**Python training Notes:**

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

**This is Part 1 of the whole training in the duration 17 to 31 July**

**This will be followed with next Part 2 session for Intermediate Python topics in the month of August.**

**Day 5: 23 Jul 2018 - Monday (2 Hrs Session)**

**Expectation Setting ASL (Assisted Self-Learning) 2Hrs session daily**

**And then do self-study and hands on assignments from below learning course link and the assignments given below here in this document:**

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

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

<https://persistentuniversity.persistent.co.in/CollaborativeLearningGroup/view.aspx?SkillId=9144>

**Topics Covered:**

Functions –

variable number of arguments \*theRest, \*\*theRest variables

global

lambda anonymous functions

List Comprehension , map () function , lambda usage in map

**\*\*\*\*\*To Do for Day 5:**

Nugget 1 : Introduction to Python & Python Fundamentals

Nugget 2 : Python Basics

Nugget 3 : Python Control Structures

Nugget 4 : Functions & Modules

Subjective Assignment for Nugget 1 to 3 : Only for self Practice

522

1. Complete reading these 4 Nuggets from <https://knowledgecenter.persistent.co.in/ViewCourse/pmoc>

2. Please execute all codes in these 4 Nuggets

3. Start solving assignment at the end of Nuggets

**Try Below Codes:**

**Functions**

**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**

#Local and Global variables

def func1():

global a

a = 1

b = 2

#----------------------------------------------------

def func2():

print "in func2 = ", a #1

a="One"

b="Two"

func1()

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

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

func2()

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()

#2nd example

def a1(num):

print "In a1 function..................",num

print a1(10) #interpreter will transfer the control to that anonymous function by passing 10 a 1 parameter

#alternative

a1= lambda x :x\*\*2 #def func1(x):return x\*\*2

result = a1(5)

print "Result = ", result

#a1() #TypeError: <lambda>() takes exactly 1 argument (0 given)

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

print (lambda:5)() #(lambda:5) this loads the anionymous function in memory

# () does the job of function call

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

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

Use of lambda:

#Use of lambda : Functional programming in Python : map() example

import math

def multiply\_by\_10 (num):

return num\*10

list1 = [1,2,3,4,5]

#1) for loop

newlist1 = []

for i in list1:

newlist1.append(multiply\_by\_10(i))

print "Original list = ", list1

print "Multiplied list = ",newlist1

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

#2) List Comprehension : alternative

newlist2 = []

#newlist2 = [multiply\_by\_10(i) for i in list1]

newlist2 = [i\*10 for i in list1]

print "Original list = ", list1

print "Multiplied list = ",newlist2

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

#3) alternative to above list comprehension - map() funtioncall

newlist3 = []

newlist3 = map(multiply\_by\_10 , list1) #multiply\_by\_10(list1[0]), multiply\_by\_10(list1[1])

print "Multiplied list = ",newlist3

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

#4)Alternative - map + lambda

newlist4 = []

newlist4 = map(lambda num : num\*10,list1)

print "Multiplied list = ",newlist4

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

"""

map(addn, l1,l2)

list.sort()

list.sort(key = len)

list.sort()

"""

elements = [1,4,9,16,25]

#using map get square root elements

square\_root\_elements = map(math.sqrt,elements)

print "Original elements = ",elements

print "Square root elements list = ",square\_root\_elements

square\_root\_elements1 = map(lambda num : math.sqrt(num),elements)

print "Square root elements list = ",square\_root\_elements1

**Assignments to do:**

1.

Given a string

sentence = 'It is raining cats and dogs'

get 1 target list with length of each word in this sentence

Hint : Use map, lambda, split appropriately

Save the solutions in a folder: **Assignments\Day5**

**Assignments\Day5 --🡪**

**Q1.py**