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

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

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

**O‘quv savollari:**

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

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

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

msg = ‘Salom Dunyo!’

**print**(msg)   # Salom Dunyo!

nomi="Jeyson"

**print**(nomi)   # Jeyson

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

text = ("Salom hurmatli kursantlar"

"Bugun biz sizlar bilan"

"Satrlar ustida amallar bajarib ko‘ramiz"

)

**print**(text)

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

‘‘‘

Bu izoh

‘‘‘

text = ‘‘‘Salom hurmatli kursantlar

Bugun biz sizlar bilan

Satrlar ustida amallar

bajarib ko‘ramiz

‘‘‘

print(text)

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

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

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

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

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

**print**(text)

Dasturning konsoldagi ko‘rinishi:

Xabar:

"Salom Dunyo"

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

URL = "D:\python\_course\name.txt"

**print**(URL)

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

c:\python\_course

ame.txt

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

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

**print**(URL)

c:\python\_course\name.txt

**Qatorga qiymatlarni kiritish**

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

user\_name = "Jeyson"

user\_age = 22

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

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

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

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

str = "hello world!"

belgi = str[0] # h

**print**(belgi)

belgi = str[6] # w

**print**(belgi)

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

**print**(belgi)

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

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

str = "salom dunyo!"

belgi = str[-1]

**print**(belgi) #!

belgi2 = str[-5]

**print**(belgi2) #u

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

str = "Salom DASTURCHILAR!"

str[1] = "R"

TypeError:'str' object does not support item assignment

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

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

str = "salom dunyo"

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

    print(belgi)

s

a

l

o

m

d

u

n

y

o

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

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

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

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

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

str = "salom dunyo"

# 0-indeksdan 5 - indeksgacha

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

**print**(sub\_str1)       # salom

# 2-indeksdan 5-indeksgacha

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

**print**(sub\_str2)       #llo

# 2 - indeksdan 9-indeksgacha oraliqda 2 qadam bilan

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

**print**(sub\_str3)       # lmdn

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

name = "Jeyson"

surname = "Stethom"

full\_name = name + " " + surname

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

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

name=" Jeyson "

age = 38

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

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

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

print("char" \* 3)   # charcharchar

print("u" \* 4)   # uuuu

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

str1 = "1a"

str2 = "aa"

str3 = "aa"

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

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

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

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

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

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

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

str1 = "Tomas"

str2 = "TOMAS"

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

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

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

print(ord("B"))      # 66

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

str = "salom dunyo!"

uzunlik = **len**(str)

print(uzunlik)    # 12

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

str = "salom dunyo"

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

print(is\_true)     # rost

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

print(is\_true)     # False

**Satr bilan ishlovchi operatorlar va metodlardan foydalanish**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

str = input(" Raqamni kiriting :")

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

    raqam = **int**(str)

    print(raqam)

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

f = "hello.py"

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

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

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

str = "    Salom   dunyo!   "

str = str.**strip**()

print(str)            # salom   dunyo!

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

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

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

Konsol chiqishi:

  Redmi 7:       500

Nokia N10:     360

**Satrda qidirish**

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

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

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

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

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

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

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

**print**(indeks)        #6

# 10 indeksdan qidirish

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

**print**(indeks)        #21

# 10 dan 15 gacha indekslarni qidirish

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

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

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

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

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

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

*# defislarni bo‘sh joylar bilan almashtirish*

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

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

*# defislarni olib tashlash*

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

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

*# faqat birinchi defisni almashtirish*

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

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

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

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

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

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

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

# probellardan ajratiladi

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

print(splitted\_text)

print(splitted\_text[6]) #singan

# vergul ustida tanaffus

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

print(splitted\_text)

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

# birinchi besh boʻshliqqa boʻlingan

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

**print**(bo‘lingan\_matn)

**print**(bo‘lingan\_matn[5])

# shoxlari singan va po‘stlog‘i singan

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

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

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

*# ajratuvchi - bo‘sh joy*

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

**print**(sentence)

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

*# ajratuvchi  - vertikal chiziq*

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

**print**(sentence)

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

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

str = "hello"

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

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

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

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

first\_name="Jeyson"

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

print(text) # Hello, Jeyson.

name="Bobo"

age=73

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

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

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

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

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

print(txt) # Salom, Jeyson.

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

print(info) # Ismi: Bobo Yoshi: 73

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

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

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

print(info) # Ismi: Bobo Age: 73

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

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

Salom, Jeyson Jeyson Jeyson

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

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

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

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

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

 To‘ldiruvchining umumiy sintaksisi quyidagicha:

{:[maxsus to‘ldiruvchi] }

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

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

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

hi = "Salom sizga {:s}"

name = "Jeyson"

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

print(formatted\_txt) # Salom sizga Jeyson

Format() usuli natijada yangi formatlangan qatorni qaytaradi.

 {:d} - butun son formatlashga misol:

txt = "{:d} ta simvol"

number = 12

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

print(formatted\_txt) # 12 ta simvol

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

txt = "{:,d} belgi"

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

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

n = 300

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

print(txt) # 200 belgi

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

num = 115.2548695

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

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

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

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

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

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

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

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

n1 = 115.2548695

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

n2 = 26

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

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

num = .123456

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

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

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

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

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

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

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

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

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

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

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

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

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

raqam = 23.8689578

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

*# 23.87   - 2.386896 e +01*

**Nazorat savollari:**

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