Functions

Functions are blocks of code that can be used over and over again to perform a specific action.

As we know, in Python, there are print (), len () etc. Many available functions are defined.

We can use it in our own code by providing access to functions defined in libraries, modules and packages. These are called predefined functions, embedded functions (built-in) or library functions. We can use ready-made functions as well as create our own functions. (User-defined)

Functions prevent code repetition and our code stays more modular and organized.

---

def "function_name"(parameter1,parameter2,..):

> **"Do something"**

return "return something" (depends on functionality)

In [1]:

```
def hello():
  print("Hello Everyone!!")
```

In [2]:

```
hello()   #calling the func   #the functions don't have any parameters
```

```
Hello Everyone!!
```

In [3]:

```
def hello(name):
  print("Hello " + name)
```

In [4]:

```
hello("Asli")
```

```
Hello Asli
```

In [5]:

```
def func_in_func(name1):
    return hello(name1)
```

In [6]:

```
func_in_func("Ugurcan")
```

```
Hello Ugurcan
```

In [7]:

```
def func1():
  print("Hello World!!")


func1()
print("Google")
func1()
func1()
func1()
func1()
```

```
Hello World!!
Google
Hello World!!
Hello World!!
Hello World!!
Hello World!!
```

In [8]:

```
def summ(a,b):
  summ = a + b
  print(summ)
```

In [9]:

```
summ(6.0,7.5)
```

```
13.5
```

In [10]:

```
t = summ(8,9)
t
```

```
17
```

In [11]:

```
def func(x,y):
  summ = x + y
  multip = x * y
  return (summ,multip)

#t = summ
#c = multip
```

In [12]:

```
t,c = func(23,45)

print(t,c)
```

```
68 1035
```

In [13]:

```
func(23,45)
```

Out[13]:

```
(68, 1035)
```

In [14]:

```
print("Sum of the values: " + str(t) + ", Multiplying of the values: " + str(c))
```

```
Sum of the values: 68, Multiplying of the values: 1035
```

In [15]:

```
#Let's write a function that it will square the entered number, but will be terminated wh
en you enter the number 5 and give us an error message.


def sqr(x):
  if x == 5:
    return ("Terminated because you entered 5")

  result = x **2
  return (result)
```

In [16]:

```
sqr(10)
```

Out[16]:

```
100
```

In [17]:

```
sqr(5)
```

Out[17]:

```
'Terminated because you entered 5'
```

In [18]:

```
d = sqr(5)
print(d)
```

```
Terminated because you entered 5
```

In [19]:

```
# Let's write a function that tells you whether the entered number is positive, negative
or zero.



def func(x):
  if x > 0:
    return ("Positive")
  elif x < 0:
    return ("Negative")
  else:
    return ("Zero")
```

In [20]:

```
for i in [-2,5,6,0,-4,-7]:
  print(func(i))
```

```
Negative
Positive
Positive
Zero
Negative
Negative
```

In [21]:

```
#factorial calculation
#0! = 1
#1!= 1
#2!= 2 * 1 =2
#6! = 6 * 5* 4 *3 * 2 *1 = 720


def factorial(num):
  factorial = 1
  if (num == 0 or num == 1):
    print("Factorial: ", factorial)
  else:
    while (num >= 1):
      factorial = factorial * num
      num -= 1
    print("Factorial: ", factorial)


# 1 * 5 = 5 = factorial
# 5 * 4 = 20
```

```
# 20 * 3 = 60
#60 * 2 = 120
# 120 * 1 = 120
```

In [22]:

```
factorial(5)
```

```
Factorial:  120
```

In [23]:

```
def faktoriyel(sayi):
  faktoriyel = 1
  for i in range(1,sayi+1):
    faktoriyel *= i
  return faktoriyel
```

In [24]:

```
faktoriyel(5)
```

Out[24]:

```
120
```

In [25]:

```
#using for loop

def factorial2(num2):
  factorial2 = 1
  if (num2 == 0 or num2 == 1):
    print("Factorial: ", factorial2)
  else:
    for i in range(factorial2, num2+1):
        factorial2 *= i
    print("Factorial: ", factorial2)
```

In [26]:

```
factorial2(6)
```

```
Factorial:  720
```

In [27]:

```
def factorial3(nums):
  factorial3 = 1
  if (nums == 0 or nums == 1):
    return ("Factorial: ", factorial3)
  else:
    for i in range(factorial3, nums+1):
       factorial3 *= i
    return (factorial3)
```

In [28]:

```
x = factorial3(6)
print(x)
```

```
720
```

In [29]:

```
x
```

Out[29]:

```
720
```

In [30]:

In [30]:

```
def hello2(name, capLetter = False):
  if capLetter:
    print("Hello " + name.upper())
  else:
    print("Hello " + name)
```

In [31]:

```
hello2("asli")
```

```
Hello asli
```

In [32]:

```
hello2("Asli", capLetter= True)
```

```
Hello ASLI
```

In [33]:

```
#lambda function
(lambda x: x + 1)(2)
```

Out[33]:

```
3
```

In [34]:

```
full_name = lambda first, last: f'Full name: {first.title()} {last.title()}'
full_name('guido', 'van rossum')
```

Out[34]:

```
'Full name: Guido Van Rossum'
```

* args and ** kwargs

- **args (Non Keyword Arguments)**
- **kwargs (Keyword Arguments)**

In [35]:

```
def multp(*args):
  result = 1
  for i in args:
        result *= i
        print(result)

# *args keeps the data as tuple type.
```

In [36]:

```
multp(4,5,6,7,8,9)
```

```
4
20
120
840
6720
60480
```

In [37]:

```
def multp1(*args):
  result = 2
  for i in args:
        result *= i # result = result * i
        print(result)
```

```
# *args keeps the data as tuple type.
```

In [38]:

```
multp1([4,5,6,7])
```

```
[4, 5, 6, 7, 4, 5, 6, 7]
```

In [39]:

```
multp1(2,3,4,5)
```

```
4
12
48
240
```

In [40]:

```
[4,5,6] * 3
```

Out[40]:

```
[4, 5, 6, 4, 5, 6, 4, 5, 6]
```

In [41]:

```
def func_kwargs(**kwargs):
    print(kwargs)

func_kwargs(name = "Murat", name2 = "Ömer", number=12345, can='Emir', beril='yılmaz')
```

```
{'name': 'Murat', 'name2': 'Ömer', 'number': 12345, 'can': 'Emir', 'beril': 'yılmaz'}
```

In [42]:

```
def salaryCalc(salary):

  if salary < 0:
    return("Invalid value")
  else:
    if 0 < salary <= 1000:
      salary = salary + salary * 0.15
    elif salary <= 2000:
      salary = salary + salary * 0.1
    elif salary <= 3000:
      salary = salary + salary * 0.05
    else:
      salary = salary + salary * 0.025

    return ("New salary: ", salary)
```

In [43]:

```
salaryCalc(-5)
```

Out[43]:

```
'Invalid value'
```

In [44]:

```
salaryCalc(800)
```

Out[44]:

```
('New salary: ', 920.0)
```

In [47]:

```
def salaryCalc2():
```

```
def salaryCalc2():

  salary = float(input("Please enter your current salary: "))

  if salary < 0:
    return("Invalid value")
  else:
    if 0 < salary <= 1000:
      salary = salary + salary * 0.15
    elif salary <= 2000:
      salary = salary + salary * 0.1
    elif salary <= 3000:
      salary = salary + salary * 0.05
    else:
      salary = salary + salary * 0.025

    return ("New salary: ", salary)
```

In [48]:

```
new_salary = salaryCalc2()
print(new_salary)
```

```
Please enter your current salary: 9.8
('New salary: ', 11.270000000000001)
```

Let's write a function that returns a random word from a list.

Modules

import numpy

import tensorflow as tf

import myModules

myModules.myFunc()

from myModules import *

myFunc()

In [49]:

```
words = ["artificial","intelligence","machine","learning","python","programming"]


#from random import *
import random as rnd

def randomWord(words):
  index = rnd.randint(0, len(words)-1)
  return words[index]
```

In [50]:

```
len(words)
```

Out[50]:

```
6
```

In [51]:

```
word = randomWord(words)
print(word)
```

```
python
```

Global & Local Variables

Global & Local Variables

In [52]:

```
x = 5

print(x)
```

```
5
```

In [53]:

```
def display():
  x = 4
  return(x)
```

In [54]:

```
display()
```

Out[54]:

```
4
```

In [55]:

```
print(x)
```

```
5
```

Methods

Functions are called by name, it can take parameters inside and optionally the resulting value can be used outside of the function.

Methods are also called by name, in many ways they are like functions, but calling is performed through an object such as a String or list.

object.methodName(parameter)

In [59]:

```
s = input("Please enter a name: ")


print(s.upper())
```

```
Please enter a name: global ai hub
GLOBAL AI HUB
```

In [60]:

```
#it does not return any value
list1 = [1,2,3,4,5,6]

list1.remove(4)
list1
```

Out[60]:

```
[1, 2, 3, 5, 6]
```

In [61]:

```
list1
```

Out[61]:

```
[1, 2, 3, 5, 6]
```

In [62]:

```
list1.index(6)
```

Out[62]:

```
4
```

In [63]:

```
#return the index of the element with the highest value in a given list.

myList = [45,7,23,6,12,78]

maxElement = max(myList)

maxIndex = myList.index(maxElement)

print(maxIndex)
```

```
5
```

Exceptions

- **Programmer Errors**
- **Program Bugs**
- **Exceptions**

In [64]:

```
# error example,SyntaxError.

print "Hello World!"
```

```
  File "<ipython-input-64-c7f6c99ffc2b>", line 3
    print "Hello World!"
                       ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("Hello World!")?
```

In [65]:

```
#bug example.

num1 = input("Enter the first integer: ")
num2 = input("Enter the second integer: ")

print(num1, "+", num2, "=", num1 + num2)
```

```
Enter the first integer: 4
Enter the second integer: 5
4 + 5 = 45
```

In [67]:

```
#exception example, ValueError.

num3 = int(input("First integer: "))
num4 = int(input("Second integer: "))

print(num3, "/",num4, "=", num3/num4)
```

```
First integer: 4
Second integer: 7.8

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-67-2ca720ac0e5e> in <module>
      2
      3 num3 = int(input("First integer: "))
----> 4 num4 = int(input("Second integer: "))
      5
```

```
      6 print(num3, "/",num4, "=", num3/num4)

ValueError: invalid literal for int() with base 10: '7.8'
```

In [68]:

```
# ZeroDivisionError.


num3 = int(input("First integer: "))
num4 = int(input("Second integer: "))

print(num3, "/",num4, "=", num3/num4)
```

```
First integer: 7
Second integer: 0

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-68-c273cdf866df> in <module>
      5 num4 = int(input("Second integer: "))
      6
----> 7 print(num3, "/",num4, "=", num3/num4)

ZeroDivisionError: division by zero
```

Exception Handling

try:

> **the situations where we can get exceptions**

except "Exception Name":

> **the operations in case of exceptions**

In [69]:

```
x = "Alan Turing"

int(x)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-69-417db27f4b47> in <module>
      1 x = "Alan Turing"
      2
----> 3 int(x)

ValueError: invalid literal for int() with base 10: 'Alan Turing'
```

In [70]:

```
try:
  int(x)

except ValueError:
  print("Please enter an integer value!!!")
```

```
Please enter an integer value!!!
```

In [72]:

```
num3 = input("First integer: ")
num4 = input("Second integer: ")

try:
```

```
  num3_int = int(num3)
  num4_int = int(num4)

  print(num3_int, "/",num4_int, "=", num3_int/num4_int)

except ValueError:
  print("Please enter an integer value!!!")
```

```
First integer: 4
Second integer: 4.5
Please enter an integer value!!!
```

In [73]:

```
num3 = input("First integer: ")
num4 = input("Second integer: ")

try:
  num3_int = int(num3)
  num4_int = int(num4)

  print(num3_int, "/",num4_int, "=", num3_int/num4_int)

except ZeroDivisionError:
    print("Please enter the second input different than 0 value!!!")
    ebru = int(input("Enter a integer number"))
ebru
```

```
First integer: 7
Second integer: 0
Please enter the second input different than 0 value!!!
Enter a integer number8
```

Out[73]:

```
8
```

In [74]:

```
num3 = input("First integer: ")
num4 = input("Second integer: ")

try:
  num3_int = int(num3)
  num4_int = int(num4)

  print(num3_int, "/",num4_int, "=", num3_int/num4_int)

except ValueError:
  print("Please enter an integer value!!!")
except ZeroDivisionError:
  print("Please enter the second input different than 0 value!!!")
except:
  print("Unknown error...")
```

```
First integer: 3
Second integer: 9
3 / 9 = 0.3333333333333333
```

In [75]:

```
num3 = input("First integer: ")
num4 = input("Second integer: ")

try:
  num3_int = int(num3)
  num4_int = int(num4)

  print(num3_int, "/",num4_int, "=", num3_int/num4_int)

except (ValueError, ZeroDivisionError):
  print("Error!!!")
```

```
First integer: 3
Second integer: 0
Error!!!
```

In [76]:

```python
#try/except/as


num3 = input("First integer: ")
num4 = input("Second integer: ")

try:
  num3_int = int(num3)
  num4_int = int(num4)

  print(num3_int, "/",num4_int, "=", num3_int/num4_int)

except ValueError as error:
  print("Error!!!")
  print("Error message: ", error)
```

```
First integer: 3
Second integer: 9.6
Error!!!
Error message:  invalid literal for int() with base 10: '9.6'
```

In [77]:

```python
#exception handling in loop structure


while True:
    num1 = input("First number: (Press q for quit the program): ")

    if num1 == "q":
        break

    num2 = input("Second number: ")

    try:
        num1_int = int(num1)
        num2_int = int(num2)
        print(num1_int, "/", num2_int, "=", num1_int / num2_int)
    except (ValueError, ZeroDivisionError):
        print("Error!")
        print("Please try again!")
```

```
First number: (Press q for quit the program): 4
Second number: 0
Error!
Please try again!
First number: (Press q for quit the program): 6
Second number: 7.3
Error!
Please try again!
First number: (Press q for quit the program): q
```

In [78]:

```python
"""
exception handling in functions
using raise command
"""


def reverse(s):

    if (type(s) != str):
        raise ValueError("Please enter a String type.")
    else:
```

```
        return s[::-1]
```

In [79]:

```
reverse("python")
```

Out[79]:

```
'nohtyp'
```

In [80]:

```
reverse(12)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-80-9be0aeb641b6> in <module>
----> 1 reverse(12)

<ipython-input-78-4362af5e3e81> in reverse(s)
      8
      9     if (type(s) != str):
---> 10         raise ValueError("Please enter a String type.")
     11     else:
     12         return s[::-1]

ValueError: Please enter a String type.
```