**Python training Notes:**

**Course Name:** **SCRIPT 307: Basic Python**

**Day 3: 13 Sep 2017**

**Expectation Setting : ASL (Assisted Self-Learning) 3Hrs session daily**

**And then do self-study and hands on assignments form below learning course link:**

<https://knowledgecenter.persistent.co.in/ViewCourse/pmoc>

**Micro Learning Online link :** <https://persistentuniversity.persistent.co.in/microlearning/Course/837>

Welcome to Micro Learning!!

You can now complete a course by learning small units at a time, i.e. learning Micro Nuggets. Take out 15 minutes time daily and complete one micro nugget and a short quiz. Click on Complete button to ensure the completion and to get access to next Micro Nugget on next day.

Use the link given above to visit the Micro Learning Site to view the next micro nugget.

***Please visit the following URL to view the collaborative learning group***

<https://persistentuniversity.persistent.co.in/CollaborativeLearningGroup/View.aspx?SkillID=8451>

**Topics:**

**Functions**

**\*\*\*\*\*To Do for Day3:**

Nugget 1 : Introduction to Python & Python Fundamentals

Nugget 2 : Python Basics

Nugget 3 : Python Control Structures

663

Nugget 4 : Functions & Modules

1. Complete reading above Nuggets

2. Please excute all codes in these Nuggets

3. Start solving assignment at the end of Nuggets

4. Solve Subjective Assignment for Nugget 1 to 3 : Only for self Practice

**Try Below Codes:**

**13\_for\_demo.py**

#for loop

x = [1.0, 2.0, 3.0] #access first element x[0] o/p 1.0

for n in x:

print(n)

n=n\*\*2 #modifying n does not change X list element

print x

print "--------------------------------------------"

# for and range

x = [1, 3, -7, 4, 9, -5, 4]

for i in range(len(x)): #len(x) is 7 range(7) =[0,1,2,3,4,5,6]

#i is indexes of your list

print i ," = ", x[i]

"""

if x[i] < 0:

print("Found a negative number at index ", i)

x.append(1,2)

"""

**Functions**

**1\_Func1.py**

def func2(): #function definations

print "In Func2...."

def func1():

print "In func1....."

func2()

return "Hello"

'''

print "Start of the script"

func2() #function call

print "END!!"

'''

ret = func1() #calling/invoking a function

print "retr value of function call = ", ret #default return value of a function is None

print "END!!!!!"

**2\_Func1\_return.py**

def func2():

print "In Func2...."

def func1():

print "In func1....."

func2()

#return "SUCCESS" #return string

#return ['xyz',1,3.145] #return LIST

result = func1()

print result

**3\_Func1\_parameter.py**

#Keyword parameters

def hello(name): #catching place -formal argument name="Python"

"""Parameter passing"""

print "Hello "+name

result = hello() #TypeError: hello() takes exactly 1 argument (0 given)

#result = hello("Python") #invoking place

print result

**4\_Func1\_parameter\_by\_name.py**

#2)parameter passing by names

def greet(title, name): #keyword arguments - formal/catching arguments

print "Hello "+title+ " "+name

#greet("ABC", "Mr.")

greet(name="ABC", title="Mr.") #sequence can be anything with these named parameter

#greet(name=[1,2,3,4])

**5\_Func1\_parameter\_dafault.py**

#3) fixed/default value parameter

def calculate\_tax(cost, rate=0.2): #1)cost = 1000 rate=0.2 2)cost = 1000 rate=0.5

return cost+(cost\*rate)

print "Tax with Default rate 0.2 = ",calculate\_tax(1000) #default rate 0.2 will be considered

print "Tax with rate 0.5 passed value = ",calculate\_tax(1000,0.5) #passes rate=0.5 will be considered

**6\_Func1\_parameter\_variable.py**

#4) variable number of arguments : First way of variable arguments

def tupleVarArgs(arg1, arg2='defaultB', \*theRest): #\*theRest is tuple variable as variable arguments

"""varaible argumnets"""

print 'formal arg 1:', arg1 #abc

print 'formal arg 2:', arg2 #123

print "Variable argument list = ", theRest

'''

for eachXtrArg in theRest:

print 'another arg:', eachXtrArg

'''

tupleVarArgs('abc', 123, 'xyz', 456.789)

print "----------------------------------------------------"

tupleVarArgs('abc')

**7\_Func1\_parameter\_variable\_dictionary.py**

#\*\*theRest dictionary

def dictVarArgs(arg1, arg2='defaultB', \*\*theRest):

"""arguments in dictionary ,In this \*\*theRest is dictionary"""

print 'formal arg 1:', arg1

print 'formal arg 2:', arg2

print "----------------------------------------------------"

print "Variable argument dictionary =",theRest

"""

for eachXtrArg in theRest.keys(): #keys return me a List [c, id]

print 'Xtra arg %s: %s' % (eachXtrArg, str(theRest[eachXtrArg]))

"""

dictVarArgs(1220, 740.0, c='grail', id='1000')

print "----------------------------------------------------"

**8\_Func\_scope\_of\_var.py**

#Local and Global variables

def func1():

global a

a = 1

b = 2

a="One"

b="Two"

func1()

print "a=",a # a=1 if a is global else a= "One"

print "b=",b #b="two"

print "----------------------------------------------------"

**lambda function**

print lambda:1 #this loads a function without any name anonymous , after : is the return value

print "----------------------------------------------------"

a= lambda:1

print a

print a()

Example 2:

d = lambda p: p \* 2

t = lambda p: p \* 3

x = 2

x = d(x) #x=4 function call pointing to that anonymous fun

x = t(x) #x=12

x = d(x) #24

print x

**11\_Command\_line\_arg.py**

#command line argumnets

import sys

if (len(sys.argv)==2):

print "argv list = ",sys.argv #is a list

print "First argumnet fron argv list = ",sys.argv[0] #current file name

print "Second argument from argv list =",sys.argv[1]

print dir(sys)

else:

print "Insufficient arguments!!!try again!!!"

Assignment:

1. Accept 2 numbers from keyboard. Pass these as keyworded arguments and let function return the addition answer.
2. Complete Emp dictionary example with multiple records adding in disctionary as per keyboard input till blank entry**.**

**------------------------------------------------------------------------------------------------------------------------------**