FUNCTIONS

First class objects are implemented in python.  
first class objects can be placed inside variables, placed inside data structures, passed inside arguments and can be returned by other functions.

>>> def fco():

    ‘this is a docstring,that does not interact with func’

    ‘ ‘ ‘docstrin 2’ ‘ ‘

    “ “ “ docstring 3” “ “

    “docstrig4”

    return 24

////////////////////////print(fco.\_\_doc\_\_) 🡪 gives ‘this is a docstring,that does not interact with func’ : only the first string is taken as docstring

>>> a=[fco(),42]

>>> a

[24, 42]

>>> def returns\_fco():

return fco()

>>> returns\_fco()

24

>>> def fco\_as\_params(params):

return params

>>> fco\_as\_params(fco())

24

>>> inside\_a\_variable=fco()

>>> inside\_a\_variable

24

///////////////////////

#Functions are first class objects

#functions are called first class OBJECTS because they can be passed as parameters,

#placed inside variables and data structures

def new\_f():

    return 24

a=new\_f()

b=[new\_f(), 34, 'lincoln']

print(a,b)

#///////////////////////////////////////////////////////////////////////////

def function\_object():

    print("Hello World")

Hello = function\_object #Function is being assigned to a variable

Hello() #variable called using paranthesis

hello= function\_object()

print(hello)

#///////////////////////////////////////////////////////////////////////////

def squares\_and\_returns\_integer(*val*):

*val*\*\*=2

    return *val*

def Higher\_order\_function\_that\_prints(*func*):

    print(*func*(x))

x=int(input("enter x "))

Higher\_order\_function\_that\_prints(squares\_and\_returns\_integer) #function being passed as argument

#/////////////////////////////////////////////////////////////////////////////////////

def high\_func(*x*):

    def func\_multiplier(*y*):

        return *x*\**y*

    return func\_multiplier #returning a function

variable\_func = high\_func(10)

print(variable\_func(2))

////////////////

# GLOBAL (global), NON-LOCAL (nonlocal), LOCAL

Any data outside of functions placed inside the environment is a global data, it is **not carried** inside the functions, but data inside the function is carried to every function nested inside that function. And data inside any function can be made global data using the – global keyword.

Any data inside a function is local and cannot be affected by the global data,

In order to call a data inside a function that is not assigned locally, that data must be declared global or assigned a data inside it.

A keyword called nonlocal can be prefixed to data, to carry the changes of the data to the enclosing block

The nonlocal keyword cannot be passed to a global data.

#GLOBAL DATA unless explicitly assigned with global keyword

#is otherwise unaffected from assigning values inside functions

a="AAA"

b="BBB"

x="XXX"

y="YYY"

z="ZZZ"

def atmos():

    global a #global a gets altered to - 111

    a=111 #this is now global

    b=222

    x=333

    y=444

    z=555

    def terra():

        global a #global a gets altered again to -1

        global x #global a gets altered again to -2

        a=1 #this is now global

        #b's value is the data from the enclosed loop

        x=3 #this is now global

        y=4

        z=5

        def subway():

            a="one"

            #nonlocal a/x - returns Syntax error when non local used for a global variable

            #b's value is the data from the enclosed loop

            x="three"

            nonlocal y #non local y gets assigned for the enclosed block terra()'s : y to -"three"

            y="four"

            global z #only global z gets altered to -"four"

            z="five" #this is now global

            print(a,b,x,y,z)

        subway()

        print(a,b,x,y,z)

    terra()

    print(a,b,x,y,z)

atmos()

print(a,b,x,y,z)

/////////////////////

LIFETIME: Every localised data inside a function has a finite lifetime, that is... The data ceases from the memory after the function execution is completed.