print(not 0)   # True

**in operatori**

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

qiymat **in** [qiymat\_to‘plamidagi]

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

xabar = "hello world!"

hi = "hello"

**print**(hi in xabar)

zol= "golden"

**print**(zol in xabar)

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

xabar = "hello world!"

hi = "hello"

**print**(hi not in xabar)  # False

golden = " golden "

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

**Nazorat savollari:**

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