**Lab4**

*Functions and Modules*

https://www.tutorialspoint.com/python

http://marcuscode.com/lang/python/functions

**SECTION 1: FUNCSIONS**

1. Defining a function

def printme( str ):

"This prints a passed string into this function"

print (str)

return

<<Try to understand the function elements>>

Calling function

printme("This is first call")

printme("Again second")



1. Pass by reference

# Function definition is here

def changeme( mylist ):

"This changes a passed list into this function"

print ("Values inside the function: ", mylist)

mylist[2]=50

print ("Values inside the function after change: ", mylist)

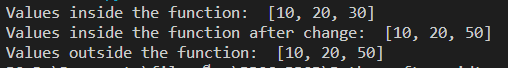
return

# Now you can call changeme function

mylist = [10,20,30]

changeme( mylist )

print ("Values outside the function: ", mylist)



One more example where argument is being passed by reference and the reference is being overwritten inside the called function.

# Function definition is here

def changeme( mylist ):

"This changes a passed list into this function"

# This would assign new reference in mylist

mylist = [1,2,3,4]

print ("Values inside the function: ", mylist)

return

# Now you can call changeme function

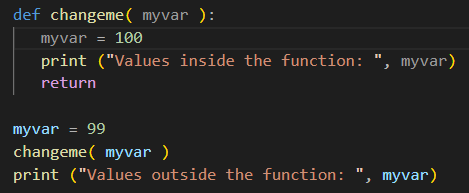
mylist = [10,20,30]

changeme( mylist )

print ("Values outside the function: ", mylist)



**Exercise:** Let’s try to pass on of the primitive datatype (e.g. Integer, Float, String, Boolean) and see that it is passed by reference or pass by value





1. Arguments

Required arguments

# Function definition is here

def printme( str ):

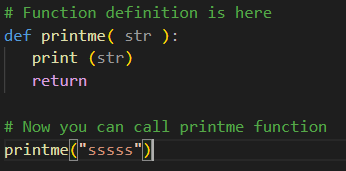
"This prints a passed string into this function"

print (str)

return

# Now you can call printme function

printme()





Keyword arguments

# Function definition is here

def printme( str ):

"This prints a passed string into this function"

print (str)

return

# Now you can call printme function

printme( str = "My string")



# Function definition is here

def printinfo( name, age ):

"This prints a passed info into this function"

print ("Name: ", name)

print ("Age ", age)

return

# Now you can call printinfo function

printinfo( age = 50, name = "miki" )



Default arguments

# Function definition is here

def printinfo( name, age = 35 ):

"This prints a passed info into this function"

print ("Name: ", name)

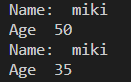
print ("Age ", age)

return

# Now you can call printinfo function

printinfo( age = 50, name = "miki" )

printinfo( name = "miki" )

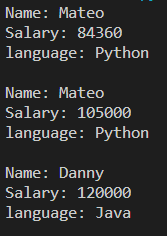


Text

Description automatically generated with medium confidence

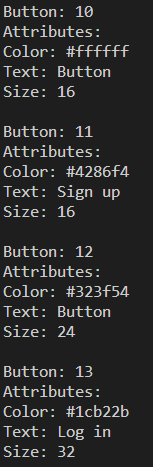
Text

Description automatically generated with low confidence



Text

Description automatically generated



Variable-length arguments

# Function definition is here

def printinfo( arg1, \*vartuple ):

"This prints a variable passed arguments"

print ("Output is: ")

print (arg1)

for var in vartuple:

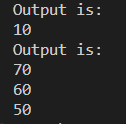
print (var)

return

# Now you can call printinfo function

printinfo( 10 )

printinfo( 70, 60, 50 )



<<what is the value of arg1 in printinfo(70, 60, 50)>>

70

<<what is the value of \*vartuple in printinfo(70, 60, 50)>>

(60, 50)

1. Return statement

# Function definition is here

def sum( arg1, arg2 ):

# Add both the parameters and return them."

total = arg1 + arg2

print ("Inside the function : ", total)

return total

# Now you can call sum function

total = sum( 10, 20 )

print ("Outside the function : ", total )



1. Lambda function

# Function definition is here

sum = lambda arg1, arg2: arg1 + arg2

# Now you can call sum as a function

print ("Value of total : ", sum( 10, 20 ))

print ("Value of total : ", sum( 20, 20 ))



A picture containing text

Description automatically generated



Calendar

Description automatically generated



Built-in functions for lambda: filter() and map()

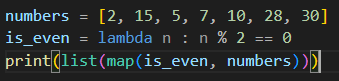
Calendar

Description automatically generated



True or False

is\_even = lambda n : n % 2 == 0





Using Lambda with map (changing the sequence with new value)

Calendar

Description automatically generated with medium confidence



**Exercise**: Describe the different between filter and map

Filter

Filter function takes 2 arguments, a function and a list. It returns a list of values that the given function returns true upon

Map

Map function takes 2 arguments, a function and a list. It returns a list of values that was passed through the given function

1. Global and local variables

total = 0 # This is global variable.

# Function definition is here

def sum( arg1, arg2 ):

# Add both the parameters and return them."

total = arg1 + arg2; # Here total is local variable.

print ("Inside the function local total : ", total)

return total

# Now you can call sum function

sum( 10, 20 )

print ("Outside the function global total : ", total )



1. Example

Chart, scatter chart

Description automatically generated





Graphical user interface, text, application

Description automatically generated





**SECTION 3: I/O**

1. Open and close file

# Open a file

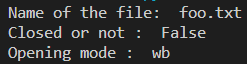
fo = open("foo.txt", "wb")

print ("Name of the file: ", fo.name)

print ("Closed or not : ", fo.closed)

print ("Opening mode : ", fo.mode)

fo.close()



1. Write a file

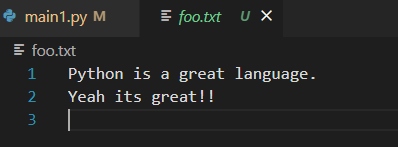
# Open a file

fo = open("foo.txt", "w")

fo.write( "Python is a great language.\nYeah its great!!\n")

# Close opend file

fo.close()



1. Read a file

# Open a file

fo = open("foo.txt", "r+")

str = fo.read(10)

print ("Read String is : ", str)

# Close opened file

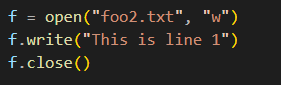
fo.close()



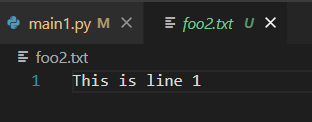
**Exercise:** Try all the mode to open the file. Use your own example to explain the different of each mode. You must display the result with explanation.

**Mode “w”**

Open the file in write mode. This mode is for writing only. It creates the file if the file does not already exist and it overwrites the file if the file already exists.

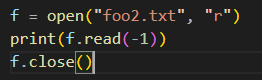


Result



**Mode “r”**

Open the file in read mode. This mode is for reading the content within the file only. This is also the default mode.

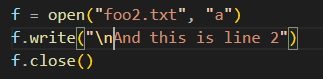


Result



**Mode “a”**

Open the file in append mode. It puts the pointer at the end of the file and you will be able to start writing from there. If the file does not exist, it creates the file.



Result

