**Data types**

Base16

Base8 nubers in python

**Int –** natural numbers (natural sonlar)

**Float –** butun sonlar

**Complex – complex numbers** 1 + 2j

CTRL + Enter to run the code

SHIFT + Enter to run and open new window

-- Variables should start with a letter or the underscore

-- Variable lar letter bilan yoki unserscore bilan boshlanishi kerak.

**Getting user input**

Masalan: c =float(input('Enter First Number'))

**Print ni ichida ham string ham int ni ishlatish:**

print("Figure is " + str(int(p)))

print("Figure is" , int(p) )

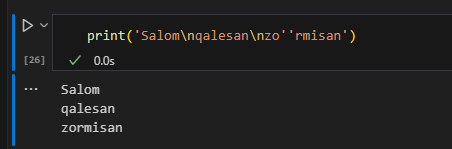
**Escape character**

print("What's up \"buddy\"  ")

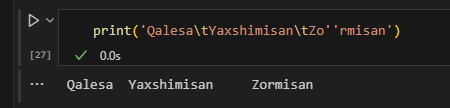
**Connections:**

**\n** ni ishlatish **enter** commandini berish degandek gap, printni ichida kelsa ishlaydi, variable bilan chaqirsa ishlamaydi

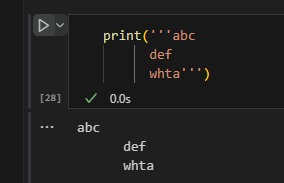
print ('It is \'\nt')



**\t** ni ishlatish tab ni bosgqandek bo’ladi. printni ichida kelsa ishlaydi, variable bilan chaqirsa ishlamaydi



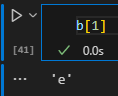
Multiline connect:



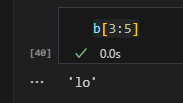
Indexing va slicing

Masalan shu variable bor:

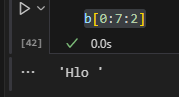
b[1]   #bu qavslar bilan stringni ichidagi harflardan nechinchisini chiqishini bersak bo'ladi, 0 dan boshlanadi sanoq



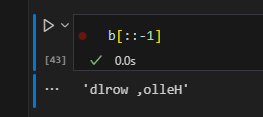
b[3:5] bu esa 3 chi characterdan boshlab, 5 chigacha chiqaradi.



b[0:7:2] #bu esa 0 chi characterdan boshlab(ya’ni 1), 7chi charactergacha olib, 2 ta dan tashab tashab oladi.



-da berilsa oxirgisi teskarisiga oladi. harflarni



**Index**

q = 'sardorjonumidvich@gmail.com'

position = q.index("@")+1

q[position::]

index bilan string dagi ma’lum bir characterni qidirib topsa bo’ladi.

Buni uchun qaysi variabledagi indexligini aniqlash uchun *variable.index* ni ishlatamiz, keyin qavs ichida qaysi stringligi. +1 qo’yilganligi sababi @ bilan qidirilgan va o’sha @ ni olib tashlash uchun

**Dir()** bilan fuctionlarni ko’rsa bo’ladi.

w.capitalize() birinchi kelgan harfni katta qilib beradi.

w.title() bu ham birinchi kelgan harfni katat qiladi lkn ikkita alohida so’zlar bo’lsa ikkovini ham alohida alohida qiladi.M: ‘Hello, World’

w.upper() upper esa hammasini katta qiladi.M: ‘HELLO, WORLD’

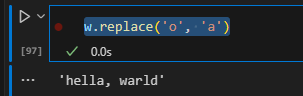
w.lower() kichkina qiladi

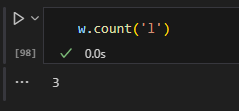
w.swapcase() kattalarni kichik qiladi, kichiklarni katta.

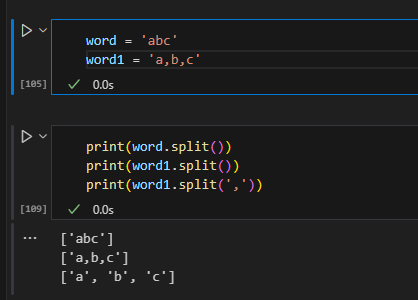
w.index('l') topib beradi berilgan harfni qaysi joyda turganini

w.find('dwl') bu ham xuddi index dek faqat berilmagan harflarni kirgizganda, shunchaki -1 chiqarib beradi.

w.replace('o', 'a') bu o harflarni topib, a ga o’zgartiradi.



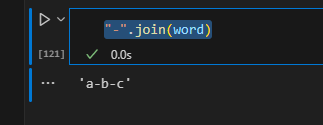
Count ni ichida biror characterni beramiz, u esa o’sha nechtaligini topib beradi.



Demak, split characterlarni ajratib beruvchi, nima bilan ajratish kerakligini kiritib ketish kerak lekin.

g.strip() strip bo’sh qolgan joylarni o’chirish uchun kerak. (shuningdek lstrip-chap tarafdagi bo’sh joylarni o’chiradi, rstript o’ng tarafdagini.)

"-".join(word) join bilan qaysi belgi bilan ajratish kerakligini ko’rsatib, shunday result olishimiz mumkin:

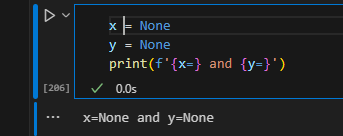


import random

random.randint(1,6) bu function ni random library dan import qilib yozamiz. Bu yerda ichiga kirgizilgan sonlar ortasidagilaridan random chiqarib beradi. (1 dan 6 gacha)

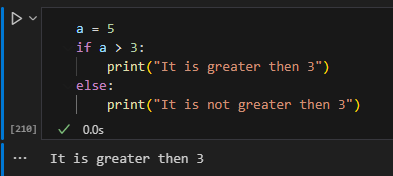


Print ni ichida (stringni ichida, yonida ) variable ni keltirishni ikki xil usuli

Shuningdek, {} ni ichida x= bilan yozib ketsak, python tushunib result ni x = variable qilib chiqarib beradi.

**If else**

Bool() bo’m bosh yoki 0 dan boshqalari true chiqaradi, bular esa false

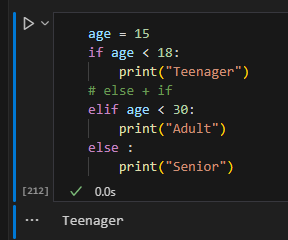


**If** *condition* :

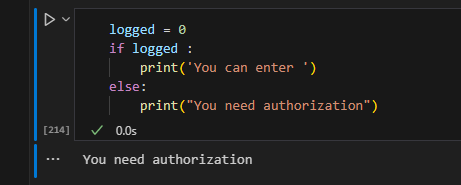
True\_condition\_result

**Else**:

False\_condition\_result



1 dan ortiq if ni ishlatmoqchi bo’lsak, else and if combination ni ishlatamiz; **ELIF**

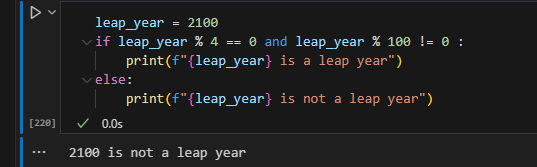
Agarda if ga condition bermagan holatda u auto true or false ligini tekshiradi: (varable yoki so’zni):Masalan;  


% - bo’lib qoldigini chiqarish; (5 % 2) result >> 1

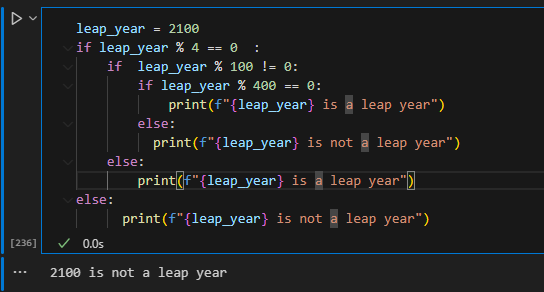
// - bo’lib butun qismini chiqarish; (5 // 2) result >> 2

/ - oddiy bo’lish; (5 / 2) result >> 2.5

Conditionlar ni ko’paytirishni ikki xil usuli :



Faqat iflik varianti:



* **List** is a collection which is ordered and changeable. Allows duplicate members.
* [**Tuple**](https://www.w3schools.com/python/python_tuples.asp) is a collection which is ordered and unchangeable. Allows duplicate members.
* [**Set**](https://www.w3schools.com/python/python_sets.asp) is a collection which is unordered, unchangeable\*, and unindexed. No duplicate members.
* [**Dictionary**](https://www.w3schools.com/python/python_dictionaries.asp) is a collection which is ordered\*\* and changeable. No duplicate members.
* \*Set *items* are unchangeable, but you can remove and/or add items whenever you like.
* \*\*As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

**List nima?**

List bu [] ni ichidagi itemlar hisoblanadi.

List stringlarni collection ilib saqlyadi va uni o’zgartirsa bo’ladi. Bitta variable ni ichida bir nechta itemlarni saqlash uchun ishlatilinadi.

List items are ordered, changeable, and allow duplicate values.

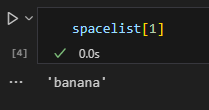
The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

**List ni ichidagi itemlarni sanash** uchun**:**spacelist = ['apple', 'banana', 'orange','limon','tomato','potato','catton']

print(len(spacelist))

**Access list items:**

Demak list items ga access qilish uchun [] dan foydalanamiz:

* da bersak oxiridan sanab oladi listdan.

**Range da olish uchun [a:b]** beriladi. Example: a is included and b not included

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[2:5]) result >>>> "cherry", "orange", "kiwi"

#Remember that the first item is position 0,

#and note that the item in position 5 is NOT included

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[-4:-1]) >>>> result : orange , kiwi , melon

#Negative indexing means starting from the end of the list.

#This example returns the items from index -4 (included) to index -1 (excluded)

#Remember that the last item has the index -1,

**Checking whether is in lists or not:**

#check  with if in lists

if 'apple' in spacelist:

    print('apple in the list')

Change Item Value

thislist = ["apple", "banana", "cherry"]  
thislist[1] = "blackcurrant"  
print(thislist) result >>> ["blackcurrant ", "banana", "cherry"

Range dagi bir nechta valuelarni o’zgartirish:

spacelist[3:5] = 'Limoncha', 'Pomildori'

spacelist

**Agarda range da kamroq value berib lekin koproq insert qilsak ham insert bo’laveradi lekin length o’zgarib kattarib ketadi.**

Insert qilish:

spacelist.insert(3,'phone')

spacelist

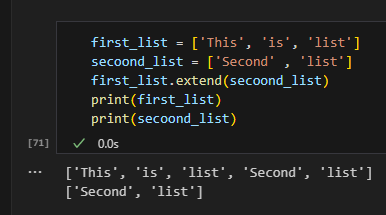
result: >>>> ['apple', 'banana', 'apelsin', 'phone', 'Limoncha', 'Pomildori', 'potato', 'catton']

**Append** faqatgina stringni davomidan qo’sha oladi,xohlagan joyimizga qosholmaydi shuni uchun shu yerda, **insert** ni ishlatamiz: Masalan: birinchi index number beriladi, keyin nimani qoshish;

a.insert(5,'w')

a

Ikkita list ni qo’shish uchun **extend** dan foydalaniladi. Qaysiga qo’shmoqchi bo’lsak birinchi yozib keyin qavs ichiga kelyotgan list ni yozamiz. Shuningdek, extend bilan lyuboy iterable objectlarni ishlatsa bo’ladi, (lists ,tuples, dictionaries,sets )



**Farqi: Append bitta elementniorqadan qo’shish uchun**

**Insert aniq bir positionga qoshish uchun.**

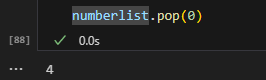
**Extend esa ikkita list ni bir biriga qo’shish uchun ishlatilinadi.**

spacelist.extend(mytuple)

spacelist

numberlist.pop() bu esa ochirib tashlaydi, listdagi itemlarni. Default holatda: eng oxirgisini.

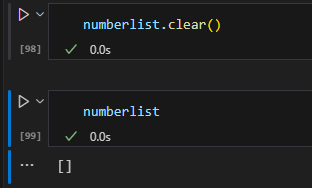
Yoki ichiga index raqam berish kerak.



numberlist.remove(2) remove esa specific value bo’yicha o’chiradi, () ichiga nmani o’chirmoachi bo’lsak beramiz.

spacelist.remove('phone')

spacelist



Numberlist.clear esa ichidagi hamma itemlarni o’chirib tashlaydi

Shuningdek del keyword orqalo ham ochirsa boladi. Accessing bilan inde beriladi, agar berilmasa butun list ni o’chirib yuboradi.

thislist = ["apple", "banana", "cherry"]  
del thislist[0]  
print(thislist) >>> result: "banana", "cherry"

thislist = ["apple", "banana", "cherry"]  
del thislist  
print(thislist) >>> result: []

**Difference :** *Remove* da aniq nimani o’chirmoqchi bo’lsak o’zini yozamiz,Agarda specified value 2 ta bo’lsa birinchisini o’xhiradi

Pop da esa index ko’rsatiladi, Ko’rsatilmasa oxirgisini o’chiradi.

clear esa hammasini o’chiradi.

thislist = ["apple", "banana", "cherry"]  
for x in thislist:  
  print(x) **result:** >>> apple , banana, cherry

*A short hand for loop that will print all items in a list:*

thislist = ["apple", "banana", "cherry"]  
[print(x) for x in thislist] **result:** >>> apple , banana, cherry

**List comprehension**

newspace = [i for i in spacelist if i != 'apelsin']

newspace

buni resulti apelsindan tashqari hamma itemlarni yangi listga olib otadi.

**Shuningdek** yangi listdagi itemlarni ozgartirib otkazishimiz ham mumkin:

newspace = [i.upper() for i in spacelist]

newspace

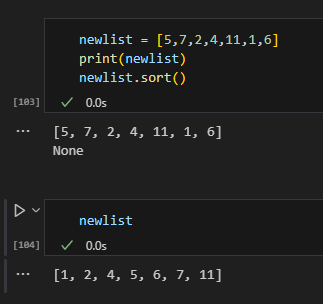
bu holatda hammasi uppercase bo’lib o’tadi.

**Condition berib otkazish:**

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x if x != "banana" else "orange" for x in fruits]

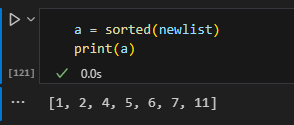
print(newlist) result: >>>> "apple", " orange ", "cherry", "kiwi", "mango"

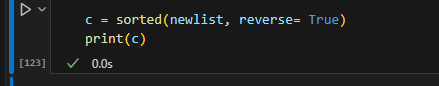


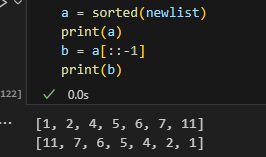
Sort ni ishlatish:

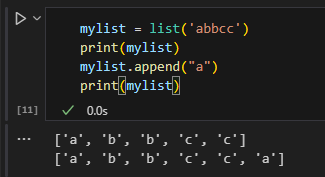
Sort case sensetive hisoblanadi, shuni uchun birinchi kattalarini sort qilib chiqarib, keyin kichiklarini sort qiladi.

Shuni yodda tutish kerakki, none result ga ega bo’lgan methodlar, print qilganda ham variable ga qo’ysa ham none deb chiquradi, lekin u asosiy variable ga ta’sir qilib o’zini ozgartiradi.  
Agar da sortni variable ga ishlatmoqchi bo’lsak;

 **Sorted** ni ishlatamiz

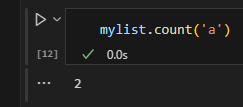
agarda desc kerak bo’lsa buni ishlatsa ham bo’ladi yoki **accessing** ni ham ishlatsa bo’ladi.

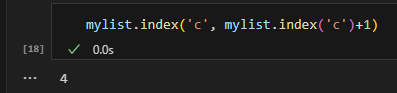


**Mutable and immutable**

Listlar bilan kealdigan functionlar:  
1. Append bu biror character ni qo’shadi

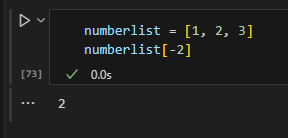
Count ga berilgan characterni sanaydi





Index ni ishlatish

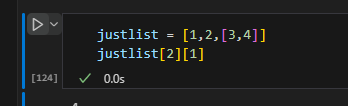
mylist.reverse() bu orqacha qilib yozib beradi, lkn har doim ham emas

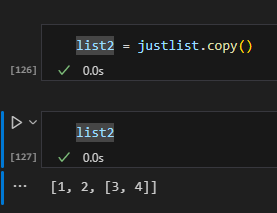
 Accessing

[]

bir xil ishlaydi.

.

 List ichida list kelish holati, bunda accessing element da 2 ni bersak butun listni nazarda tutadi 2 da . keyingi brecket da esa ichidagi listdan belgilaymiz.



Copy esa nusxa oladigan method.

**Slice operator bilan ham copy qilsa bo’ladi.**

thislist = ["apple", "banana", "cherry"]  
mylist = thislist[:]  
print(mylist)

**list() method bilan ham copy qilsa bo’ladi**

thislist = ["apple", "banana", "cherry"]  
mylist = list(thislist)  
print(mylist)

**Shallow copy and deep copy**

Agarda listdagi item ozgaradigan bo’lsa, shallow copy bo’lib ozgaradi ya’ni ikkinchi listdagi narsa ham ozgaradi copy qilib o’tilgan list da ham. Lekin birinchi asosiy listdagi item o’zgartirilsa ocpy dagi listda xech narsa o’zgarmaydi.

**Joins in lists**

Eng oddiy usuli joinni:

superlist = fruits + spacelist

superlist

yoki for ni ishlatish orqali:

for i in fruits:

    spacelist.append(i)

print(spacelist)

yoki extend.

**ALL List Methods**

Python has a set of built-in methods that you can use on lists.

Method Description

**append()** Adds an element at the end of the list

**clear**() Removes all the elements from the list

**copy**() Returns a copy of the list

**count()** Returns the number of elements with the specified value

**extend()** Add the elements of a list (or any iterable), to the end of the current list

**index()** Returns the index of the first element with the specified value

**insert()** Adds an element at the specified position

**pop()** Removes the element at the specified position

**remove()** Removes the item with the specified value

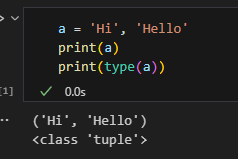
**reverse()** Reverses the order of the list

**sort()** Sorts the list

**Tuples**

Tuple ham 4 ta data collection lar ning biri hisoblanadi. Tuple ni o’zgartirib bo’lmaydi va aniq orderi mavjud. Index ishlata olamiz degani bu. Duplicate lar ni ham qabul qiladi.

Tuple – ni sherigi,



Listdan farqi immutible ligi hisoblanadi. Ya’ni list dagilarni o’zgartirib bo’lsa, tuple dagilarni ozgartirrb bomaydi.  
Bitta value lik tuple: agar vergul bo’lmasa uni string deb eslab qoladi.

thistuple = ("apple",)

print(type(thistuple)) result >:>> class tuple

#NOT a tuple

thistuple = ("apple") result >:>> class str

print(type(thistuple))

**ACCESSING bir xil LIST dagidek.**

## Check if Item Exists

thistuple = ("apple", "banana", "cherry")  
if "apple" in thistuple:  
  print("Yes, 'apple' is in the fruits tuple")

**Tuple ni Update qilish:**

Tuple ni ozini update qilib bo’lmaydi lekin uni listga o’tkazib keyin update qilsak bo’ladi:

clock = ('kary','Camera','Course','Map')

bizda shunaqa tuple bor edi; uni birinchi value sini ozgartirishimiz kerak:

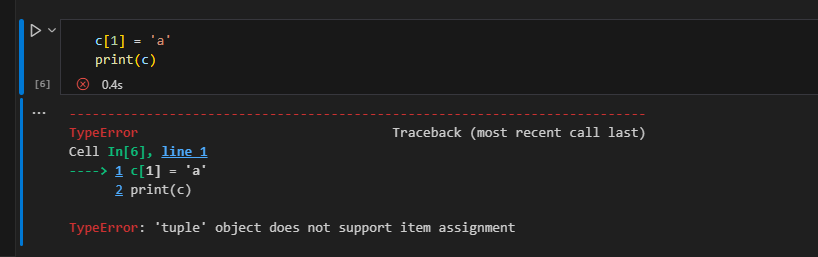
clock1 = list(clock)

clock1[0]= "Kareness"

clock  = tuple(clock1)

print(clock)

buni listga o’tkazib keyin ozgartirib yana tuplega otkazsak boladi.



Agar da **x =** dan keyin yani variable dan keyin togridan togri kelsa value lar ularni tuple deb eslab qoladi.

**Value Add qilish uchunam shunday qilinadi, remove qilish uchun ham**

Adding:

thistuple = ("apple", "banana", "cherry")  
y = list(thistuple)  
y.append("orange")  
thistuple = tuple(y) result >>> ("apple", "banana", "cherry","orange")

Removing:

thistuple = ("apple", "banana", "cherry")  
y = list(thistuple)  
y.remove("apple")  
thistuple = tuple(y) result >>> ("banana", "cherry")

**UNPACKING a Tuple**

Biz pythonda tuple lar ni malum variablelarga tenglashtirib chiqishimiz mumkin bu usulda:

Clock = ('Kareness', 'Camera', 'Course', 'Map', 'Calendar', 'Calendar')

(care, cam, cor, ma, \*cal) = clock

print(care)

print(cam)

print(cor)

print(ma)

print(cal)

result:

Kareness

Camera

Course

Map

['Calendar', 'Calendar']

Bu malum bir variable larga berish hisoblanadi, agarda variable lar kam bo’lsa value lardan, unda \* ni ishlatamiz.

**LOOP lar ham list dagidek**

If you want to multiply the content of a tuple a given number of times, you can use the \* operator:

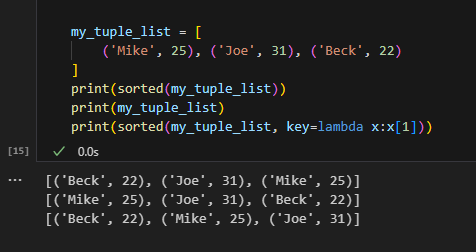
fruits = ("apple", "banana", "cherry")  
mytuple = fruits \* 2  
  
print(mytuple

**ALL METHODS OF TUPLE**

**count()** Returns the number of times a specified value occurs in a tuple

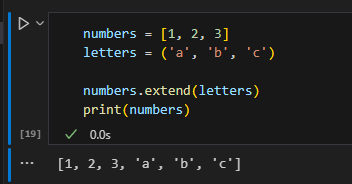
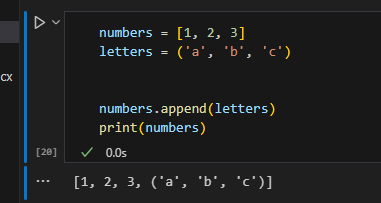
**index()** Searches the tuple for a specified value and returns the position of where it was found

Lambda x[y]:x[y] listdagi qaysi valuedan order qilishni ko’rsatish uchun ishlatsa bo’ladi. Key= ni ichida:

\

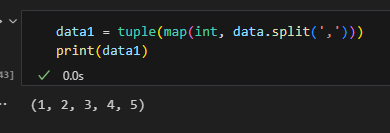
Append bilan list bn tuple ni atdelni atdelni qilib qo’shsa bo’ladi.

Misol:

Cast = map

Biror listdagi valuelarni boshqa value ga o’tkazish:



Dictionary dan key ni qidirish

‘age’ in mich

‘key\_name’ in dictionary

Unique value ni add qilish uchun add ishlatlinadi.

Update 2 tadan ortiqlarini ham qo’sha oladi.

**Sets**

A set is a collection which is unordered, unchangeable\*, and unindexed.

* Set items are unchangeable, but you can remove items and add new items True va 1 same value deb hisoblanadi setsda va duplicate deb 1 ni olib tashlaydi va true ni chiqaradi.
* False ham 0 bilan duplicate bo’ladi.

Set da ham itemlarini sanash uchun len() ishlatilinadi.

Set create qilish uchun **set()** dan foydalaniladi.

thisset = set(("apple", "banana", "cherry")) # note the double round-brackets  
print(thisset)

*Add items to sets*

Value qo’shish uchun setga **add()** dan foydalanamiz.

myset.add('pineapple')

Shunda pineapple item qoshiladi setimizga.

Agarda ikkita setlarni qoshmoqchi bo’lsak **update()** dan foydalanamiz.

yourset = {'phone', 'Notebook', 'Laptop', 'Mouse', 'Motherbord'}

myset.update(yourset)

print(myset)

result: {'Notebook', 'carrot', 'phone', 'orange', 'apple', 'pineapple', 'banana', 'Mouse', 'Laptop', 'Motherbord'}

**Update()** da kelayotgan narsa har doim ham set bolishi shartmas, tuple, list, dictionary larni ham setga qo’shsa bo’ladi.

thisset = {"apple", "banana", "cherry"}  
mylist = ["kiwi", "orange"]  
  
thisset.update(mylist)  
  
print(thisset) result : {"apple", "banana", "cherry", "kiwi", "orange"}

*Delete items from set*

Remove(), discard(), pop(), clear(), del

thisset = {"apple", "banana", "cherry"}  
thisset.remove("banana")  
print(thisset)

thisset = {"apple", "banana", "cherry"}  
thisset.discard("banana")  
print(thisset)

**remove()** va **discard()** ni farqi shundaki agar remove ga berilgan o’chirilishi kerak bo’lgan item setda bo’lmasa *error* qaytaradi, **discard()** esa hech qanday error qaytarmaydi o’sha item bo’lmasa ham.

**Pop()** esa random itemni o’chirib tashlaydi.

**Clear()** burun setni ichidagilarni ochirib tozalab tashlaydi.

**Del** ham tozalab tashlaydi butunlay va ishlatilinishi ham boshqacha.

thisset = {"apple", "banana", "cherry"}  
del thisset  
print(thisset)

*Join sets*

There are several ways to join two or more sets in Python.

The union() and update() methods joins all items from both sets.

The intersection() method keeps ONLY the duplicates.

The difference() method keeps the items from the first set that are not in the other set(s).

The symmetric\_difference() method keeps all items EXCEPT the duplicates

set1 = {"a", "b", "c"}

set2 = {1, 2, 3}

set3 = {"John", "Elena"}

set4 = {"apple", "bananas", "cherry"}

a  = set1.union(set2,set3,set4)

b = set1 | set2 | set3 | set4

c = myset = set1.union(set2, set3, set4)

union va | usullari ikkalasi bir xil.

Union bilan boshqa data typlelarni qo’shishni iloji bor lekin | bilan faqatgina setlarni setlarga qo’shsa bo’ladi.

The update() method inserts all items from one set into another.

The update() changes the original set, and does not return a new set.

The intersection() method will return a new set, that only contains the items that are present in both sets.

set1 = {"apple", "banana", "cherry"}  
set2 = {"google", "microsoft", "apple"}  
set3 = set1.intersection(set2)  
print(set3) result: >>> set3 = {“apple”}

You can use the & operator instead of the intersection() method, and you will get the same result.

set1 = {"apple", "banana", "cherry"}  
set2 = {"google", "microsoft", "apple"}  
set3 = set1 & set2  
print(set3) result: >>> set3 = {“apple”}

Bunda ham & faqatgina set bilan setni join qilganda ishlaydi, intersection() esa hamma typelar bilan join qila oladi.

The intersection\_update() method will also keep ONLY the duplicates, but it will change the original set instead of returning a new set.

set1 = {"apple", "banana", "cherry"}  
set2 = {"google", "microsoft", "apple"}  
set1.intersection\_update(set2)  
print(set1)

Agarda ikkita setda 1 , 0 va True, False bo’lsa ularni ham duplicate deb hisobga oladi.

set1 = {"apple", 1,  "banana", 0, "cherry"}  
set2 = {False, "google", 1, "apple", 2, True}  
set3 = set1.intersection(set2)  
print(set3) result: >>> {“False”, “apple”, “True”}

**Difference()**

Difference esa birinchi berilgan setda bor va ikkinchisida yo’qlarini chiqarib beradi.

The difference() method will return a new set that will contain only the items from the first set that are not present in the other set.

set1 = {"apple", "banana", "cherry"}  
set2 = {"google", "microsoft", "apple"}  
set3 = set1.difference(set2)  
print(set3) result: >>>> {“banana”,”cherry”}

You can use the - operator instead of the difference() method, and you will get the same result.

set1 = {"apple", "banana", "cherry"}  
set2 = {"google", "microsoft", "apple"}  
set3 = set1 - set2  
print(set3)

Bunda ham – operatori faqat setlar bilan ishlaydi qoganlari bilan – qilolmaydi.

The difference\_update() method will also keep the items from the first set that are not in the other set, but it will change the original set instead of returning a new set.

**Symmetric\_difference()**

The symmetric\_difference() method will keep only the elements that are NOT present in both sets.

Symmetric\_difference esa bir birida yogini chiqaradi ya’ni ikkalasida borini olib tashlab qogan hammasini chiqaradi.

set1 = {"apple", "banana", "cherry"}  
set2 = {"google", "microsoft", "apple"}  
set3 = set1.symmetric\_difference(set2)  
print(set3) result: >>> {'google', 'banana', 'microsoft', 'cherry'}

You can use the ^ operator instead of the symmetric\_difference() method, and you will get the same result.

set1 = {"apple", "banana", "cherry"}  
set2 = {"google", "microsoft", "apple"}  
set3 = set1 ^ set2  
print(set3) result: >>> {'google', 'banana', 'microsoft', 'cherry'}

The symmetric\_difference\_update() method will also keep all but the duplicates, but it will change the original set instead of returning a new set.

**ALL METHODS**

|  |  |  |
| --- | --- | --- |
| **Method** | **Shortcut** | **Description** |
| [add()](https://www.w3schools.com/python/ref_set_add.asp) |  | Adds an element to the set |
| [clear()](https://www.w3schools.com/python/ref_set_clear.asp) |  | Removes all the elements from the set |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) |  | Returns a copy of the set |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | [-](https://www.w3schools.com/python/ref_set_difference.asp) | Returns a set containing the difference between two or more sets |
| [difference\_update()](https://www.w3schools.com/python/ref_set_difference_update.asp) | [-=](https://www.w3schools.com/python/ref_set_difference_update.asp) | Removes the items in this set that are also included in another, specified set |
| [discard()](https://www.w3schools.com/python/ref_set_discard.asp) |  | Remove the specified item |
| [intersection()](https://www.w3schools.com/python/ref_set_intersection.asp) | [&](https://www.w3schools.com/python/ref_set_intersection.asp) | Returns a set, that is the intersection of two other sets |
| [intersection\_update()](https://www.w3schools.com/python/ref_set_intersection_update.asp) | [&=](https://www.w3schools.com/python/ref_set_intersection_update.asp) | Removes the items in this set that are not present in other, specified set(s) |
| [isdisjoint()](https://www.w3schools.com/python/ref_set_isdisjoint.asp) |  | Returns whether two sets have a intersection or not |
| [issubset()](https://www.w3schools.com/python/ref_set_issubset.asp) | [<=](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns whether another set contains this set or not |
|  | [<](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns whether all items in this set is present in other, specified set(s) |
| [issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp) | [>=](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns whether this set contains another set or not |
|  | [>](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns whether all items in other, specified set(s) is present in this set |
| [pop()](https://www.w3schools.com/python/ref_set_pop.asp) |  | Removes an element from the set |
| [remove()](https://www.w3schools.com/python/ref_set_remove.asp) |  | Removes the specified element |
| [symmetric\_difference()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | [^](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | Returns a set with the symmetric differences of two sets |
| [symmetric\_difference\_update()](https://www.w3schools.com/python/ref_set_symmetric_difference_update.asp) | [^=](https://www.w3schools.com/python/ref_set_symmetric_difference_update.asp) | Inserts the symmetric differences from this set and another |
| [union()](https://www.w3schools.com/python/ref_set_union.asp) | [|](https://www.w3schools.com/python/ref_set_union.asp) | Return a set containing the union of sets |
| [update()](https://www.w3schools.com/python/ref_set_update.asp) | [|=](https://www.w3schools.com/python/ref_set_update.asp) | Update the set with the union of this set and others |

**Dictionary**

Dictionary lar key: value shaklida itemlar saqlaydi. Ular ordered, changeable va duplicatlarni saqlamaydi.

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
print(thisdict)

result: >>> {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

Shuningdek dictionarylardan ma’lum bir key ni berib valuesini chiqarsak ham boladi.

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
print(thisdict["brand"]) result: >>> ’Ford’

When we say that dictionaries are ordered, it means that the items have a defined order, and that order will not change.

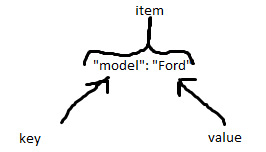
Unordered means that the items do not have a defined order, you cannot refer to an item by using an index.

Lekin bir xil key lik ikkita value bo’lolmaydi ikkinchi kelgan value chiqib qoladi resultda:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964,  
  "year": 2020  
}  
print(thisdict)

result: >>> {'brand': 'Ford', 'model': 'Mustang', 'year': 2020}

Len bilan nechta item borligini sanash mumkin.



It is also possible to use the dict() constructor to make a dictionary.

thisdict = dict(name = "John", age = 36, country = "Norway")

print(thisdict)

result: >>>> {'name': 'John', 'age': 36, 'country': 'Norway'}

**Accessing**

Birinchi usuli nu dict name dan keyin [] lar ichida key ni berish.

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
x = thisdict["model"] result: >>> “Mustang”

Shuningdek **get()** ni ham ishlatsak bo”ladi.

x = thisdict.get("model")

The keys() method will return a list of all the keys in the dictionary.

x = thisdict.keys() result >>> dict\_keys(['brand', 'model', 'year'])

The values() method will return a list of all the values in the dictionary.

x = thisdict.values()

result : >>>

dict\_values(['Ford', 'Mustang', 1964])

The items() method will return each item in a dictionary, as tuples in a list. Bu esa har bitta item ni tuple shaklida olib beradi.

x = thisdict.items()

dict\_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])

**Checking if the key**

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
if "model" in thisdict:  
  print("Yes, 'model' is one of the keys in the thisdict dictionary")

**Change Items**

mydict = {

  "brand": "Ford",

  "model": "Mustang",

  "year": 1964

}

mydict["brand"] = "BMW"

mydict result: {'brand': 'BMW', 'model': 'Mustang', 'year': 1964}

**Update()** bilan ham o’zgartirsa bo’ladi.

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.update({"year": 2020})

**Add item**

Ham xuddi shundan boladi

Yani dict\_name[‘key\_name’] = value

Yoki dictname.update({“key”: value})

The pop() method removes the item with the specified key name:

hisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.pop("model")  
print(thisdict)

The popitem() method removes the last inserted item (in versions before 3.7, a random item is removed instead):

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.popitem()  
print(thisdict)

The del keyword removes the item with the specified key name

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
del thisdict["model"]  
print(thisdict)

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
del thisdict  
print(thisdict) #this will cause an error because "thisdict" no longer exists.

Loop in dict:

Print all key names in the dictionary, one by one

for x in thisdict:  
  print(x)

Print all values in the dictionary, one by one:

for x in thisdict:  
  print(thisdict[x]

You can also use the values() method to return values of a dictionary:

for x in thisdict.values():  
  print(x)

You can use the keys() method to return the keys of a dictionary:

for x in thisdict.keys():  
  print(x)

Loop through both keys and values, by using the items() method:

for x, y in thisdict.items():  
  print(x, y)

Dict methods

clear() Removes all the elements from the dictionary

copy() Returns a copy of the dictionary

fromkeys() Returns a dictionary with the specified keys and value

get() Returns the value of the specified key

items() Returns a list containing a tuple for each key value pair

keys() Returns a list containing the dictionary's keys

pop() Removes the element with the specified key

popitem() Removes the last inserted key-value pair

setdefault() Returns the value of the specified key. If the key does not exist: insert the key, with the specified value

update() Updates the dictionary with the specified key-value pairs

values() Returns a list of all the values in the dictionary

**Functions**

Argument va Parametr :

**Parametr** bu funksiyasi yaratayotganda qavs ichida beriladigan variable hisoblanadi.

**Argument** esa call qilyotganda qavsga yoziladigani.

def funksiya(fname):

    fname = fname.capitalize()

    print(f"Meni ismim {fname}")

funksiya('sardor')

Masalan shunqa:

def myy1(\*meva):

    print(f'Bu meva juda mazali {meva[1]}'  )

myy1('Banan','Olma','Limon')

agar da functonni argumentiga nechta value berilishini bilmasak **args\*** dan foydalanamiz: Lekin bu keyin tuple ga aylanib qoladi.

Agarda \*\*kwargs da foydalansak ham, u ham bir nechta kirgizilsa ham argumentga call qilib beradi va bu dictionary qaytaradi : Misol

def func(\*\*name):

    print('My name is ' + name['lname']  )

func(fname = 'Ismatov', lname = 'Sardor')

DEFAULT value berish:  
def funcon(country = 'Uzbekistan'):

    print('I''m from ' + country)

funcon('Norway')

funcon()

funcon('USA')

Result: >>> Im from Norway Im from Uzbekistan Im from USA

**RETURN**

To let a function return a value, use the return statement:

def calcul(x):

    return x \* 10

print(calcul(2))

print(calcul(3))

**PASS**

function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.

**Positional-Only Arguments**

Bu argumentlarni joylashuvini aniq qiib beruvchi hisoblanadi: / bilan yoziladi, parametrni ichiga:

def toys(names, color, / ):

    print(f'You bought {color} {names}' )

toys('helicopter','orange')

agarda parametrlardan keyin / belgisini ishlatsak, functionni call qilyotganimizda key = value berib otirish shart bo’lmaydi va python ni ozi order boyicha qooyib beradi.

**Keyword-Only Arguments**

Bu esa faqat keyword bilan berilganda argumentlar qabul qiladi: \* bilan berialdi, bu belgidan keyingi kirgiziladigan argumentlar keyword bilan berilishi kerak x =3 masalan:

def my\_function(\*, x):  
  print(x)  
  
my\_function(x = 3)

Any argument before the / , are positional-only, and any argument  after the \*, are keyword-only.

def my\_function(a, b, /, \*, c, d):  
  print(a + b + c + d)  
  
my\_function(5, 6, c = 7, d = 8)

### **Using** time.sleep()

The time.sleep() function can be used to pause the program for a specific period of time (in seconds) before executing the function

**Lambda**

lambda arguments : expression

Lambda bir nechta argumentlarni olishi mumkin, lekin faqatgina bitta expression oladi.

x = lambda x : x + 15

print(x(3))

Bu kod bilan kiritilgan raqamga 15 ni qoshishni buyuramiz.

x = lambda a, b : a \* b  
print(x(5, 6))

Ikkita argumentni bir biriga kopaytirish functioni.

x = lambda a, b, c : a + b + c  
print(x(5, 6, 2))