**Unit- VII**

**Regular Expressions**

The regular expressions can be defined as the sequence of characters which are used to search for a pattern in a string. The module re provides the support to use regex in the python program. The re module throws an exception if there is some error while using the regular expression.

The **re** module must be imported to use the regex functionalities in python.

**import re**

**Regex Functions**

The following regex functions are used in the python.

1)match-

This method matches the regex pattern in the string with the optional flag. It returns true if a match is found in the string otherwise it returns false.

2)search-

This method returns the match object if there is a match found in the string.

3)findall-

It returns a list that contains all the matches of a pattern in the string.

4)split-

Returns a list in which the string has been split in each match.

5)sub-

Replace one or many matches in the string.

## Forming a regular expression

A regular expression can be formed by using the mix of meta-characters, special sequences, and sets.

### Meta-Characters

Metacharacter is a character with the specified meaning.

|  |  |  |
| --- | --- | --- |
| **Metacharacter** | **Description** | **Example** |
| [] | It represents the set of characters. | "[a-z]" |
| \ | It represents the special sequence. | "\r" |
| . | It signals that any character is present at some specific place. | "Ja.v." |
| ^ | It represents the pattern present at the beginning of the string. | "^Java" |
| $ | It represents the pattern present at the end of the string. | "point" |
| \* | It represents zero or more occurrences of a pattern in the string. | "hello\*" |
| + | It represents one or more occurrences of a pattern in the string. | "hello+" |
| {} | The specified number of occurrences of a pattern the string. | "java{2}" |
| | | It represents either this or that character is present. | "java|point" |
| () | Capture and group |  |

### Special Sequences

Special sequences are the sequences containing \ followed by one of the characters.

|  |  |
| --- | --- |
| **Character** | **Description** |
| \A | It returns a match if the specified characters are present at the beginning of the string. |
| \b | It returns a match if the specified characters are present at the beginning or the end of the string. |
| \B | It returns a match if the specified characters are present at the beginning of the string but not at the end. |
| \d | It returns a match if the string contains digits [0-9]. |
| \D | It returns a match if the string doesn't contain the digits [0-9]. |
| \s | It returns a match if the string contains any white space character. |
| \S | It returns a match if the string doesn't contain any white space character. |
| \w | It returns a match if the string contains any word characters. |
| \W | It returns a match if the string doesn't contain any word. |
| \Z | Returns a match if the specified characters are at the end of the string. |

### Sets

A set is a group of characters given inside a pair of square brackets. It represents the special meaning.

|  |  |  |
| --- | --- | --- |
| **SN** | **Set** | **Description** |
| 1 | [arn] | Returns a match if the string contains any of the specified characters in the set. |
| 2 | [a-n] | Returns a match if the string contains any of the characters between a to n. |
| 3 | [^arn] | Returns a match if the string contains the characters except a, r, and n. |
| 4 | [0123] | Returns a match if the string contains any of the specified digits. |
| 5 | [0-9] | Returns a match if the string contains any digit between 0 and 9. |
| 6 | [0-5][0-9] | Returns a match if the string contains any digit between 00 and 59. |
| 10 | [a-zA-Z] | Returns a match if the string contains any alphabet (lower-case or upper-case). |

**The findall() function:-**

This method returns a list containing a list of all matches of a pattern within the string. It returns the patterns in the order they are found. If there are no matches, then an empty list is returned.

**Exampl**e:-

import re

str = "How are you. How is everything"

matches = re.findall("How", str)

print(matches)

**Example:-**

import re

value = "pillu 123 sewwty 456 manujanupari"

list = re.findall("[mp]\w+", value)

print(list)

## The match object

The match object contains the information about the search and the output. If there is no match found, the None object is returned.

### Example

**import** re

str = "How are you. How is everything"

matches = re.search("How", str)

**print**(type(matches))

**print**(matches) #matches is the search object

**The Match object methods**

There are the following methods associated with the Match object.

span(): It returns the tuple containing the starting and end position of the match.

string(): It returns a string passed into the function.

group(): The part of the string is returned where the match is found.

**Example:-**

import re

str = "How are you. How is everything"

matches = re.search("How", str)

print(matches.span())

print(matches.group())

print(matches.string)

**The search() Function**

The search() function searches the string for a match, and returns a Match object if there is a match.If there is more than one match, only the first occurrence of the match will be returned:

**Example**

Search for the first white-space character in the string:

import re

txt = "The rain in Spain"

x = re.search("\s", txt)

print("The first white-space character is located in position:", x.start())

**\*If no matches are found, the value None is returned**:

**Example**

Make a search that returns no match:

import re

txt = "The rain in Spain"

x = re.search("Portugal", txt)

print(x)

**The split() Function**

The split() function returns a list where the string has been split at each match:

**Example**

Split at each white-space character:

import re

txt = "The rain in Spain"

x = re.split("\s", txt)

print(x)

\*You can control the number of occurrences by specifying the maxsplit parameter:

**Example**

Split the string only at the first occurrence:

import re

txt = "The rain in Spain"

x = re.split("\s", txt, 1)

print(x)

**The sub() Function**

The sub() function replaces the matches with the text of your choice:

**Example**

Replace every white-space character with the number 9:

import re

txt = "The rain in Spain"

x = re.sub("\s", "9", txt)

print(x)

You can control the number of replacements by specifying the count parameter:

**Example**

Replace the first 2 occurrences:

import re

txt = "The rain in Spain"

x = re.sub("\s", "9", txt, 2)

print(x)

**Example:-**

import re

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

match = re.search(regex, "I was born on June 24")

if match != None:

print "Match at index %s, %s" % (match.start(), match.end())

print "Full match: %s" % (match.group(0))

print "Month: %s" % (match.group(1))

print "Day: %s" % (match.group(2))

else:

print "The regex pattern does not match."

**Output:-**

Match at index 14, 21

Full match: June 24

Month: June

Day: 24

**Example:-**

import re

value = "vmssangolasoalpur"

m = re.search("(so.\*)", value)

if m:

print("search:", m.group(1))

m = re.match("(so.\*)", value)

if m:

print("match:", m.group(1))

**Output:-**

('search:', 'soalpur')

**Python program to check the validity of a Password**

In this program, we will be taking a password as a combination of alphanumeric characters along with special characters, and check whether the password is valid or not with the help of few conditions.

**Primary conditions for password validation :**

1. Minimum 8 characters.
2. The alphabets must be between [a-z]
3. At least one alphabet should be of Upper Case [A-Z]
4. At least 1 number or digit between [0-9].
5. At least 1 character from [ \_ or @ or $ ].

Here we have used the **re** module that provide support for [regular expressions in Python](http://www.geeksforgeeks.org/regular-expression-python-examples-set-1/). Along with this the re.search() method returns False (if the first parameter is not found in the second parameter) This method is best suited for testing a regular expression more than extracting data. We have used the re.search() to check the validation of alphabets, digits or special characters. To check for white spaces we use the “\s” which comes in the module of the regular expression.

|  |
| --- |
| **Example:-**  import re  password = "R@m@\_f0rtu9e$"  flag = 0  while True:      if (len(password)<8):          flag = -1          break      elif not re.search("[a-z]", password):          flag = -1          break      elif not re.search("[A-Z]", password):          flag = -1          break      elif not re.search("[0-9]", password):          flag = -1          break      elif not re.search("[\_@$]", password):          flag = -1          break      elifre.search("\s", password):          flag = -1          break      else:          flag = 0          print("Valid Password")          break    if flag ==-1:      print("Not a Valid Password") |

**URL in a String**

we will need to accept a string and we need to check if the string contains any URL in it. If the URL is present in the string, we will say URL’s been found or not and print the respective URL present in the string. We will use the concept of Regular Expression of Python to solve the problem.

**Example:-**

|  |
| --- |
| import re   def Find(string):       url = re.findall('http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]      |[!\*\(\), ]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', string)      return url  string = 'My Profile: https://auth.geeksforgeeks.org  / user / Chinmoy % 20Lenka / articles in  the portal of http://www.geeksforgeeks.org/'  print("Urls: ", Find(string)) |

**Output:**

Urls: ['https://auth.geeksforgeeks.org/user/Chinmoy%20Lenka/articles',

'http://www.geeksforgeeks.org/']

**Example:-**

import re

defisValidURL(str):

regex = ("((http|https)://)(www.)?" +

"[a-zA-Z0-9@:%.\_\\+~#?&//=]" +

"{2,256}\\.[a-z]" +

"{2,6}\\b([-a-zA-Z0-9@:%" +

".\_\\+~#?&//=]\*)")

p = re.compile(regex)

if (str == None):

return False

if(re.search(p, str)):

return True

else:

return False

url = "https://www.geeksforgeeks.org"

if(isValidURL(url) == True):

print("Yes")

else:

print("No")

**Email\_validation:-**

Given a string, write a Python program to check if the string is a valid email address or not. An email is a string (a subset of ASCII characters) separated into two parts by @ symbol, a “personal\_info” and a domain, that is personal\_info@domain.

we are using the search() method of re module. so let’s see the description of it.

re.search() : This method either returns None (if the pattern doesn’t match), or re.MatchObject contains information about the matching part of the string. This method stops after the first match, so this is best suited for testing a regular expression more than extracting data.

**Example:-**

import re

regex = r'\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'

def check(email):

if(re.fullmatch(regex, email)):

print("Valid Email")

else:

print("Invalid Email")

if \_\_name\_\_ == '\_\_main\_\_':

email = "ankitrai326@gmail.com"

check(email)

email = "my.ownsite@our-earth.org"

check(email)

email = "ankitrai326.com"

check(email)

**Example:-**

import re

regex = r'\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'

def check(email):

if(re.fullmatch(regex, email)):

print("Valid Email")

else:

print("Invalid Email")

if \_\_name\_\_ == '\_\_main\_\_':

email = input("Enter the email\_id=")

check(email)

**Advantages Of Regular Expression:-**

1. A regular expression is a compact way of describing sets of strings which conform to a pattern. Opportunities to use them arise when dealing with strings with specific structure
2. a regular expression is a pattern describing a certain amount of text. That makes them ideally suited for searching, text processing and data validation.
3. Searching with regular expressions enables you to get results with just one search instead of many searches.
4. Update large amounts of data by searching and replacing with regular expressions. The benefit of using regular expressions to update data is that with a single regex pattern, you can dynamically update a variety of data.