List

The len() function returns the number of items in an object.

```
mylist = ["apple", "pear", "peach"]
print(mylist)                           # return: ['apple', 'pear', 'peach']
```

```
x = len(mylist)
print(x)                                # return: 3
```

f-string

A formatted string literal or f-string is a string

```
name = "Fred"
print(f"He said his name is {name}.")   # return: He said his name is Fred.
```

Function

The def keyword is used to create, (or define) a function.

```
def my_function():
    print("Hello from a function")

my_function()                           # return: Hello from a function
```

for in range

```
total = 0
for number in range(1, 101):
    if number % 2 == 0:                 # Csak a páros számokat adja össze 1-100-
ig
        total += number
print(total)                            # return: 2550
```

```
total1 = 0
for number in range(1, 101):
```

```
    total1 += number                    # Minden számot összead 1-100-if
print(total1)                           # return: 5050
```

```
x = "awesome"

def myfunc():
    x = "fantastic"
    print("Python is " + x)

myfunc()                                # return: Python is fantastic
```

Global keyword

We use global keyword to read and write a global variable inside a function.

```
x = "awesome"

def myfunc():
    global x
    x = "fantastic"

myfunc()
print("Python is " + x)                 # return: Python is fantastic
```

Import module

```
import random

print(random.randrange(1,5))            # return: random number 1-4
```

String methods

Strip method remove spaces at the beginning and at the end of the string.

```
x = "Hello World"
print(x[1:8])                           # return: ello Wo
print(x.upper())                        # return: HELLO WORLD
print(x.lower())                        # return: hello world
print(x.strip())                        # return: Hello World
print(x.replace("H", "J"))              # return: Jello World
print(x.split(","))                     # return: ['Hello World']
```

Format method

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))  # return: I want 3 pieces of item
567 for 49.95 dollars.
```

List Append Method

```
fruits = ["banana", "orange", "kiwi", "apple", "pear"]
newlist = []
for x in fruits:
        if "a" in x:
            newlist.append(x)

print(newlist)                  # return: ['banana', 'orange', 'apple', 'pear']
```

Sort

```
thislist = ["banana", "orange", "kiwi", "apple", "pear"]
thislist.sort()
print(thislist)                     # return: ['apple', 'banana', 'kiwi',
'orange', 'pear']
```

List (start:end)

```
letters = ["a", "b", "c", "d"]
letters[1:3]
print(letters[1:3])                 # return: ['b', 'c']
```

Tuple

```
thistuple = ("apple", "banana", "cherry")   # return:   apple
                                                        banana
                                                        cherry
for x in thistuple:
    print(x)
```

```
thistuple = ("apple", "banana", "cherry")
for i in range(len(thistuple)):             # return:   apple
                                                      banana
                                                      cherry
print(thistuple[i])
```

```
thistuple = ("apple", "banana", "cherry")
i = 0                                        # return:  apple
                                                      banana
                                                      cherry
while i < len(thistuple):
    print(thistuple[i])
    i = i + 1
```

class

```
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
    def printname(self):
        print(self.firstname, self.lastname)
x = Person("Csaba", "Bajzáth")
x.printname()                                   # return: Csaba Bajzáth
```

abs()

Python abs() function is used to return the absolute value of a number, i.e., it will remove the negative sign of the number.

```
x = abs(-7.25)

print(x)                                            # 7.25
```

```
def myfunc(a = "bad"):
    print("Python is " + a)
myfunc(a = "awesome")                           # return: Python is awesome
```

Dictionary

```
dict = {
    "Csaba":"Szilvi",
    "Márk":"Zsani",
    "Peti":"Bogi",

}
print(dict)                                     # return: {'Csaba': 'Szilvi',
'Márk': 'Zsani', 'Peti': 'Bogi'}
```

```
myset = {"alma", 12, True}
print(myset)                                    # return: {True, 12, 'alma'}
```

```
a = "Hello World"
print(a[1])                                         # return: e
```

```
for x in "banana":
    print(x)                                        # return:   b a n a n a
```

```
a = "Hello World"
print(len(a))                                       # return: 11
```

Slicing

```
b = "Hello World"
print(b[2:5])                                       # return: llo
```

```
total = 0
for number in range(1,11):
    if number % 2 == 0:
        total += number
print(total)                                        # return: 30
```

```
n = 0
while n < 10:
```

```
    n += 1
    print(n)                                        # return: 1 2 3 4 5 6 7 8 9 10
```

```
for num in range(0, 10):
    if num % 2 == 1:
        print(num)                                  # return: 1 3 5 7 9
```

```
thislist = ["apple", "banana", "kiwi", "pear", "orange"]
i = 0
while i < len(thislist):
    print(thislist[i])
    i = i + 1                           # return: apple banana kiwi pear orange
```

```
fruits = ["apple", "banana", "kiwi", "pear", "orange"]
newlist = []

for x in fruits:
    if "a" in x:
        newlist.append(x)

print(newlist)                          # return: ['apple', 'banana', 'pear',
'orange']
```

sort()

The sort() method sorts the list ascending by default.

```
sortedlist = [10, 12, 356, 99, 77, 1, 23]
sortedlist.sort()
print(sortedlist)                       # return: [1, 10, 12, 23, 77, 99, 356]
```

```
number = -20
absolute_number = abs(number)
print(absolute_number)                  # return: 20
```

```
x = "alma"
print(x.capitalize())                   # return: Alma
```

```
a = "Banán"
print(a.casefold())                     # return: banán
```

```
x = 5
y = "John"
print(type(x))                          # return: <class 'int'>
print(type(y))                          # return: <class 'str'>
```

```
x, y, z = "Orange", "Banana", "Mandarin"
print(x)                                # return: Orange
print(y)                                # return: Banana
print(z)                                # return: Mandarin
```

```
x = y = z = "Orange"
print(x)                                # return: Orange
print(y)                                # return: Orange
print(z)                                # return: Orange
```

```
fruits = ["Orange", "Banana", "Kiwi"]
x, y, z = fruits
print(x)                                # return: Orange
print(y)                                # return: Banana
print(z)                                # return: Kiwi
```

```
x = "awesome"

def myfunc():
    print("Python is " + x)

myfunc()                                # return: Python is awesome
```

```
def myfunc1(q = "bad"):
    q = "awesome"
    print("Python is " + q)
```

```
myfunc1()                               # return: Python is awesome
```

```
x = "awesome"

def myfunc2():
    global x
    x = "fantastic"

myfunc2()
print("Python is " + x)                 # return: Python is fantastic
```

```
txt = "The best thing in life are free!"
print("free" in txt)                    # return: True
```

```
txt1 = "The best thing in life are free!"
if "free" in txt1:
    print("Yes, 'free' is present." )   # return: Yes, 'free' is present.
```

split()

```
b = "HelloWorld"
print(b[1:6])                           # return: elloW
print(b.split(","))                     # return: ['HelloWorld']
```

format()

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))                  # return: My name is John, and I am 36
```

insert()

```
list = ["apple", "pear", "kiwi", "cherry"]
list.insert(2, "orange")
print(list)                             # return: ['apple', 'pear', 'orange',
'kiwi', 'cherry']
```

append()

```
list1 = ["apple", "pear", "kiwi", "cherry"]
list1.append("melon")
print(list1)                            # return: ['apple', 'pear', 'kiwi',
'cherry', 'melon']
```

extend()

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)                         # return: ['apple', 'banana', 'cherry',
'mango', 'pineapple', 'papaya']
```

extend - tuple

```
list = ['apple', 'banana', 'cherry']
tuple = ("kiwi", "orange")
list.extend(tuple)
print(list)                             # return: ['apple', 'banana', 'cherry',
'kiwi', 'orange']
```

remove()

```
list = ["apple", "banana", "cherry"]
list.remove("banana")
print(list)                             # return: ['apple', 'cherry']
```

pop()

```
list1 = ["apple", "banana", "cherry"]
list1.pop(0)
print(list1)                            # return: ['banana', 'cherry']
```

for in

```
for x in list:
    print(x)

list1 = ["apple", "banana", "pear"]
for i in range(len(list1)):
    print(list1[i])                    # return: apple banana pear
```

while

```
list2 = [1, 2, 3, 4, 5, 6, 7,]
i = 0
while i < len(list2):
print(list2[i])                        # return: 1 2 3 4 5 6 7
i = i + 1
```

```
list3 = [1, 2, 3, 4]
[print(x) for x in list3]              # return: 1 2 3 4
```

```
fruits = ["apple", "banana", "cherry"]
newlist = []
for x in fruits:
    newlist.append(x)

print(newlist)                         # return: ['apple', 'banana', 'cherry']
```

```
fruits1 = ["apple", "banana", "cherry"]
newlist = [x for x in fruits1 if 'a' in x]
print(newlist)                         # return: ['apple', 'banana']
```

tuple

```
tuple = ("apple", "banana", "kiwi", "orange", "mango", "pear")
print(tuple)                           # return: ('apple', 'banana', 'kiwi',
'orange', 'mango', 'pear')
print(len(tuple))                      # return: 6
print(type(tuple))                     # return: <class 'tuple'>
print(tuple[1])                        # return: banana
```

```
print(tuple[1:5])                       # return: ('banana', 'kiwi', 'orange',
'mango')
```

```
x = ("apple", "banana")
print(type(x))                          # tuple: ordered, indexed, unchangeable,
allow duplicate values
y = ["apple", "banana"]
print(type(y))                          # list: ordered, indexed, changeable,
allow duplicate values
z = {"apple", "banana"}
print(type(z))                          # set: unordered, unindexed, unchangeable,
o = {
    "alma" : "apple",
    "körte" : "pear"
}
print(type(o))                          # dictionary: ordered, changeable, do not
allow duplicates
```

While loop

```
n = 1
while n < 100:
    n += 1
    print(n)                            # return: 1 2 3 ....... 99 100
```

For loop

```
all_fruits = ["apple", "banana", "kiwi"]
for fruit in all_fruits:
    print(fruit)                        # apple banana kiwi
```

Negative indexing means start from the end

```
tuple = ("apple", "banana", "kiwi")
print(tuple[-1])                        # return: "kiwi"
```

```
thistuple1 = ("apple", "banana", "cherry")
if "banana" in thistuple1:
    print("Yes")                        # return: Yes
```

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
print(thistuple)                        # return: ('apple', 'banana', 'cherry',
'orange')
```

```
thistuple1 = ("apple", "banana", "kiwi")
y = ("orange",)
thistuple1 += y
print(thistuple1)                       # return: ('apple', 'banana', 'kiwi',
'orange')
```

```
thistuple = ("apple", "banana", "kiwi")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
print(thistuple)                        # return: ('banana', 'kiwi')
```

```
thistuple = ("apple", "banana", "cherry")
for i in range(len(thistuple)):
    print(thistuple[i])                 # return: apple banana cherry
```

```
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2
print(mytuple)                          # return: ('apple', 'banana', 'cherry',
'apple', 'banana', 'cherry')
```

count()

count() method returns the number of times a specified value appears in the tuple

```
thistuple1 = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple1.count(5)
print(x)                                # return: 2
```

Tuple index() method - Search for the first occurrence of the value 8, and return its position:

```
thistuple2 = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple2.index(8)
print(x)                                # return: 3
```

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

```
thisset1 = set(("apple", "banana", "kiwi"))
print(thisset1)                         # return: {'apple', 'banana', 'kiwi'}
```

add()

To add one item to a set use the add() method.

```
thisset4 = {"banana", "cherry", "kiwi"}
thisset4.add("orange")
print(thisset4)                      # return: {'kiwi', 'banana', 'cherry',
'orange'}
```

update()

To add items from another set into the current set, use the update() method.

```
thisset5 = {"pear", "apple", "melon"}
tropical = {"kiwi", "mango", "pineapple"}
thisset5.update(tropical)
print(thisset5)              # return: {'pineapple', 'apple', 'kiwi', 'pear',
'melon', 'mango'}
```

```
thisset6 = {"apple", "banana", "cherry"}
myset = ["kiwi", "orange"]
thisset6.update(myset)
print(thisset6)                             # return: {'orange', 'cherry', 'kiwi',
'apple', 'banana'}
```

```
thisset7 = {"apple", "banana", "cherry"}
thisset7.remove("apple")
print(thisset7)                             # return: {'cherry', 'banana'}
```

pop()

the pop() method to remove an item, but this method will remove the last item

```
thisset8 = {"apple", "kiwi", "pear"}
x = thisset8.pop()
print(x)                                    # return: kiwi
print(thisset8)                             # return: {'apple', 'pear'}
```

You can loop through the set items by using a for loop:

```
thisset0 = {"apple", "banana", "kiwi"}
for x in thisset0:
    print(x)                                # return: apple kiwi banana
```

union()

the union() method that returns a new set containing all items from both sets

```
set1 = {"apple", "banana", "kiwi"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)                                 # return: {1, 'banana', 2, 3, 'kiwi',
'apple'}
```

update()

the update() method that inserts all the items from one set into another

```
set4 = {"apple", "banana", "kiwi"}
set5 = {1, 2, 3}
set4.update(set5)
print(set4)                                 # return: {1, 'banana', 2, 3, 'kiwi',
'apple'}
```

intersection_update()

The intersection_update() method will keep only the items that are present in both sets.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
```

```
x.intersection_update(y)
print(x)                                    # return: {'apple'}
```

intersection method

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

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.intersection(y)
print(z)                                    # return: {'apple'}
```

symmetric_difference_update

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

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.symmetric_difference_update(y)
print(x)                                # return: {'google', 'cherry', 'banana',
'microsoft'}
```

symmetric_difference method

The symmetric_difference() method will return a new set, that contains only the elements that are NOT present in both sets.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.symmetric_difference(y)
print(z)                      # return: {'banana', 'cherry', 'google',
'microsoft'}
```

Dictionaries

Dictionaries are used to store data values in key:value pairs.

```
thisdict = {
    "brand" : "Ford",
    "model" : "Mustang",
    "year"  : "1964",
    "colors" : ["red", "white", "blue"]
}
```

```
print(thisdict)       # {'brand': 'Ford', 'model': 'Mustang', 'year': '1964',
'colors': ['red', 'white', 'blue']}
print(thisdict["brand"])                    # return: Ford
print(type(thisdict))                       # return: <class 'dict'>
```

You can access the items of a dictionary by referring to its key name, inside square brackets:

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

There is also a method called get() that will give you the same result:

```
x = thisdict.get("model")
print(x)                                    # return: Mustang
```

keys()

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

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
y = thisdict.keys()
print(y)                                    # return: dict_keys(['brand', 'model',
'year'])
```

values()

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

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
```

```
}
z = thisdict.values()
print(z)                                    # return: dict_values(['Ford',
'Mustang', 1964])
```

items()

The items() method will return each item in a dictionary, as tuples in a list.

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
b = thisdict.items()
print(b)                          # return: dict_items([('brand', 'Ford'), ('model',
'Mustang'), ('year', 1964)])
```

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

```
thisdict["year"] = 2018
print(thisdict)                             # return: {'brand': 'Ford', 'model':
'Mustang', 'year': 2018}
```

update()

The update() method will update the dictionary with the items from the given argument.

```
thisdict.update({"year": 2020})
print(thisdict)                             # return: {'brand': 'Ford', 'model':
'Mustang', 'year': 2020}
```

```
thisdict["color"] = "red"
print(thisdict)                    # return: {'brand': 'Ford', 'model': 'Mustang',
'year': 2020, 'color': 'red'}
```

```
thisdict.update({"color": "green"})
print(thisdict)                   # return: {'brand': 'Ford', 'model': 'Mustang',
'year': 2020, 'color': 'green'}
```

pop()

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

```
thisdict.pop("model")
print(thisdict)                              # return: {'brand': 'Ford', 'year':
2020, 'color': 'green'}
```

You can loop through a dictionary by using a for loop.

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
for x in thisdict:
    print(x)                                # return: brand model year
```

Print all values in the dictionary, one by one:

```
for x in thisdict:
    print(thisdict[x])                         # return: Ford Mustang 1964
```

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

```
for x in thisdict.values():
    print(x)                                   # return: Ford Mustang 1964
```

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

```
for x in thisdict.keys():
    print(x)                                    # return: brand model year
```

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

```
for x, y in thisdict.items():
    print(x,y)                                  # return:   brand Ford
                                                          model Mustang
                                                          year 1964
```

copy()

Make a copy of a dictionary with the copy() method:

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
mydict = thisdict.copy()
print(mydict)                                   # return: {'brand': 'Ford',
'model': 'Mustang', 'year': 1964}
```

dict()

```
# Another way to make a copy is to use the built-in function dict().
mydict = dict(thisdict)
print(mydict)
```

if ... elif .... else

```
e = 200
f = 33
if f > e:
    print("f greater than e")
elif e == f:
    print("e and f are equal")
else:
    print("e greater than f")                   # return: e greater than f
```

if ... elif

```
c = 33
d = 33
if c > d:
    print("c is greater than d")
elif c == d:
    print("c and d are equal")                      # return: c and d are equal
```

Short Hand If

```
if e > f: print("e is greater than f")
```

Short Hand If ... Else

```
g = 2
h = 200
print("A") if g > h else print("B")
```

multiple else statements on the same line

```
j = 333
k = 333
print("A") if j > k else print("=") if j == k else print("B")
```

python060