**Standart funksiyalar**

**Python’da daraja va ildiz**

**math modulining pow metodi**

math modulining pow metodi berilgan sonni darajaga ko‘targan qiymatni qaytaradi. Metodning birinchi parametrida son, ikkinchisida esa uni qaysi darajaga ko‘tarish kerakligi ko‘rsatiladi. \*\* operatoridan farqli o‘laroq, math.pow har doim haqiqiy son (float) qaytaradi.

**Sintaksis:**

import **math**

**math**.**pow**(son, daraja)

**Misol 1:** 3 sonini kvadratga ko‘tarish

import **math**

**print**(**math**.**pow**(3, 2))

**Natija:**

9.0

**Misol 2:** Sonni manfiy darajaga ko‘tarish

import **math**

**print**(**math**.**pow**(3, -2))

**Natija:**

0.1111111111111111

**Misol 3:** Manfiy sonni kubga ko‘tarish

import **math**

**print**(**math**.**pow**(-3, 3))

**Natija:**

-27.0

**math modulining sqrt metodi**

math modulining sqrt metodi berilgan sonning kvadrat ildizini qaytaradi. Har doim haqiqiy son (float) qaytariladi.

**Sintaksis:**

import **math**

**math**.**sqrt**(son)

**Misol 1:** 25 sonining kvadrat ildizini topish

import **math**

**print**(**math**.**sqrt**(25))

**Natija:**

5.0

**Misol 2:** 10 sonining kvadrat ildizini topish

import **math**

**print**(**math**.**sqrt**(10))

**Natija:**

3.1622776601683795

**Misol 3:** Manfiy sonning kvadrat ildizini olishga urinib ko‘rish

import **math**

**print**(**math**.**sqrt**(-100))

**Natija:**

ValueError: math domain error

**Python’da yaxlitlash funksiyalari**

**round funksiyasi**

round funksiyasi birinchi parametr sifatida berilgan o‘nlik sonni matematik qoidalar bo‘yicha eng yaqin butun songa yoki kerakli o‘nlik raqamlar soniga yaxlitlaydi. Ikkinchi parametr majburiy emas va u sonning kasr qismini nechta raqamgacha qoldirish kerakligini belgilaydi.

**Sintaksis:**

**round**(son, [o'nlik raqamlar soni])

**Misol 1:** 3.458 sonini butun songa yaxlitlash

**print**(**round**(3.458))

**Natija:**

3

**Misol 2:** 3.6771 sonini butun songa yaxlitlash

**print**(**round**(3.6771))

**Natija:**

4

**Misol 3:** 3.458 sonini 2-o‘nlik raqamgacha yaxlitlash

**print**(**round**(3.458, 2))

**Natija:**

3.46

**Misol 4:** -1.4567 sonini 3-o‘nlik raqamgacha yaxlitlash

**print**(**round**(-1.4567, 3))

**Natija:**

-1.457

**math modulining floor metodi**

math.floor metodi berilgan sonni har doim pastga qarab butun songa yaxlitlaydi.

**Sintaksis:**

import **math**

**math**.**floor**(son)

**Misol 1:** 25.78 sonini pastga qarab yaxlitlash

import **math**

**print**(**math**.**floor**(25.78))

**Natija:**

25

**Misol 2:** -4.88 sonini pastga qarab yaxlitlash

import **math**

**print**(**math**.**floor**(-4.88))

**Natija:**

-5

**math modulining ceil metodi**

math.ceil metodi berilgan sonni har doim yuqoriga qarab butun songa yaxlitlaydi.

**Sintaksis:**

import **math**

**math**.**ceil**(son)

**Misol 1:** 25.38 sonini yuqoriga qarab yaxlitlash

import **math**

**print**(**math**.**ceil**(25.38))

**Natija:**

26

**Misol 2:** -5.25 sonini yuqoriga qarab yaxlitlash

import **math**

**print**(**math**.**ceil**(-5.25))

**Natija:**

-5

**Python’da ekstremal sonlar**

**min funksiyasi**

min funksiyasi parametr sifatida berilgan sonlar guruhidagi eng kichik sonni qaytaradi. Sonlar guruhini vergul bilan ajratilgan holda berish yoki ro‘yxat, tuple, set kabi ketma-ketlik shaklida berish mumkin.

**Sintaksis:**

**min**(guruh yoki ketma-ketlik)

**Misol 1:** Sonlar guruhidan eng kichik sonni topish

num1 = 2

num2 = 5

num3 = 10

**print**(**min**(num1, num2, num3))

**Natija:**

2

**Misol 2:** Manfiy sonlar guruhidan eng kichik sonni topish

num1 = -1

num2 = -3

num3 = -6

**print**(**min**(num1, num2, num3))

**Natija:**

-6

**Misol 3:** Ro‘yxat orqali eng kichik sonni topish

lst = [10, 5, 4]

**print**(**min**(lst))

**Natija:**

4

**Misol 4:** Tuple orqali eng kichik sonni topish

tlp = (-2, 6, 9)

**print**(**min**(tlp))

**Natija:**

-2

**max funksiyasi**

max funksiyasi parametr sifatida berilgan sonlar guruhidagi eng katta sonni qaytaradi. Sonlar guruhini vergul bilan ajratilgan holda berish yoki ro‘yxat, tuple, set kabi ketma-ketlik shaklida berish mumkin.

**Sintaksis:**

**max**(guruh yoki ketma-ketlik)

**Misol 1:** Sonlar guruhidan eng katta sonni topish

num1 = 2

num2 = 5

num3 = 10

**print**(**max**(num1, num2, num3))

**Natija:**

10

**Misol 2:** Manfiy sonlar guruhidan eng katta sonni topish

num1 = -1

num2 = -3

num3 = -6

**print**(**max**(num1, num2, num3))

**Natija:**

-1

**Misol 3:** Ro‘yxat orqali eng katta sonni topish

lst = [10, 5, 4]

**print**(**max**(lst))

**Natija:**

10

**Misol 4:** Tuple orqali eng katta sonni topish

tlp = (-2, 6, 9)

**print**(**max**(tlp))

**Natija:**

9

**Python’da tasodifiy sonlar (random)**

**random modulining random metodi**

random modulining random metodi 0.0 dan 1.0 gacha bo‘lgan pseudo-tasodifiy sonni qaytaradi. Metodga parametr berilmaydi va har chaqirilganda yangi son hosil bo‘ladi.

**Sintaksis:**

import **random**

**random**.random()

**Misol:** Pseudo-tasodifiy sonni chiqarish

import **random**

**print**(**random**.random())

**Natija (misol):**

0.28387701784150676

**random modulining randint metodi**

randint metodi berilgan oraliqdagi pseudo-tasodifiy butun sonni qaytaradi. Birinchi parametr oraliqning boshlanishi, ikkinchi parametr esa oraliqning oxiri.

**Sintaksis:**

import **random**

**random**.randint(boshlangich, oxir)

**Misol 1:** 0 dan 20 gacha butun son hosil qilish

import **random**

**print**(**random**.randint(0, 20))

**Natija (misol):**

19

**Misol 2:** Manfiy sonlar oraliqida butun son hosil qilish

import **random**

**print**(**random**.randint(-15, -5))

**Natija (misol):**

-6

**random modulining uniform metodi**

uniform metodi berilgan oraliqdagi pseudo-tasodifiy haqiqiy sonni (float) qaytaradi. Birinchi parametr oraliqning boshlanishi, ikkinchi parametr esa oraliqning oxiri.

**Sintaksis:**

import **random**

**random**.uniform(boshlangich, oxir)

**Misol 1:** 0 dan 10 gacha haqiqiy son hosil qilish

import **random**

**print**(**random**.uniform(0, 10))

**Natija (misol):**

4.840831811035069

**Misol 2:** Manfiy sonlar oraliqida haqiqiy son hosil qilish

import **random**

**print**(**random**.uniform(-4, -1))

**Natija (misol):**

-3.2516379041520467

**random modulining randrange metodi**

randrange metodi berilgan oraliqdan tasodifiy sonni qaytaradi. Birinchi parametr (majburiy emas) boshlanish, ikkinchi parametr (majburiy) oxir, uchinchi parametr (majburiy emas) esa qadamni belgilaydi.

**Sintaksis:**

import **random**

**random**.randrange([boshlangich], oxir, [qadam])

**Misol 1:** 1 dan 10 gacha, qadam=2 bo‘yicha tasodifiy son

import **random**

**print**(**random**.randrange(1, 10, 2))

**Natija (misol):**

5

**Misol 2:** 1 dan 10 gacha, faqat boshlanish va oxir ko‘rsatib

import **random**

**print**(**random**.randrange(1, 10))

**Natija (misol):**

4

**Misol 3:** Faqat oxir parametrini ko‘rsatib (0 dan 10 gacha)

import **random**

**print**(**random**.randrange(10))

**Natija (misol):**

5

**random modulining choice metodi**

choice metodi berilgan ketma-ketlikdan (string, ro‘yxat, tuple) tasodifiy elementni qaytaradi.

**Sintaksis:**

import **random**

**random**.choice(ketma\_ketlik)

**Misol 1:** Stringdan tasodifiy element olish

txt = 'abcde'

**print**(random.choice(txt))

**Natija (misol):**

*'e'*

**Misol 2:** Ro‘yxatdan tasodifiy element olish

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

**print**(random.choice(lst))

**Natija (misol):**

5

**Misol 3:** Tupledan tasodifiy element olish

tpl = ('1', '2', '3', '4', '5')

**print**(random.choice(tpl))

**Natija (misol):**

*'4'*

**random modulining sample metodi**

sample metodi berilgan ketma-ketlikdan tasodifiy elementlar tanlovini qaytaradi. Birinchi parametr ketma-ketlik, ikkinchi parametr esa tasodifiy tanlanadigan elementlar soni. Python 3.9 va undan keyingi versiyalarda uchinchi ixtiyoriy parametr counts yordamida ayrim elementlarni ko‘p tanlash imkoniyati mavjud.

**Sintaksis:**

import **random**

**random**.sample(ketma\_ketlik, element\_soni, [k=counts])

**Misol 1:** Ro‘yxatdan 3 tasodifiy element tanlash

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

**print**(random.sample(lst, 3))

**Natija (misol):**

[2, 1, 5]

**Misol 2:** Tupledan 2 tasodifiy element tanlash

tpl = (1, 2, 3, 4, 5)

**print**(random.sample(tpl, 2))

**Natija (misol):**

[3, 2]

**Misol 3:** range yordamida ketma-ketlikdan 3 element tanlash

**print**(random.sample(**range**(0, 10), 3))

**Natija (misol):**

[0, 8, 9]

**Misol 4:** Ro‘yxatdan elementlarni counts yordamida takrorlash imkoniyati bilan tanlash (Python 3.9+)

lst = [1, 2, 3]

**print**(random.sample(lst, counts=[2, 3, 4], k=3))

Bu kod quyidagi ro‘yxat bilan teng:

lst = [1, 1, 2, 2, 2, 3, 3, 3, 3]

**print**(random.sample(lst, 3))

**Misol 5:** Setdan element tanlash (Python 3.9 va undan keyingi versiyalarda ogohlantirish beradi)

st = {1, 2, 3, 4, 5}

**print**(random.sample(st, 2))

**Natija (misol):**

[5, 2]  
DeprecationWarning: Sampling *from* a set deprecated

**random modulining shuffle metodi**

shuffle metodi o‘zgartiriladigan ketma-ketlikdagi elementlarning tartibini aralashtiradi. U faqat **ro‘yxat** bilan ishlaydi, chunki tuple o‘zgarmas va set tartibsiz. Metod ishlagach, asl ro‘yxat o‘zgartiriladi va metod None qaytaradi.

**Sintaksis:**

import **random**

**random**.shuffle(royxat)

**Misol 1:** Ro‘yxat elementlarini aralashtirish

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

random.shuffle(lst)

**print**(lst)

**Natija (misol):**

[4, 3, 2, 1, 5]

**Misol 2:** Tuple elementlarini aralashtirish (xato)

tpl = ('1', '2', '3', '4', '5')

random.shuffle(tpl)

**print**(tpl)

**Natija:**

TypeError: 'tuple' *object* does *not* support item assignment

**Misol 3:** Set elementlarini aralashtirish (xato)

st = {'a', 'b', 'c', 'd'}

random.shuffle(st)

**print**(st)

**Natija:**

TypeError: 'set' *object is not* subscriptable

**random modulining seed metodi**

seed metodi tasodifiy sonlar generatorini ma’lum bir boshlang‘ich qiymat bilan ishga tushiradi yoki saqlaydi. Shu orqali tasodifiy sonlar har safar bir xil ketma-ketlikda hosil qilinadi. Agar parametr berilmasa, har safar yangi son hosil bo‘ladi.

**Sintaksis:**

import **random**

**random**.seed(boshlangich\_raqam)

**Misol 1:** Tasodifiy sonni boshlang‘ich qiymat bilan hosil qilish

import **random**

**random**.seed(5)

**print**(**random**.random())

**Natija:**

0.6229016948897019

**Misol 2:** Bir xil boshlang‘ich qiymat bilan sonni qayta hosil qilish

import **random**

**random**.seed(5)

**print**(**random**.random())

**random**.seed(5)

**print**(**random**.random())

**Natija:**

0.6229016948897019  
0.6229016948897019

**Pythonda sonning moduli**

**abs funksiyasi**

abs funksiyasi sonning modulini qaytaradi, ya’ni manfiy sonni musbatga aylantiradi.

**Sintaksis:**

**abs**(son)

**Misol 1:** -5 sonining modulini chiqarish

num = -5

**print**(**abs**(num))

**Natija:**

5

**Misol 2:** 10 sonining modulini chiqarish

num = 10

**print**(**abs**(num))

**Natija:**

10

**Misol 3:** -2.5 haqiqiy sonining modulini chiqarish

num = -2.5

**print**(**abs**(num))

**Natija:**

2.5

**Python’da sonlar bilan matematik amallar**

**sum metodi**

sum metodi sonlar ketma-ketligidagi barcha elementlar yig‘indisini qaytaradi. Qaytariladigan qiymat qo‘shilayotgan sonlar turiga bog‘liq.

**Sintaksis:**

**sum**(sonlar\_ketma\_ketligi)

**Misol 1:** Butun sonlar ketma-ketligi yig‘indisi

lst = [1, 2, 3]

**print**(**sum**(lst))

**Natija:**

6

**Misol 2:** Haqiqiy sonlar ketma-ketligi yig‘indisi

lst = [2.1, 4.2, 4.3]

**print**(**sum**(lst))

**Natija:**

12

**math modulining fsum metodi**

fsum metodi sonlar ketma-ketligidagi barcha elementlar yig‘indisini qaytaradi. Qaytariladigan qiymat har doim haqiqiy son (float) bo‘ladi.

**Sintaksis:**

import **math**

**math**.**fsum**(sonlar\_ketma\_ketligi)

**Misol 1:** Ro‘yxatdagi sonlar yig‘indisi

import **math**

lst = [1, 2, 3]

**print**(**math**.**fsum**(lst))

**Natija:**

6.0

**Misol 2:** Manfiy sonlar ketma-ketligi yig‘indisi

import **math**

lst = [-2, -4, -6]

**print**(**math**.**fsum**(lst))

**Natija:**

-12.0

**math modulining factorial metodi**

factorial metodi berilgan sonning faktorialini hisoblaydi, ya’ni 1 dan shu songacha bo‘lgan barcha tabiiy sonlar ko‘paytmasini qaytaradi. Parametr sifatida faqat musbat son berish mumkin, aks holda xato chiqadi.

**Sintaksis:**

import **math**

**math**.**factorial**(son)

**Misol 1:** 5 sonining faktoriali

import **math**

**print**(**math**.**factorial**(5))

**Natija:**

120

**Misol 2:** -4 sonining faktorialini hisoblash (xato)

import **math**

**print**(**math**.**factorial**(-4))

**Natija:**

ValueError: factorial() *not* defined *for* negative values

**Python’da sonlarni bo‘lish**

**math modulining remainder metodi**

remainder metodi bir sonni ikkinchisiga bo‘lgandagi qoldiqni qaytaradi. Birinchi parametr — bo‘linadigan son, ikkinchi parametr — bo‘luvchi. % operatoridan farqli o‘laroq, metod har doim haqiqiy son (float) qaytaradi.

**Sintaksis:**

import **math**

**math**.**remainder**(bo‘linuvchi, boluvchi)

**Misol 1:** 25 ni 5 ga bo‘lgandagi qoldiq

import **math**

**print**(**math**.**remainder**(25, 5))

**Natija:**

0.0

**Misol 2:** 30 ni 4 ga bo‘lgandagi qoldiq

import **math**

**print**(**math**.**remainder**(30, 4))

**Natija:**

-2.0

**Misol 3:** 10 ni 3 ga bo‘lgandagi qoldiq

import **math**

**print**(**math**.**remainder**(10, 3))

**Natija:**

1.0

**Misol 4:** 10 ni 0 ga bo‘lish (xato)

import **math**

**print**(**math**.**remainder**(10, 0))

**Natija:**

ValueError: math domain error

**math modulining fmod metodi**

fmod metodi haqiqiy sonlarni bo‘lgandagi qoldiqni qaytaradi. Birinchi parametr — bo‘linadigan son, ikkinchi parametr — bo‘luvchi.

**Sintaksis:**

import **math**

**math**.**fmod**(bo‘linuvchi, bo‘luvchi)

**Misol 1:** 10.8 ni 3 ga bo‘lgandagi qoldiq

import **math**

**print**(**math**.**fmod**(10.8, 3))

**Natija:**

1.8000000000000007

**Misol 2:** -4.56 ni 3 ga bo‘lgandagi qoldiq

import **math**

**print**(**math**.**fmod**(-4.56, 3))

**Natija:**

-1.5599999999999996

**divmod funksiyasi**

divmod funksiyasi ikki sonni bo‘lgandagi butun bo‘linma va qoldiqni juftlik (tuple) ko‘rinishda qaytaradi. Birinchi parametr — bo‘linadigan son, ikkinchi parametr — bo‘luvchi.

**Sintaksis:**

**divmod**(bo‘linuvchi, boluvchi)

**Misol 1:** 20 ni 5 ga bo‘lgandagi natija

**print**(**divmod**(20, 5))

**Natija:**

(4, 0)

**Misol 2:** 21 ni 5 ga bo‘lgandagi natija

**print**(**divmod**(21, 5))

**Natija:**

(4, 1)

**Misol 3:** -15 ni 2 ga bo‘lgandagi natija

**print**(**divmod**(-15, 2))

**Natija:**

(-8, 1)

**math modulining modf metodi**

modf metodi berilgan sonning kasr va butun qismini juftlik (tuple) ko‘rinishda qaytaradi.

**Sintaksis:**

import **math**

**math**.**modf**(son)

**Misol 1:** 20 sonining kasr va butun qismini aniqlash

import **math**

**print**(**math**.**modf**(20))

**Natija:**

(0.0, 20.0)

**Misol 2:** 20.56 sonining kasr va butun qismini aniqlash

import **math**

**print**(**math**.**modf**(20.56))

**Natija:**

(0.5599999999999987, 20.0)

**Python’da belgilar registri**

**lower metodi**

lower metodi satrdagi barcha belgilarni kichik harfga o‘zgartiradi. Metod parametr qabul qilmaydi.

**Sintaksis:**

satr.lower()

**Misol 1:**

txt = 'AbcDef'

**print**(txt.**lower**())

**Natija:**

*'abcdef'*

**Misol 2:**

txt = 'AbС DeА'

**print**(txt.**lower**())

**Natija:**

*'abс deа'*

**upper metodi**

upper metodi satrdagi barcha belgilarni katta harfga o‘zgartiradi. Metod parametr qabul qilmaydi.

**Sintaksis:**

satr.upper()

**Misol 1:**

txt = 'AbcDef'

**print**(txt.**upper**())

**Natija:**

*'ABCDEF'*

**Misol 2:**

txt = 'AbС DeА'

**print**(txt.**upper**())

**Natija:**

*'ABС DEА'*

**swapcase metodi**

swapcase metodi satrdagi barcha belgilar registrini qarama-qarshi holatga o‘zgartiradi. Katta harflar kichik harfga, kichik harflar esa katta harfga aylanadi. Metod parametr qabul qilmaydi.

**Sintaksis:**

satr.swapcase()

**Misol 1:**

txt = 'abcdef'

**print**(txt.**swapcase**())

**Natija:**

*'ABCDEF'*

**Misol 2:**

txt = 'AbcDef'

**print**(txt.**swapcase**())

**Natija:**

*'aBCdEF'*

**Misol 3:**

txt = 'AbС DeА'

**print**(txt.**swapcase**())

**Natija:**

*'aBс dEf'*

**capitalize metodi**

capitalize metodi satrning birinchi belgisini katta harfga o‘zgartiradi, qolgan barcha belgilarni kichik harfga aylantiradi.

**Sintaksis:**

satr.capitalize()

**Misol 1:**

txt = 'abcde'

**print**(txt.**capitalize**())

**Natija:**

*'Abcde'*

**Misol 2:**

txt = 'abc def'

**print**(txt.**capitalize**())

**print**(txt.**title**())

**Natija:**

*'Abc def'*'Abc Def'

**casefold metodi**

casefold metodi satrdagi barcha belgilarni kichik harfga o‘zgartiradi. Metod parametr qabul qilmaydi. casefold metodi lower metodidan farqli o‘laroq faqat Unicode belgilar bilan ishlaydi.

**Sintaksis:**

satr.casefold()

**Misol 1:**

txt = 'AbcDef'

**print**(txt.**casefold**())

**Natija:**

*'abcdef'*

**Misol 2:**

txt = 'AbС DeА'

**print**(txt.**casefold**())

**Natija:**

*'abс deа'*

**title metodi**

title metodi satrdagi har bir so‘zning birinchi belgisini katta harfga aylantiradi, qolgan barcha belgilarni kichik harfga o‘zgartiradi.

**Sintaksis:**

satr.title()

**Misol 1:**

txt = 'abcde'

**print**(txt.**title**())

**Natija:**

*'Abcde'*

**Misol 2:**

txt = 'abc def'

**print**(txt.**title**())

**print**(txt.**capitalize**())

**Natija:**

*'Abc Def'*'Abc def'

**Python’da satrlarni bo‘lish (parsing) yoki bo‘laklarga ajratish metodlari haqida.**

**split metodi**

split metodi satrni belgilangan ajratuvchi bo‘yicha bo‘lib, natijada ro‘yxat (list) qaytaradi. Ikkinchi ixtiyoriy parametr sifatida nechta bo‘lish amalga oshirilishini ko‘rsatish mumkin. Standart bo‘yicha satr cheksiz bo‘linadi.

**Sintaksis:**

satr.split(ajratuvchi, [bo‘lish\_soni])

**Misol 1:**  
Satrni bir marta bo‘lish:

txt = 'ab\_ac\_dea'

**print**(txt.**split**('\_', 1))

**Natija:**

['ab', 'ac\_dea']

**Misol 2:**  
Bo‘lish sonini ko‘rsatmasdan:

txt = 'ab\_ac\_dea'

**print**(txt.**split**('\_'))

**Natija:**

['ab', 'ac', 'dea']

**rsplit metodi**

rsplit metodi satrni oxiridan boshlab belgilangan ajratuvchi bo‘yicha bo‘lib, natijada ro‘yxat (list) qaytaradi. Ikkinchi ixtiyoriy parametr sifatida nechta bo‘lish amalga oshirilishini ko‘rsatish mumkin. Standart bo‘yicha satr cheksiz bo‘linadi.

**Sintaksis:**

satir.rsplit(ajratuvchi, [bo‘lish\_soni])

**Misol 1:**

Satrni oxirgi uchrashuv bo‘yicha bir marta bo‘lish:

txt = 'ab\_ac\_dea'

**print**(txt.**rsplit**('\_', 1))

**Natija:**

['ab\_ac', 'dea']

**Misol 2:**  
Bo‘lish sonini ko‘rsatmasdan:

txt = 'ab\_ac\_dea'

**print**(txt.**rsplit**('\_'))

**Natija:**

['ab', 'ac', 'dea']

**partition metodi**

partition metodi satrni belgilangan ajratuvchi bo‘yicha birinchi uchrashuvdan bo‘lib, natijada 3 elementli kортеж qaytaradi: (bo‘lingan qism oldi, ajratuvchi, bo‘lingan qism orqa).

**Sintaksis:**

satr.partition(ajratuvchi)

**Misol 1:**

txt = 'abc\_dea'

**print**(txt.**partition**('\_'))

**Natija:**

('abc', '\_', 'dea')

**Misol 2:**

Satrda bir nechta ajratuvchi bo‘lsa ham faqat birinchi bo‘lish amalga oshadi:

txt = 'ab\_cd\_ea'

**print**(txt.**partition**('\_'))

**Natija:**

('ab', '\_', 'cd\_ea')

**rpartition metodi**

rpartition metodi satrni belgilangan ajratuvchi bo‘yicha oxirgi uchrashuvdan bo‘lib, natijada 3 elementli kортеж qaytaradi: (bo‘lingan qism oldi, ajratuvchi, bo‘lingan qism orqa).

**Sintaksis:**

satr.rpartition(ajratuvchi)

**Misol 1:**

txt = 'abc\_dea'

**print**(txt.**rpartition**('\_'))

**Natija:**

('abc', '\_', 'dea')

**Misol 2:**

Satrda bir nechta ajratuvchi bo‘lsa, bo‘lish oxirgi uchrashuv bo‘yicha amalga oshadi:

txt = 'ab\_cd\_ea'

**print**(txt.**rpartition**('\_'))

**Natija:**

('ab\_cd', '\_', 'ea')

**join metodi**

join metodi ro‘yxat elementlarini satrga birlashtiradi va ularni belgilangan ajratuvchi bilan ajratadi (bo‘shliq, vergul va h.k.).

**Sintaksis:**

'ajratuvchi'.**join**(royxat\_satrlar)

**Misol 1:**

Elementlarni bo‘shliq bilan birlashtirish:

lst = ['ab', 'cd', 'ef']

txt = ' '.**join**(lst)

**print**(txt)

**Natija:**

*'ab cd ef'*

**Misol 2:**

Elementlarni vergul bilan birlashtirish:

lst = ['ab', 'cd', 'ef']

txt = ','.**join**(lst)

**print**(txt)

**Natija:**

*'ab,cd,ef'*

**Python’da satrlarni formatlash**

**strip metodi**

strip metodi satrning boshidan va oxiridan berilgan belgilarni olib tashlab, natijaviy satrni qaytaradi. Metodning ixtiyoriy parametrida qaysi belgilarni o‘chirmoqchi ekanligimizni ko‘rsatamiz. Agar parametr ko‘rsatilmasa, metod faqat boshidagi va oxiridagi bo‘sh joylarni o‘chiradi.

Sintaksis:

satr.strip([o‘chiriladigan belgilar])

Misol 1:

txt = 'abcdea'

**print**(txt.**strip**('a'))

Natija:

'bcde'

Misol 2:

txt = ' abcdea '

**print**(txt.**strip**('a'))

Natija:

' abcdea '

Izoh: Endi satrning birinchi va oxirgi belgisi bo‘sh joy bo‘lgani uchun va berilgan 'a' belgisi mavjud emasligi sababli, metod satrni o‘zgartirmadi.

Misol 3:

txt = ' abcdea '

**print**(txt.**strip**())

Natija:

'abcdea'

**lstrip metodi**

lstrip metodi satrning boshidan berilgan belgilarni olib tashlab, natijaviy satrni qaytaradi. Metodning ixtiyoriy parametrida qaysi belgilarni o‘chirmoqchi ekanligimizni ko‘rsatamiz. Agar parametr ko‘rsatilmasa, metod faqat boshidagi bo‘sh joylarni o‘chiradi.

Sintaksis:

satr.lstrip([o‘chiriladigan belgilar])

Misol 1:

txt = 'abcdea'

**print**(txt.**lstrip**('a'))

Natija:

'bcdea'

Misol 2:

txt = ' abcdea '

**print**(txt.**lstrip**('a'))

Natija:

' abcdea '

Izoh: Endi satrning birinchi belgisi bo‘sh joy bo‘lgani uchun va berilgan 'a' belgisi mavjud emasligi sababli, metod satrni o‘zgartirmadi.

Misol 3:

txt = ' abcdea '

**print**(txt.**lstrip**())

Natija:

'abcdea '

**rstrip metodi**

rstrip metodi satrning oxiridan berilgan belgilarni olib tashlab, natijaviy satrni qaytaradi. Metodning ixtiyoriy parametrida qaysi belgilarni o‘chirmoqchi ekanligimizni ko‘rsatamiz. Agar parametr ko‘rsatilmasa, metod faqat oxiridagi bo‘sh joylarni o‘chiradi.

Sintaksis:

satr.rstrip([o‘chiriladigan belgilar])

Misol 1:

txt = 'abcdea'

**print**(txt.**rstrip**('a'))

Natija:

'abcde'

Misol 2:

txt = ' abcdea '

**print**(txt.**rstrip**('a'))

Natija:

' abcdea '

Izoh: Endi satrning oxirgi belgisi bo‘sh joy bo‘lgani uchun va berilgan 'a' belgisi mavjud emasligi sababli, metod satrni o‘zgartirmadi.

Misol 3:

txt = ' abcdea '

**print**(txt.**rstrip**())

Natija:

' abcdea'

**format metodi**

format metodi satr ichidagi bo‘sh qavslarga kerakli qiymatlarni joylashtirish uchun ishlatiladi. Metodning parametrida qavslarga qo‘yiladigan qiymatni ko‘rsatamiz.

Sintaksis:

satr.format(qiymat)

Misol 1:

txt = 'This is {}'

**print**(txt.**format**('text'))

Natija:

'This is text'

Misol 2:

txt = 'This is {}'

**print**(txt.**format**(123))

Natija:

'This is 123'

Misol 3:

txt = 'This is {}'

lst = ['a', 'b', 'c']

**print**(txt.**format**(lst[0]))

Natija:

'This is a'

**zfill metodi**

zfill metodi satrni berilgan uzunlikka yetguncha oldidan nol bilan to‘ldiradi. Satrning kerakli uzunligi metod parametri orqali beriladi.

Sintaksis:

satr.zfill(uzunlik)

Misol 1:

txt = 'abc'

**print**(txt.**zfill**(6))

Natija:

'000abc'

Misol 2:

txt = 'abc'

**print**(txt.**zfill**(2))

Natija:

'abc'

**ljust metodi**

ljust metodi satrni chapga tekislab, qolgan joyni to‘ldiradi. Birinchi parametr sifatida satr uzunligini, ikkinchi ixtiyoriy parametr sifatida esa to‘ldiruvchi belgini berish mumkin (agar belgini ko‘rsatmasak, sukut bo‘yicha bo‘sh joy ishlatiladi).

Sintaksis:

satr.ljust(uzunlik, [to‘ldiruvchi belgi])

Misol 1:

txt = 'abc'

**print**(txt.**ljust**(6, '!'))

Natija:

'abc!!!'

Misol 2:

txt = 'abc'

**print**(txt.**ljust**(6))

Natija:

'abc '

**rjust metodi**

rjust metodi satrni o‘ngga tekislab, qolgan joyni to‘ldiradi. Birinchi parametr sifatida satr uzunligini, ikkinchi ixtiyoriy parametr sifatida esa to‘ldiruvchi belgini berish mumkin (agar belgini ko‘rsatmasak, sukut bo‘yicha bo‘sh joy ishlatiladi).

Sintaksis:

satr.rjust(uzunlik, [to‘ldiruvchi belgi])

Misol 1:

txt = 'abc'

**print**(txt.**rjust**(6, '!'))

Natija:

'!!!abc'

Misol 2:

txt = 'abc'

**print**(txt.**rjust**(6))

Natija:

' abc'

**Python’da satrlarda qidiruv**

**startswith metodi**

startswith metodi satrning ko‘rsatilgan kichik satr bilan boshlanishini tekshiradi. Natija sifatida **True** yoki **False** qaytaradi.

Birinchi parametr sifatida kerakli kichik satrni beramiz, ikkinchi va uchinchi ixtiyoriy parametrlar qidiruvning boshlanish va tugash indekslarini belgilaydi.

**Sintaksis**

satr.startswith(kichik\_satr, [qidiruv\_boshlanish\_indeksi], [qidiruv\_tugash\_indeksi])

**Misol**

Satr 'a' bilan boshlanishini tekshiramiz:

txt = 'abcadea'

**print**(txt.**startswith**('a', 3, 6))

Natija:

*True*

Indeks chegaralarini belgilash bilan tekshirish:

txt = 'abcadea'

**print**(txt.**startswith**('a', 3, 6))

Natija:

*True*

**endswith metodi**

endswith metodi satrning ko‘rsatilgan kichik satr bilan tugashini tekshiradi va natija sifatida **True** yoki **False** qaytaradi.

Birinchi parametr sifatida kerakli kichik satrni beramiz, ikkinchi va uchinchi ixtiyoriy parametrlar qidiruvning boshlanish va tugash indekslarini belgilaydi.

**Sintaksis**

satr.endswith(kichik\_satr, [qidiruv\_boshlanish\_indeksi], [qidiruv\_tugash\_indeksi])

**Misollar**

Kichik satr 'a' ni topamiz, qidiruv boshlanish va tugash indekslarini ko‘rsatib:

txt = 'abcadea'

**print**(txt.**endswith**('a', 0, 4))

Natija:

*True*

Indekslarni o‘zgartirib:

txt = 'abcadea'

**print**(txt.**endswith**('a', 0, 3))

Natija:

*False*

Satr oxirida 'a' bor-yo‘qligini tekshirish:

txt = 'abcadea'

**print**(txt.**endswith**('a'))

Natija:

*True*

**find metodi**

find metodi satrda berilgan kichik satrning birinchi mos kelishining indeksini qaytaradi.

* Birinchi parametr sifatida kerakli kichik satrni beramiz.
* Ikkinchi va uchinchi ixtiyoriy parametrlar qidiruvning boshlanish va tugash indekslarini belgilaydi.
* Agar kichik satr topilmasa, metod **-1** qaytaradi.

**Sintaksis**

satr.find(kichik\_satr, [qidiruv\_boshlanish\_indeksi], [qidiruv\_tugash\_indeksi])

**Misollar**

Kichik satr 'a' ni topamiz, qidiruv boshlanish va tugash indekslarini ko‘rsatib:

txt = 'abcadea'

**print**(txt.**find**('a', 1, 4))

Natija:

3

Indekslarni o‘zgartirib:

txt = 'abcadea'

**print**(txt.**find**('a', 1, 3))

Natija:

-1

Satrda kichik satrni indekslarsiz qidirish:

txt = 'abcadea'

**print**(txt.**find**('a'))

Natija:

0

**index metodi**

index metodi satrda berilgan kichik satrning birinchi mos kelishining indeksini qaytaradi.

* Birinchi parametr sifatida kerakli kichik satrni beramiz.
* Ikkinchi va uchinchi ixtiyoriy parametrlar qidiruvning boshlanish va tugash indekslarini belgilaydi.
* Agar kichik satr topilmasa, metod **xato (exception)** chiqaradi.

**Sintaksis**

satr.index(kichik\_satr, [qidiruv\_boshlanish\_indeksi], [qidiruv\_tugash\_indeksi])

**Misollar**

Kichik satr 'a' ni indekslarsiz topish:

txt = 'abcadea'

**print**(txt.**index**('a'))

Natija:

0

Qidiruv boshlanish va tugash indekslarini ko‘rsatib:

txt = 'abcadea'

**print**(txt.**index**('a', 1, 4))

Natija:

3

**rfind metodi**

rfind metodi satrning oxiridan boshlab berilgan kichik satrning indeksini qaytaradi.

* Birinchi parametr sifatida topmoqchi bo‘lgan satr yoki kichik satrni beramiz.
* Ikkinchi va uchinchi ixtiyoriy parametrlar qidiruvning boshlanish va tugash indekslarini belgilaydi.
* Agar kichik satr topilmasa, metod **-1** qaytaradi.

**Sintaksis**

satr.rfind(nimani\_topish, [qidiruv\_boshlanish\_indeksi], [qidiruv\_tugash\_indeksi])

**Misollar**

Satr oxiridan 'a' ni topish:

txt = 'abacdea'

**print**(txt.**rfind**('a'))

Natija:

6

Qidiruv indekslarini ko‘rsatib:

txt = 'abacdea'

**print**(txt.**rfind**('a', 1, 3))

Natija:

2

Topilmagan kichik satr:

txt = 'abacdea'

**print**(txt.**rfind**('f'))

Natija:

-1

**rindex metodi**

rindex metodi satr oxiridan boshlab berilgan kichik satrning eng katta indeksini qaytaradi.

* Birinchi parametr sifatida topmoqchi bo‘lgan satr yoki kichik satrni beramiz.
* Ikkinchi va uchinchi ixtiyoriy parametrlar qidiruvning boshlanish va tugash indekslarini belgilaydi.
* **rfind** metodidan farqli o‘laroq, agar kichik satr topilmasa, rindex **ValueError** xatosini chiqaradi.

**Sintaksis**

satr.rindex(nimani\_topish, [qidiruv\_boshlanish\_indeksi], [qidiruv\_tugash\_indeksi])

**Misollar**

Satr oxiridan 'a' ni topish:

txt = 'abacdea'

**print**(txt.**rindex**('a'))

Natija:

6

Qidiruv indekslarini ko‘rsatib:

txt = 'abacdea'

**print**(txt.**rindex**('a', 1, 3))

Natija:

2

Topilmagan kichik satr:

txt = 'abacdea'

**print**(txt.**rindex**('f'))

Natija:

ValueError: substring *not* found

**count metodi**  
count metodi berilgan satrda kichik satrning necha marta uchrashini qaytaradi.

* Birinchi parametr sifatida topmoqchi bo‘lgan kichik satrni beramiz.
* Ikkinchi va uchinchi ixtiyoriy parametrlar qidiruvning boshlanish va tugash indekslarini belgilaydi.

**Sintaksis**

satr.count(kichik\_satr, [qidiruv\_boshlanish\_indeksi], [qidiruv\_tugash\_indeksi])

**Misollar**

Qidiruvning boshlanish va tugash indekslarini ko‘rsatib:

txt = 'abcadea'

**print**(txt.**count**('a', 0, 2))

Natija:

1

Satr bo‘ylab barcha uchrashlarni hisoblash:

txt = 'abсadea'

**print**(txt.**count**('a'))

Natija:

3

**replace metodi**

replace metodi satrda qidirish va almashtirish ishlarini bajaradi.

* Birinchi parametr sifatida almashtirmoqchi bo‘lgan kichik satrni beramiz.
* Ikkinchi parametr sifatida uning o‘rniga qo‘yiladigan satrni beramiz.
* Uchinchi ixtiyoriy parametr bilan almashtirishlar sonini belgilash mumkin.

**Sintaksis**

satr.replace(oladigan, qo‘yiladigan, [almashtirish\_soni])

**Misollar**

Barcha 'a' harflarini '!' bilan almashtirish:

txt = 'abacdea'

**print**(txt.**replace**('a', '!'))

Natija:

'!b!cde!'

Almashtirishlar sonini belgilash:

txt = 'abacdea'

**print**(txt.**replace**('a', '!', 2))

Natija:

'!b!cdea'

**Python’da satrni tekshirish**

**istitle metodi**

istitle metodi har bir so‘zning birinchi belgisi bosh harf ekanligini tekshiradi. Metodga parametr berilmaydi. Metod natijada True yoki False qiymatlarini qaytaradi.

**Sintaksis**

satr.istitle()

**Misol**

Har bir so‘z bosh harf bilan boshlanishini tekshiramiz:

txt = 'Abc Def'

**print**(txt.**istitle**())

Natija:

*True*

Endi ikkinchi so‘zning birinchi belgisini kichik harfga o‘zgartiramiz va yana tekshiramiz:

txt = 'Abc def'

**print**(txt.**istitle**())

Natija:

*False*

Birinchi so‘zga raqam qo‘shib yana tekshiramiz:

txt = '12abc Def'

**print**(txt.**istitle**())

Natija:

*False*

**isupper metodi**

isupper metodi satrdagi barcha belgilar bosh harf ekanligini tekshiradi. Metodga parametr berilmaydi. Natija True yoki False bo‘ladi.

**Sintaksis**

satr.isupper()

**Misol**

Satrdagi barcha belgilar bosh harf ekanligini tekshiramiz:

txt = 'ABCDEF'

**print**(txt.**isupper**())

Natija:

*True*

Endi satrda aralash belgilar bo‘lsa:

txt = 'AbcDef'

**print**(txt.**isupper**())

Natija:

*False*

**islower metodi**

islower metodi satrdagi barcha belgilar kichik harf ekanligini tekshiradi. Metodga parametr berilmaydi. Natija True yoki False bo‘ladi.

**Sintaksis**

satr.islower()

**Misol**

Satrdagi barcha belgilar kichik harf ekanligini tekshiramiz:

txt = 'abcdef'

**print**(txt.**islower**())

Natija:

*True*

Endi satrda aralash belgilar bo‘lsa:

txt = 'AbcDef'

**print**(txt.**islower**())

Natija:

*False*

**isalpha metodi**

isalpha metodi satr faqat harflardan iboratligini tekshiradi. Metodga parametr berilmaydi. Natija True yoki False bo‘ladi.

**Sintaksis**

satr.isalpha()

**Misol**

Satr faqat harflardan iboratligini tekshiramiz:

txt = 'abcade'

**print**(txt.**isalpha**())

Natija:

*True*

Endi satrda boshqa belgilar bo‘lsa:

txt = 'abcade12'

**print**(txt.**isalpha**())

Natija:

*False*

**isalnum metodi**

isalnum metodi satrdagi belgilar faqat harflar va raqamlardan iboratligini tekshiradi. Metodga parametr berilmaydi. Natija True yoki False bo‘ladi.

**Sintaksis**

satr.isalnum()

**Misol**

Satr faqat harflar va raqamlardan iboratligini tekshiramiz:

txt = 'abcadea12'

**print**(txt.**isalnum**())

Natija:

*True*

Endi satrda boshqa belgilar bo‘lsa:

txt = '@abcadea12'

**print**(txt.**isalnum**())

Natija:

*False*

**isdigit metodi**

isdigit metodi satr faqat raqamlardan iboratligini tekshiradi. Metodga parametr berilmaydi. Natija True yoki False bo‘ladi.

**Sintaksis**

satr.isdigit()

**Misol**

Satr faqat raqamlardan iboratligini tekshiramiz:

txt = '12345'

**print**(txt.**isdigit**())

Natija:

*True*

Endi satrda boshqa belgilar bo‘lsa:

txt = '12345ab'

**print**(txt.**isdigit**())

Natija:

*False*

**isnumeric metodi**

isnumeric metodi satr faqat raqamlardan iboratligini tekshiradi. isdigit metodidan farqli o‘laroq, isnumeric barcha turdagi raqamli qiymatlarni, jumladan, rim raqamlari va kasrlarni ham tekshiradi.

Metodga parametr berilmaydi. Natija True yoki False bo‘ladi.

**Sintaksis**

satr.isnumeric()

**Misollar**

Satr faqat raqamlardan iboratligini tekshiramiz:

txt = '12345'

**print**(txt.**isnumeric**())

Natija:

*True*

Satrda boshqa belgilar bo‘lsa:

txt = '12345ab'

**print**(txt.**isnumeric**())

Natija:

*False*

Satr rim raqamlarini o‘z ichiga olganda:

txt = 'Ⅻ'

**print**('isdigit:', txt.**isdigit**())

**print**('isnumeric:', txt.**isnumeric**())

Natija:

isdigit: *False*isnumeric: *True*

Satr kasr raqamni o‘z ichiga olganda:

txt = '⅓'

**print**('isdigit:', txt.**isdigit**())

**print**('isnumeric:', txt.**isnumeric**())

Natija:

isdigit: *False*isnumeric: *True*

**isspace metodi**

isspace metodi satr faqat bo‘shliq belgilaridan iboratligini tekshiradi. Bo‘shliqlardan tashqari, metod quyidagi belgilarni ham tan oladi: '\t' (gorizontal bo‘shliq), '\n' (yangi qator), '\v' (vertikal bo‘shliq), '\f' (keyingi sahifaga o‘tish), '\r' (satr boshiga qaytish).

Metodga parametr berilmaydi. Natija True yoki False bo‘ladi.

**Sintaksis**

satr.isspace()

**Misollar**

Satr faqat bo‘shliq belgilaridan iboratligini tekshirish:

txt = ' '

**print**(txt.**isspace**())

Natija:

*True*

Satrda boshqa belgilar bo‘lsa:

txt = 'abc de'

**print**(txt.**isspace**())

Natija:

*False*

Satrda '\n' belgisi bo‘lsa:

txt = '\n'

**print**(txt.**isspace**())

Natija:

*True*