TECHelium

Lets Face it, now

Project Oriented Python

Regular Expressions & Programs

What is Regular Expression?

- Regular expression are most widely used feature of python.
- It is used to search a specific pattern/sub-string in a given string.
- A regular expression is a special sequence of characters that helps you match or find subset of string, using a specialized syntax held in a pattern
- There are two key functions to handle regular expression in python.
  - re.match
  - re.search

Regular Expression application

- Regular expressions are used to search some substring in a given string
  - Search if the port is disabled in output of ‘show ip interface fo/1’
  - Search if system return ‘success’ when some command is given
- Regular expressions are also used to match some string and get its value
  - Search for substring ‘Root bridge’ and get bridge id of root.
  - Search for substring ‘Mac address’ and get its value

Match function

- The function match a pattern in the given string
- Syntax: re.match (pattern, string, flags=0)
  - pattern : This is the regular expression to be matched
  - string : This is the string in which pattern will be searched.
  - flag: one or more modifier that can be clubbed together using bitwise 'OR' (|)

```
#!c:\Python27\python
import re

str = " Spanning tree Bridge ID is 2048:0023:0304:3040"

Matchstring= re.match(r'(.*) is (.*) ', str, re.M|re.I)


print Matchstring.group()

print Matchstring.group(1)

print Matchstring.group(2)
```

Output

```
Spanning tree Bridge ID is 2048:0023:0304:3040
Spanning tree Bridge ID
2048:0023:0304:3040
```

Return

- The re.match function returns a match object on success
  - group(num) of match is used to get the matched expression
- Returns None on failure

Search function

- The function searches the first occurrence of a pattern in the given string

- Syntax: re.search (pattern, string, flags=0)
  - pattern : This is the regular expression to be searched
  - string : This is the string in which pattern will be searched.
  - flag: one or more modifier that can be clubbed together using bitwise 'OR' (|)

```
#!c:\Python27\python
import re

str = " Spanning tree Bridge ID is 2048:0023:0304:3040"

Matchstring= re.search(r' is (.*) ', str, re.M|re.I)


print Matchstring.group()

print Matchstring.group(1)
```

Output

```
is 2048:0023:0304:3040
2048:0023:0304:3040
```

Difference between Match and Search

- Match function start matching the pattern from the start of the string.
- If the pattern does not matches the beginning of string, it return null
- For example if pattern is ‘face’ , then match function will check if the string starts with ‘face’
- Search function search the pattern in between the string.
- For example if the pattern is ‘face’, then search function will search for face in between the string.

Regular expression pattern

- Except for some control characters, **all characters match themselves.**
- **+, ?, ., *, ^, $, ( ), [ ], { }, |, **
- To match ‘harish’ in a string ‘My name is Harish’ the regular expression pattern could be ‘harish’ itself
- With re.I flag we can ignore case sesitivity.

```
#!c:\Python27\python
import re

str = “ My name is Harish HARISH Kandpal”

Matchstring= re.search(r’ harish (.*) ’, str, re.M|re.I)


print Matchstring.group()

print Matchstring.group(1)
```

Output

```
Harish HARISH Kandpal
HARISH Kandpal
```

Exercise

- Write a program to find out the output of following thing from string “There was many Hare , but Are difficult to take CARE of” .
  - r’are ‘ with re.I flag
  - r’Are with and without re.I flag
  - r’ARE with and without re.I flag

- Write a program to print the name along with the sir name of the candidate from string “My Name is xxxx yyyy” where xxxx is any name and yyyy is any sir name. Use only name as pattern.

Re pattern – control characters '+'

- +
- re+ : One or more occurrence of preceding expression
  - Example: r'cap+' means ca + 1 or more p.
  - Matches: cap, capp, cappp, cappppp
  - Not matches: ca
- Quiz: list matching for r'a+b' , r'cat+rat'

```
#!c:\Python27\python
import re

str ing= ['I have cap', 'I have cat', 'I have capped']

for i in string:
    mat = re.search(r'cap+', i,re.I)
    if mat:
        print mat.group()
    else :
        print "no match in", i

print "over"
```

Output

```
cap
no match in 1
capp
over
```

Re pattern – control characters '?'

- **?**
- re? : Zero or one occurrence of preceding expression
  - Example: cap? means ca + o or one p.
  - Matches: ca, cap, capp, cappp,
  - cap?a does not matches cappa. It only matches caa or capa
- Quiz: list matching for r'a?b' , r'cat?a', r'ca?b+

```
#!c:\Python27\python
import re

str = ['I have cap', 'I have cat', 'I have capped']

i=0
while i < 3 :
  mat = re.search(r'cap?', str[i],re.I)
  if mat:
     print mat.group()
  else :
     print "no match in", i

  i += 1

print "over"
```

Output

```
cap
ca
capp
over
```

Re pattern – control characters ‘*’

- *
- re* : Zero or more occurrence of preceding expression
  - Example: cap*a means ca + o or more p + a.
  - Matches: caa, capa, cappa, cappppa,
- Quiz: list matching for r’a*b’ , r’cat*a’, r’ca*bc+d?’

```
#!c:\Python27\python
import re

str = ['I have cap', 'I have cat', 'I have capped']

i=0
while i < 3 :
  mat = re.search(r'cap?', str[i],re.I)
  if mat:
     print mat.group()
  else :
     print "no match in", i

  i += 1

print "over"
```

Output

```
cap
ca
capp
over
```

Re pattern – control characters ‘{n}’

- **{n} {n,}**
- re{n} : n number of occurrence of preceding expression
- Re{n,} : n or more occurrence of preceding expression
  - Example: cap{2} means ca + 2 p.
    - Matches: capp,
    - Not match: cap, cappp
  - Example cap{2,} or cap{2,4}
- Quiz: list matching for r’a{3}b’ , r’cat{1,5}at’, r’are{3}b+c*d?’

```
#!c:\Python27\python
import re

str = ['I have cap', 'I have capp', 'I have cappped']

i=0
while i < 3 :
  mat = re.search(r'cap{2}', str[i],re.I)
  if mat:
     print mat.group()
  else :
     print "no match in", i

  i += 1

print "over"
```

Output

```
no match in 0
capp
capp
over
```

Re pattern – control characters ‘.’

- .
- re. : Matches any single character except newline. Example: cap.t means cap + any charater+ t .
  - Matches: capat, capnt ,cappt
  - Not match: captpp, cappptp
- Quiz: list matching for r’c.b’ , r’cat.+’, r’are.*’

```
#!c:\Python27\python
import re

str = ['I have cap', 'I have cappt', 'I have capppted']

i=0
while i < 3 :
  mat = re.search(r'cap.t', str[i],re.I)
  if mat:
     print mat.group()
  else :
     print "no match in", i

  i += 1

print "over"
```

Output

```
no match in 0
cappt
no match in 2
over
```

Re pattern – control characters '()'

- ()
- (re) : Group regular expression and remember matched text. It is used to collect the data.
  - Example: '(.*) cap (.*) would collect all text before and after the cap.
- Quiz: Collect IP address from "IP address: Interface IP address 120.2.3.4"

```
#!c:\Python27\python
import re

strn  = " I have blue cap that is big"


mat = re.search(r'(.*) cap (.*)', strn,re.I)

if mat:
    print mat.group()
    print mat.group(1)
    print mat.group(2)

print "over"
```

Output

```
I have blue cap that is big
I have blue
that is big
```

Re pattern – control characters ‘[]’

- []
- Re[] : Matches any single character in [].
  - Example: ‘[abc] would collect any character a or b or c.

”

```
#!c:\Python27\python
import re

strn  = “ I have blue cap that is big”


mat = re.search(r‘([abc].*) ‘, strn,re.I)

if mat:
    print mat.group()

print "over"
```

Output

```
blue cap that is big
over
```

Re pattern – other control characters

- ^ : matches start of line
  - r'^This .*' : means find 'This' which is at start of line'
- $: Matches end of line
- \s: Match whitespace
  - r'(.*)\s' : will capture complete a word
- \S: match non-whitespace
  - r' \S(.*)\s : Will capture all characters starting non blank till blank

Re pattern – control characters (?=...)

- (?=...)
- Positive lookahead assertion. This succeeds if the contained regular expression, represented here by ..., successfully matches at the current location, and fails otherwise.
- But, once the contained expression has been tried, the matching engine doesn't advance at all; the rest of the pattern is tried right where the assertion started

```
Match = re.match(r"^(?=.*\d)(?=.*[A-Z]) (?=.*[@#$%&*]) ", password1 )
```

Compile regular expression

- re.compile('<regular expression>')

Example

p = re.compile('.* = .*')

Match = p.match("string = something")

Other re functions

- Search and replace
- .sub(*replacement*, *string*[, *count*=0])

- p = re.compile('(blue|white|red)') >>> p.sub('colour', 'blue socks and red shoes') 'colour socks and colour shoes'

Usage of regex in real life problem

Program to collect Link-ID

- Program to collect all the link ID from Router link states
- Store the output in a string

>> ospf_output = “””

“””

- Search for “Router Link State”

>>re.search(r"^(.+) Router Link State

- Skip till end of the line

>> (.*\n)*

- Skill till next non-blank line at beginning

>> ?^\S

- Now collect all till a new line with blank comes

>> (.*\n)*?^\s

- Show ip ospf database router-id

OSPF Router with ID (1.1.1.1) (Process ID 1)

Router Link States (Area 0)

Link ID	ADV Router	Age	Seq#	Checksum	Link count
1.1.1.1	1.1.1.1	20	0x80000001	0x9253	2
18.1.8.0	1.1.1.1	11	0x80000001	0x8093	3
19.1.8.0	11.11.12.13	11	0x80000001	0x8093	4
18.1.9.0	1.1.1.1	11	0x80000001	0x8093	3

Summary Net Link States (Area 0)

Link ID	ADV Router	Age	Seq#	Checksum
18.1.8.0	1.1.1.1	11	0x80000001	0x8093

Router Link States (Area 1)

Link ID	ADV Router	Age	Seq#	Checksum	Link count
1.1.1.1	1.1.1.1	20	0x80000001	0x23E1	1

```
match_string = "Router Link States"
matchObj = re.search( r"^(.+) %s (.*\n)*?^\S(.*\n)*?^\s" %match_string, ospf_output,re.M)
```

Program part-1

```
#!c:\Python27\python

    import re

    ospf_output = """  OSPF Router with ID (1.1.1.1) (Process ID 1)


                Router Link States (Area 0)

    Link ID         ADV Router      Age         Seq#       Checksum Link count
    1.1.1.1         1.1.1.1         20          0x80000001 0x9253   2
    18.1.8.0        1.1.1.1         11          0x80000001 0x8093   3
    19.1.8.0        11.11.12.13     11          0x80000001 0x8093   4
    18.1.8.0        1.1.1.1         11          0x80000001 0x8093   3

                Summary Net Link States (Area 0)

    Link ID         ADV Router      Age         Seq#       Checksum
    18.1.8.0        1.1.1.1         11          0x80000001 0x8093

                Router Link States (Area 1)

    Link ID         ADV Router      Age         Seq#       Checksum Link count
    1.1.1.1         1.1.1.1         20          0x80000001 0x23E1   1

    """

    match_string = "Router Link States“

    matchObj = re.search( r"^(.+) %s (.*\n)*?^\S(.*\n)*?^\s" %match_string,
    ospf_output ,re.M)

    if matchObj :

       next_str = matchObj.group()

       print “next_str = ", matchObj.group()

    else :

       print "no match“
```

OUTPUT

```
Next_str =          Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link
count
1.1.1.1         1.1.1.1         20          0x80000001 0x9253   2
18.1.8.0        1.1.1.1         11          0x80000001 0x8093   3
19.1.8.0        11.11.12.13     11          0x80000001 0x8093   4
18.1.8.0        1.1.1.1         11          0x80000001 0x8093   3
```

Program to collect Link-ID – part2

- Now we have Router Link States data
- We need to collect all link ID count
- We can use findall function to get all IP addressess
- Link ID are first IP address in each line
- All IP addresses are 1 to 3digit number followed by '.' followed by '.' again and so on for three time
- The three digit can be any number though it must be between 1 to 224.
- Findall function return all found string in a list.

Router Link States (Area 0)

Link ID	ADV Router	Age	Seq#	Checksum	Link count
1.1.1.1	1.1.1.1	20	0x80000001	0x9253	2
18.1.8.0	1.1.1.1	11	0x80000001	0x8093	3
19.1.8.0	11.11.12.13	11	0x80000001	0x8093	4
18.1.9.0	1.1.1.1	11	0x80000001	0x8093	3

```
matchObj = re.findall( r'([0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}) .*\n',next_str)
```

Program part-2

```
match_string = "Router Link States“

matchObj = re.search( r"^(.+) %s (.*\n)*?^\S(.*\n)*?^\s"
%match_string, ospf_output ,re.M)

if matchObj :
        next_str = matchObj.group()
        print “next_str = ", matchObj.group()
else :
        print "no match“


matchObj = re.findall( r'([0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3})
.*\n',next_str)

 i=0

 while matchObj[i]:
    print i, "->" , matchObj[i]
    i = i +1
else :
   print "No match for findall"
```

OUTPUT

```
0 -> 1.1.1.1
1 -> 18.1.8.0
2 -> 19.1.8.0
3 -> 18.1.8.0
```

Exercise

1. Write a program to find the IP address in a given string
2. Write a program to validate date.
3. Write a program to validate password.

Thanks