**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 10: 30 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:**

Regular Expressions

**\*\*\*\*\*To Do for Day 10:**

Nugget 1 : Introduction to Python & Python Fundamentals

Nugget 2 : Python Basics

Nugget 3 : Python Control Structures

Nugget 4 : Functions & Modules

Nugget 5 : I/O & Exception handling

Nugget 6 : Regular Expressions in Python

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

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

\*\*Also read the Python documentation for the topics covered till date.

**Alternative material for Self Learning:**

**Micro Learning Course**

Also you can enroll to [**SCRIPT 302: BASIC PYTHON**](https://persistentuniversity.persistent.co.in/coursedetails.aspx?cid=837) from Micro Learning Course <https://persistentuniversity.persistent.co.in/microlearning>

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.

You can also view my Python Video recording as a part Smart India Hackathon 2018 at -

<https://www.youtube.com/watch?v=zF-0vJeIxt8&feature=youtu.be>

**Try Below Codes:**

1\_RegEx\_Compile.py

import re

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

regexp = re.compile("PSL") #"PSL" as pattern to be searched where in a given original string

s1="PSLaaaaa Welcome to aaaaaa. PSL persistent......" #match found

s2="aaaaa PSL Welcome to aaaaaa. persistent......" #match not found as function returns None

if regexp.match(s2): #match function matches the pattern at beginning only

print "Match found"

else:

print "Match Not found"

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

**2\_RegEx\_match.py**

import re

s1="aaaaaaPSL Welcome to PSL. PSL, Pune"

s2="PSLWelcome to PSL. PSL, Pune"

pattern =r"PSL" #raw string

#m = re.match(pattern,s2) #match at the beginning, sucessful

m = re.match(pattern,s1) #match at the beginning, sucessful

print "m = ", m #match object

if m!=None:

print "Match found content = ", m.group() #match value is printed here

else:

print "Match Not found!!!"

"""

#match at the begining only

path=R"c:\data\tew\new.txt"

print "Path =", path

print ""

"""

**3\_RegEx\_search2.py**

import re

s="PSL Welcome to aaaaaaPSL. persistent......"

regex = re.search("psl",s, re.IGNORECASE) #, re.IGNORECASE

print "regex object = ", regex

if regex!=None:

print "Match Found = ",regex.group() #PSL

else:

print "Match Not found!!!"

"""

re Flags

re.IGNORECASE makes the pattern case insensitive so that it matches strings of different capitalizations

re.MULTILINE is necessary if your input string has newline characters (\n) and allows the start and end metacharacter

(^ and $ respectively) to match at the beginning and end of each line instead of at the beginning and end of the whole input string

re.DOTALL allows the dot (.) metacharacter match all characters, including the newline character (\n)

"""

**4\_RegEx\_findall.py**

import re

s="Welcome to aPSLb. \*PSL&, Pune "

patt1 ="PSL";

m = re.findall(patt1,s)

print "m = ", m #m = ['PSL', 'PSL'] list will be returned

if m!=None:

print "Match Found = ",m #Match Found = ['PSL', 'PSL']

else:

print "Match Not found!!!"

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

print re.findall('car', 'car') #['car']

print re.findall('car', 'scary') #['car']

print re.findall('car', 'carry the tarcardi to the car') #['car', 'car', 'car']

#find all occurance of PSL

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

**5\_RegEx\_search\_start.py**

import re

s="aaaaaaPSL Welcome to PSL. ^PSL, Pune" #no match

#s="PSLaaaaa Welcome to PSL. PSL, Pune" #match found

m = re.search("\^PSL",s)

#pattern1 =r"^PSL"

#m = re.search(pattern1,s)

#m = re.search("^PSL",s)

print "m = ", m

if m!=None:

print m.group()

else:

print "Match Not found!!!"

#1. Meta characters

#^ used to search exactly at the beginning

# ^ $ / \ & \* ? + | [ ] ( ) { } . these will play a vital role in RegEx pattern

**6\_RegEx\_search\_end.py**

import re

s="\n\*Pune PSL Welcome to PSL. PSL, Pune." #Match found

#s="\n\*Pune PSL Welcome to PSL. PSL, Pune\*" #Match Not found

m = re.search("Pune\.$",s)

if m!=None:

print "Match found = ",m.group()

else:

print "Match Not found!!!"

#match at the end only

**7\_RegEx\_search\_alternative.py**

import re

s="Welcome to PSL. PSL, Pune"

m = re.search("Pune|PSL|AAA|BBB|CC",s) # | is alternative PSL

#m = re.search("Pune|^PSL",s) # | is alternative Pune

print "m = ", m

if m!=None:

print m.group() #

else:

print "Match Not found!!!"

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

#match at the end only

"""

"a|b|c|d"

"a|b|c|d....|z" wrong

OR

"[abcd]" character class group, matches any 1 character

"""

**8\_RegEx\_search\_characterclass.py**

s="PSL Welcome to PSL. PSL, Pune aaa2015bbb"

m = re.search("[A-Za-z]+\s[a-z]+",s)

print "m = ", m

if m!=None:

print m.group() #2015

else:

print "Match Not found!!!"

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

**9\_RegEx\_search\_multiple\_occ.py**

import re

num = raw\_input("Enter number :")

m = re.search("^\d{4}$",num) #is it the precise pattern to validate exact 4 digit number entry from user???

print "m = ", m

if m!=None:

print "match content = ",m.group()

else:

print "Match Not found!!!"

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

"""

{num}

{min,max}

{min,}

[a-z]

[^a-z]

"\\*|\?|\."

[\*?.] all meta characters behave plainly inside [] except ^

**11\_RegEx\_split.py**

import re

s= 'str1:str2:str3'

list1 = re.split(':', s) #split function returns a list ['str1', 'str2', 'str3']

print list1

print len (list1) #3

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

s2= 'str1:str2\*str3?str4'

list2 = re.split('[\*?:]', s2) #\* & + wildcards search("") [0-9] [abcd] "\\*|\?|:" ['str1', 'str2', 'str3', 'str4']

print list2

print len (list2) #4

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

pattern = r'\d'

s3= 'str1:str2\*str3?str4'

list3 = re.split(pattern, s3) #['str', ':str', '\*str', '?str', '']

print list3

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

**12\_RegEx\_substitution.py**

import re

print re.sub("PSL","Persistent","Welcome to PSL, PSL in Pune") #Welcome to Persistent, Persistent in Pune

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

string = "If the the problem is textual, use the the re module"

print "Original string = ",string #Original string = If the the problem is textual, use the the re module

pattern = r"the the"

regexp = re.compile(pattern)

str=regexp.sub("the", string)

print "Modified string = ",str #Modified string = If the problem is textual, use the re module

**12\_RegEx\_substitution2.py**

import re

# Lets try and reverse the order of the day and month in a date

# string. Notice how the replacement string also contains metacharacters

# (the back references to the captured groups) so we use a raw

# string for that as well.

regex = r"([a-zA-Z]+) (\d+)"

#(\1)(\2)

# This will reorder the string and print:

# 24 of June, 9 of August, 12 of Dec

replacement = r"\2 \1"

print re.sub(regex, replacement, "June 24, August 9, Dec 12") #24 June, 9 August, 12 Dec

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

"""

htmltag = "<img>Image Text</img>

#or

#htmltag = "<a>Anchor tag</a>

validate a given html tag for img or a tag

pattern =r""

replacement = r"\2 \1"

"""

**Assignments to do:**

**Assignment 1 on ----->re, file handling**

1. ***Create a text file “emails.txt” and store a big list of valid and invalid email addresses on separate lines. Write a program to match the set of all valid e-mail addresses.***

**Assignment 2:**

1. Create a text file “urls.txt” and store a big list of URL’s. Write a program to match simple Web domain names that begin with "www." and end with a ".com" suffix, e.g., http://www.yahoo.com.

Solve the solutions in a folder: **Assignments\Day10**

**Assignments\Day10 --🡪**

**Q1.py**