K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Science and Humanities

Batch: P5-3 Roll No.: 16010422185
Experiment / assignment / tutorial No. 4
Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date :

**TITLE:** Regular expression in Python

**AIM: Program to demonstrate use of regular expressions in pattern matching.**

**Expected OUTCOME of Experiment:** Use of basic data structure in Python.

**Resource Needed: Python IDE**

**Theory:**

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern. RegEx can be used to check if a string contains the specified search pattern.

*RegEx Module*

Python has a built-in package called re, which can be used to work with Regular Expressions. Import the re module: import re

*RegEx in Python*

When you have imported the re module, you can start using regular expressions:

Example

Search the string to see if it starts with "The" and ends with "Spain":

```
import re
txt = "The rain in Spain"
x = re.search("^The.*Spain$", txt)
```

*RegEx Functions*

The re module offers a set of functions that allows us to search a string for a match:

| Function | Description |
|---|---|
| findall | Returns a list containing all matches |
| search | Returns a Match object if there is a match anywhere in the string |
| split | Returns a list where the string has been split at each match |
| sub | Replaces one or many matches with a string |

*Metacharacters*

Metacharacters are characters with a special meaning:

| Character | Description | Example |
|---|---|---|
| [] | A set of characters | "[a-m]" |
| \ | Signals a special sequence (can also be used to escape special characters) | "\d" |
| . | Any character (except newline character) | "he..o" |
| ^ | Starts with | "^hello" |
| $ | Ends with | "world$" |
| * | Zero or more occurrences | "aix*" |
| + | One or more occurrences | "aix+" |
| {} | Exactly the specified number of occurrences | "al{2}" |
| \| | Either or | "falls\|stays" |
| () | Capture and group | |

---

*Special Sequences*

A special sequence is a \ followed by one of the characters in the list below, and has a special meaning:

| Character | Description | Example |
|---|---|---|
| \A | Returns a match if the specified characters are at the beginning of the string | "\AThe" |
| \b | Returns a match where the specified characters are at the beginning or at the end of a word<br>(the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\bain"<br>r"ain\b" |
| \B | Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word<br>(the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\Bain"<br>r"ain\B" |
| \d | Returns a match where the string contains digits (numbers from 0-9) | "\d" |
| \D | Returns a match where the string DOES NOT contain digits | "\D" |
| \s | Returns a match where the string contains a white space character | "\s" |
| \S | Returns a match where the string DOES NOT contain a white space character | "\S" |

| | | |
|---|---|---|
| \w | Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character) | "\w" |
| \W | Returns a match where the string DOES NOT contain any word characters | "\W" |
| \Z | Returns a match if the specified characters are at the end of the string | "Spain\Z" |

*Sets*

A set is a set of characters inside a pair of square brackets [] with a special meaning:

| **Set** | **Description** |
|---|---|
| [arn] | Returns a match where one of the specified characters (a, r, or n) are present |
| [a-n] | Returns a match for any lower case character, alphabetically between a and n |
| [^arn] | Returns a match for any character EXCEPT a, r, and n |
| [0123] | Returns a match where any of the specified digits (0, 1, 2, or 3) are present |
| [0-9] | Returns a match for any digit between 0 and 9 |
| [0-5][0-9] | Returns a match for any two-digit numbers from 00 and 59 |
| [a-zA-Z] | Returns a match for any character alphabetically between a and z, lower case OR upper case |
| [+] | In sets, +, *, ., \|, (), $,{} has no special meaning, so [+] means: return a match for any + character in the string |

**Problem Definition:**

1. For given program find output

| Sr. No. | Program | Output |
|---|---|---|
| 1 | import re<br>txt = "The rain in Spain"<br>x = re.findall("ai", txt)<br>print(x) | **['ai', 'ai']** |
| 2 | import re<br>txt = "The rain in Spain"<br>x = re.findall("Portugal", txt)<br>print(x) | **[]** |
| 3 | import re<br>txt = "The rain in Spain"<br>x = re.search("\s", txt)<br>print("The first white-space character is located in position:", x.start()) | **The first white-space character is located in position: 3** |

| | | |
|---|---|---|
| 4 | import re<br>txt = "The rain in Spain"<br>x = re.search("Portugal", txt)<br>print(x) | **None** |
| 5 | import re<br>txt = "The rain in Spain"<br>x = re.split("\s", txt)<br>print(x) | **['The', 'rain', 'in', 'Spain']** |
| 6 | import re<br>txt = "The rain in Spain"<br>x = re.split("\s", txt, 1)<br>print(x) | **['The', 'rain in Spain']** |
| 7 | import re<br>txt = "The rain in Spain"<br>x = re.sub("\s", "9", txt)<br>print(x) | **The9rain 9in9Spai n** |
| 8 | import re<br>txt = "The rain in Spain"<br>x = re.sub("\s", "9", txt, 2)<br>print(x) | **The9rain 9in Spain** |
| 9 | import re<br>txt = "The rain in Spain"<br>x = re.search("ai", txt)<br>print(x) #this will print an object | **<re.Match object; span=(5, 7), match='ai'>** |
| 10 | import re<br>txt = "The rain in Spain"<br>x = re.search(r"\bS\w+", txt)<br>print(x.span()) | **(12, 17)** |

2. WAP to verify whether his credit card numbers are valid or not. A valid credit card from ABC Bank has the following characteristics:
   - It must start with a 4,5 or 6 .
   - It must contain exactly 16 digits.
   - It must only consist of digits (0-9).
   - It may have digits in groups of 4, separated by one hyphen '-'

3. From given string extract phone numbers only and save it into list.

Txt = "Dave Martin
615-555-7164
173 Main St., Springfield RI 55924
davemartin@bogusemail.com

Charles Harris

800-555-5669
969 High St., Atlantis VA 34075
charlesharris@bogusemail.com

Eric Williams
560-555-5153
806 1st St., Faketown AK 86847
laurawilliams@bogusemail.com

Corey Jefferson
900-555-9340
826 Elm St., Epicburg NE 10671
coreyjefferson@bogusemail.com"

**Books/ Journals/ Websites referred:**

1. Reema Thareja, *Python Programming: Using Problem Solving Approach*, Oxford University Press, First Edition 2017, India
2. Sheetal Taneja and Naveen Kumar, *Python Programming: A modular Approach*, Pearson India, Second Edition 2018,India

**Implementation details:**

**Code 1:**

C: > Users > User > import re.py > ...

```
1  import re
2  num=input('Enter Credit Card Number - ')
3  #num=
4  if(len(num)==19 and (re.findall('[4-6]\d{3}[-]\d{4}[-]\d{4}[-]\d{4}',num))):
5      print('Your Credit Card Number is Valid.')
6  else:
7      print('Your Credit Card Number is Invalid.')
```

**Code 2:**

C: > Users > User > import re2.py > ...

```python
import re

Txt = '''Dave Martin
615-555-7164
173 Main St., Springfield RI 55924
davemartin@bogusemail.com
Charles Harris
800-555-5669
969 High St., Atlantis VA 34075
charlesharris@bogusemail.com
Eric Williams
560-555-5153
806 1st St., Faketown AK 86847
laurawilliams@bogusemail.com
Corey Jefferson
900-555-9340
826 Elm St., Epicburg NE 10671
coreyjefferson@bogusemail.com'''

phone_numbers=[]
phone_numbers=re.findall("(\d{3}[-]*\d{3}[-]*\d{4})",Txt)
print(phone_numbers)

```

## Output(s):

## Output 1:

```
PS C:\Users\User>  c:; cd 'c:\Users\User'; & 'C:\Users\User\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\User\.vscode\extensions\ms-python.python-2023.4.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '52949' '--' 'c:\Users\User\import re.py'
Enter Credit Card Number - 5743-5624-9810-0105
Your Credit Card Number is Valid.
PS C:\Users\User>
```

```
PS C:\Users\User>  c:; cd 'c:\Users\User'; & 'C:\Users\User\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\User\.vscode\extensions\ms-python.python-2023.4.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '53014' '--' 'c:\Users\User\import re.py'
Enter Credit Card Number - 2457-9564-8321-0090
Your Credit Card Number is Invalid.
PS C:\Users\User>
```

## Output 2:

```
PS C:\Users\User> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/User/import re2.py"
['615-555-7164', '800-555-5669', '560-555-5153', '900-555-9340']
PS C:\Users\User>
```

**Conclusion:** In this experiment, we have learned the use of regular expressions package in python and the functions within it, and also the utilizations of metacharacters and special sequences, sets.

**Post Lab Descriptive Questions**

Differentiate between match and search function? Explain with suitable example.

**Ans-** In Python, both match and search functions are used to search for patterns in strings, but they have some differences in their behavior.

The match function, re.match(), attempts to match the pattern only at the beginning of the string. If the pattern is found at the start of the string, it returns a match object, otherwise, it returns None.

On the other hand, the search function, re.search(), searches for the pattern anywhere in the string even if the string contains multi-lines, and tries to find a match of the substring in all the lines of string and returns the first match found.

Example – Code:

```
# import re module
 import re

Substring =’string’

String1 ="'We are learning regex with geeksforgeeks. regex is very useful for string matching. It is fast too.'"
```

String2 ='"string We are learning regex with geeksforgeeks. regex is very useful for string matching. It is fast too.'"

```
# Use of re.search() Method

print(re.search(Substring, String1, re.IGNORECASE))

#Use of re.match() Method

print(re.match(Substring, String1, re.IGNORECASE))


# Use of re.search() Method

print(re.search(Substring, String2, re.IGNORECASE)) # Use
of re.match() Method

    print(re.match(Substring, String2, re.IGNORECASE))
```

Output:

<re.Match object; span=(75,81), match ='string'> None

<re.Match() object; span(0,6), match='string'>

<re.Match() object; span(0,6), match='string'>Substring ='string'

**Date:** **Signature of faculty in-charge :**