**Python training Notes:**

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

**Day 2: 18 Feb 2019**

**Post the session from my side do self-study and hands on assignments form below learning course link:**

<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=10329>

**Topics Covered:**

String

List, Tuple

Functions for List Tuple

For loop

range() function

List and Dictionary Comprehension

Functions

File Handling

**\*\*\*\*\*To Do for Day2 :**

Nugget 1 : Introduction to Python & Python Fundamentals

Nugget 2 : Python Basics

Nugget 3 : Nugget 3 : Python Control Structures

Nugget 4 : Functions & Modules

Nugget 5 : I/O & Exception handling

522

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

2. Please execute all codes in these Nuggets

3. Start solving assignment at the end of Nuggets

**Try Below Codes:**

**Errors**

**Python Errors:**

1. **SyntaxError**

**Example 1:**

print “PSL

**Example 2:**

#def addition(num1=1, num2): #SyntaxError

**Example 3:**

def addition(num1, \*x, num2 =2): #Syntaxerror

**Example 4:**

def addition(num1, \*\*x, num2 =2): #Syntaxerror

1. **IndentationError**

if num1 == 100: #IndentationError: expected an indented block

print "In if block...."

print "Value of num1 = ", num1

else:

print "In else block..."

**RunTime Errors :Exception class**

1. **NameError**

Example1:

print num1

**Example 2:**

Display() #NameError: name 'Display' is not defined

1. **TypeError:**

**Example 1 :**

Num1 =100

S1 = “ABC”

Num1+ s1

**Example 2:**

t2 = 1,2,3,6

t2[0] = 99

**TypeError: 'tuple' object does not support item assignment**

**Example 3:**

s1 = {1,2,3,3,3,3,3}

print "Set s1 = ", s1 #set([1, 2, 3])

print "S1[0] = ", s1[0] #TypeError: 'set' object does not support indexing

**Example 4:**

s2 = {1,2,3,3,3,3,[99,88]} #TypeError: unhashable type: 'list'

**Example 5:**

def display(): #function defination

print "Inside display...."

display(100) #TypeError: display() takes no arguments (1 given)

1. **IndexError:**

Example:

list1 = [123, 'abc', 3.145]

print "list1[5] = ", list1[5] #IndexError: list index out of range

1. **ValueError:**

Example1:

list1 = [123, 'abc', 3.145, ['xyz','nil']]

list1.remove(678) #ValueError: list.remove(x): x not in list

1. **KeyError**

**Example 1:**

empData = {'1a':35000, '2a':45000}

print empData[0] #KeyError: 0

1. **IOError**

fob = open("data.txt") #default read mode

#IOError: [Errno 2] No such file or directory: 'data.txt'

**List Comprehension**

"""

get a new list from a existing list by performing some operatio on

every element of a list

module in Python : a file - contains re usable code - functions/ Classes

"""

import math

l1 = [1,2,3,4,5]

double\_list = []

for i in l1:

double\_list.append(i\*2)

print "Original list = ", l1

print "Double list = ", double\_list

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

#alternate solution - List Comprehension

double\_list1 = []

double\_list1=[i\*2 for i in l1] #[2,4,6,8,10]

print "Original list = ", l1

print "Double list = ", double\_list1

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

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

squared\_elements =[]

squared\_elements = [i\*\*2 for i in nums]

print "Squared list = ", squared\_elements

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

squared\_root\_elements=[]

squared\_root\_elements = [math.sqrt(i) for i in squared\_elements]

print "Square rooted elemnts list = ", squared\_root\_elements

squared\_root\_elements\_int = [int(math.sqrt(i)) for i in squared\_elements]

print "Square rooted elemnts list in int= ", squared\_root\_elements\_int

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

#print "Square root of 25 = ", math.sqrt(25)

num2 = [1,2,3,4,5,77,77,88,99]

#get a list of squared elements which are less than 10 in num2

squared1 =[]

squared1 = [i\*\*2 for i in num2 if i <10]

print "Squared elements = ", squared1

**Dictionary Comprehension**

#Dictionary Comprehension

list1 = [1,2,3]

dict1 = {i : i\*\*2 for i in list1} #{1:1, 2:4, 3:9}

print "Orginal list = ", list1

print "Dictionary ", dict1

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

dict1 = {i : i\*\*2 for i in list1 if i<3} #{1:1, 2:4}

print "Orginal list = ", list1

print "Dictionary ", dict1

=====================================

**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

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

#parameter by name: Keyword arguments

def greet(title, name):

print "Hello "+title+ " "+name

print "Lower case string = ",name.lower()

greet("Sangita", "Mrs.") ##Hello Sangita Mrs.

greet(name="ABC", title="Mr.") #keyword arguments passing to a function call

#greet(title="Mr.", "NIL") # "NIL" as normal parameter (non-keyword argument)---this can not be passed after keyword argument

#Syntax error

#greet("NIL", title="Mr.") #TypeError: greet() got multiple values for keyword argument 'title'

#greet("Mr", name="NIL", value =100)#Syntax error

**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

**\*\*File Handling Topic: Execute source codes from Day2\FileDemos folder**

**Assignments to do:**

1. sort the list of following names by an ascending order of number of letters in each name

unsortedList = ['Aaaa', 'bb', 'cccccccc', 'zzzzzzzzzzzz']

1. Covert all words from a list to titlecase using List comprehension

3.

Define dictionary as

emp ={'1a':30000,'2a':40000}

1. Accept employee details from user as

str1 = raw\_input("Enter emp ID:salaray in a single line: ")#"7a :77000"

and such multiple records till user wishes.

1. print all employee details in sorted keys order
2. Increment the salary of every person by 5000 and then print the updated emp

4.Accept 2 numbers from keyboard. Pass these as keyworded arguments and let function return the addition answer.

5.Read file content of given file “empdata”, store it in dictionary.

Display all employee details by sorting employee ID’s.

*1a:ABC:25:25000 (assume 1a as emp ID, ABC as name, 25 as age, 25000 as sal)* and print total sal.

*File data: (Create the file yourself empdata.txt)*

*1a:ABC:25:25000*

*2a:XYZ:30:30000*

*3a:LMN:45:60000*

Hint--🡪

Emp = {} #in memeory data structure storage

FH =open(“file name”)

For line in FH:

Line = line.rstrip()

L1 = Line.split(“:”) #[1a, ABC, 25, 25000]

Emp[L1[0]]=l1[1:]

Emp ={‘1a’:[‘ABC’,25,25000]}

1. **Count of languages:**

**Read Country.txt file**

Store the Country data only for Language and its count in a dictionary.

Display the o/p in below 2 formats -

>>>

**Portuguese 1**

**Franch 4**

**Chinese 1**

**Vietnamese 1**

**German 1**

**English 5**

**Japanese 1**

**Greek 1**

**Indian 1**

**Spanish 3**

**Arabic 2**

**Hungerian 1**

**Italian 1**

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

**{'Portuguese': 1, 'Franch': 4, 'Chinese': 1, 'Vietnamese': 1, 'German': 1, 'English': 5, 'Japanese': 1, 'Greek': 1, 'Indian': 1, 'Spanish': 3, 'Arabic': 2, 'Hungerian': 1, 'Italian': 1}**

>>>

1. **Language and Country**

Store the Country data only for Language and its list of countries in a dictionary.

Display the o/p as shown below -

Write a function for this, store it in a module and access this in your final application.

o/p---🡪

>>>

{'Portuguese': ['Brazil'], 'Franch': ['Cameroon', 'Djibouti', 'Equatorial Guinea', 'France'], 'Chinese': ['China'], 'Vietnamese': ['Vietnam'], 'German': ['Germany'], 'English': ['United Kingdom', 'United States', 'Fiji', 'Canada', 'Ireland'], 'Japanese': ['Japan'], 'Greek': ['Greece'], 'Indian': ['India'], 'Spanish': ['Venezuela', 'Argentina', 'Honduras'], 'Arabic': ['Yemen', 'Bahrain'], 'Hungerian': ['Hungary'], 'Italian': ['Italy']}

Save the solutions in a folder: **Assignments\Day2**

Script names should be Q1.py, Q2.py, Q3.py, Q4.py, Q5.py, Q6.py, Q7.py

**Assignments\Day2 --🡪**

**Q1.py**

**Q2.py**

**Q3.py**

**Q4.py**

**Q5.py**

**Q6.py**

**Q7.py**