**Batch: P1 - 2 Roll No.: 16014022050**

**Experiment / assignment / tutorial No.**

**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 (the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\bain" 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 (the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\Bain" 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  txt = "The rain in Spain"  x = re.findall("ai", txt)  print(x) | **['ai', 'ai']** |
| 2 | import re  txt = "The rain in Spain"  x = re.findall("Portugal", txt)  print(x) | **[]** |
| 3 | import re  txt = "The rain in Spain"  x = re.search("\s", txt)  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  txt = "The rain in Spain"  x = re.search("Portugal", txt)  print(x) | **None** |
| 5 | import re  txt = "The rain in Spain"  x = re.split("\s", txt)  print(x) | **['The', 'rain', 'in', 'Spain']** |
| 6 | import re  txt = "The rain in Spain"  x = re.split("\s", txt, 1)  print(x) | **['The', 'rain in Spain']** |
| 7 | import re  txt = "The rain in Spain"  x = re.sub("\s", "9", txt)  print(x) | **The9rain9in9Spain** |
| 8 | import re  txt = "The rain in Spain"  x = re.sub("\s", "9", txt, 2)  print(x) | **The9rain9in Spain** |
| 9 | import re  txt = "The rain in Spain"  x = re.search("ai", txt)  print(x) #this will print an object | **<re.Match object; span=(5, 7), match='ai'>** |
| 10 | import re  txt = "The rain in Spain"  x = re.search(r"\bS\w+", txt)  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:**

2.

import re

print("ketaki mahajan / P1-2 / 16014022050")

cardnum = input("Enter your 16 digit credit card number: ") #taking input from user i.e card number to check

if(len(cardnum) == 19 and re.findall('[4-6]\d{3}[-]\d{4}[-]\d{4}[-]\d{4}', cardnum)): #pattern for checking

    print('The credit card number is valid')

elif(len(cardnum) == 16 and re.findall('[4-6]\d{16}', cardnum)):

    print('The credit card number is valid')

else:

    print('Invalid number')

3.

import re

print("ketaki mahajan / P1-2 / 16014022050")

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'''

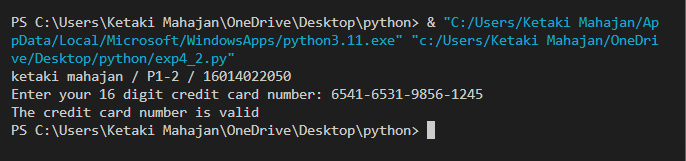
numbers = re.findall('\d{3}[-]\d{3}[-]\d{4}', txt) #checking the pattern

print(numbers)

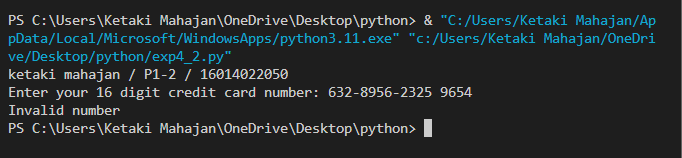
**Output(s):**

2.

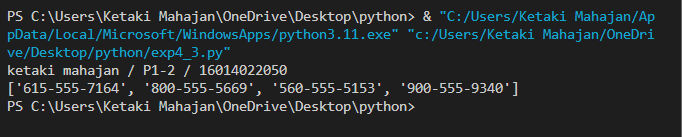
testing for **VALID** credit card number -



testing for an **INVALID** credit card number -



3.



**Conclusion:**

In this experiment, we learnt the use of regular expressions in various applications as well as the basic data structure in Python.

**Post Lab Descriptive Questions:**

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

There is a difference between the use of both functions. Both return the first match of a substring found in the string, however re.match() searches only from the beginning of the string and returns matched object if found. But if a match of substring is found somewhere in the middle of the string, it returns none.

While, re.search() searches for the whole string even if the string contains multi-lines and tries to find a match of the substring in all the lines of string.

e.g.

import re

Substring ='string'

S1 ='''hello'''

S2 ='''hiii'''

print(re.search(Substring, S1, re.IGNORECASE))

print(re.match(Substring, S1, re.IGNORECASE))

print(re.search(Substring, S2, re.IGNORECASE))

print(re.match(Substring, S2, re.IGNORECASE))

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