Object Oriented Programming

BE(CSE) II-Semester

Prepared by
S. Durga Devi,
Assistant Professor,
CSE,CBIT

Unit-4.3

Regular Expressions

S. Durga Devi ,CSE,CBIT

Regular Expressions

- Regular Expressions:
- To represent a group of strings according to a particular pattern or format.
- Formal definition is RE is a declarative mechanism to represent a group of strings according to particular format or pattern.

- **Applications of Regular Expressions**

- To develop pattern matching applications (in document search command like ctrl+f (windows) and grep in Unix etc.
- To validate a form or frame works
- To develop translators like compilers, interpreters, assemblers etc.

- In python Regular expressions can be implemented in an application using a module called ‘re’.

- ‘re’ module provides several built in classes, methods and variable to use regular expressions very easily in our application.

1. compile(): compiles pattern into regular object means which pattern to be searched is converted as regular object.

```
from re import *
pattern=compile("sai")
print(type(pattern))
```

2. finditer(): we can check how many matches are found in string.
returns iterator object which contains all the mathes found

```
Matcher=pattern.finiter(“sai raj,sai praneeth”)
```

Following are the methods in match iterator object

start() : returns starting index of the matched pattern

end(): ending index of the matched pattern(index+1) will return.

group(): returned matched pattern.

Example

```
from re import *
pattern=compile("sai")
print(type(pattern))
count=0
match=pattern.finditer("sai praneeth, basker sai, sai viswak")
for matcher in match:
    count+=1
    print(matcher.start(),"-",matcher.end(),"-",matcher.group())
print("the no of occurrences of sai is",count)
```

Out put

```
<class 're.Pattern'>
0 - 3 - sai
21 - 24 - sai
26 - 29 - sai
the no of occurrences of sai is 3
```

Note: we can pass pattern object directly as argument to finditer() function

```
from re import *
matche=finditer("sai","sai praneeth sai viswak")
for i in matche:
    print(i.start(),"-",i.group())
```

Character classes

- Character classes: used to search for group of characters.
- if you put characters to be matched in [ ] square brackets is called character classes.

Character classes	Purpose
[abc]	Search for any specified characters either a, b or c
[^abc]	Search for all the characters except abc
[a-z]	Search for any lower case alphabets
[A-Z]	Search for any upper case alphabets
[0-9]	Numbers from 0 to 9
[a-z0-9-A-Z]	Alpha numeric
[^a-zA-Z0-9]	Except alpha numeric(special characters)

```
from re import *
matcher=finditer('[abc]','a2zbA-B@$2021c')
for m in matcher:
    print(m.start(),'-',m.end(),'-',m.group())
```

```
0 - 1 - a
3 - 4 - b
13 - 14 - c
```

```
from re import *
matcher=finditer('[^abc]','a2zbA-B@$2021c')
for m in matcher:
    print(m.start(),'-',m.end(),'-',m.group())
```

```
1 - 2 - 2
2 - 3 - z
4 - 5 - A
5 - 6 - -
6 - 7 - B
7 - 8 - @
8 - 9 - $
9 - 10 - 2
10 - 11 - 0
11 - 12 - 2
12 - 13 - 1
```

```
from re import *
matcher=finditer('[a-z]','a2zbA-B@$2021c')
for m in matcher:
    print(m.start(),'-',m.end(),'-',m.group())
```

```
0 - 1 - a
2 - 3 - z
3 - 4 - b
13 - 14 - c
```

```
from re import *
matcher=finditer('[0-9]','a2zbA-B@$2021c')
for m in matcher:
    print(m.start(),'-',m.end(),'-',m.group())
```

```
1 - 2 - 2
9 - 10 - 2
10 - 11 - 0
11 - 12 - 2
12 - 13 - 1
```

Pre defined character classes

\s : search for space characters(\t,\n,\r,\f)

\S: skip space characters

\d: search for any digit

\w: any word character(alpha numeric character)

\W: any character except word(special characters)

. : every character.

```
from re import *
matcher=finditer('\s','a2zbA- B@$2021 c')
for m in matcher:
    print(m.start(),'-',m.end(),'-',m.group())
```

```
6 - 7 -
14 - 15 -
```

Quantifiers: the number of occurrences

No of occurrences of specified pattern matched.

Pattern	Descirption
a	Exactly one ‘a’
a+	At least one a(either 1 or more)
a*	Zero or more number(0 or more)
a?	Atmost 1 (0 or 1)
a{m}	Exactly m no of a’s
a{m,n}	Minimum m and maximum n
^a	Whether given string starts with ‘a’ or not
a$	Whether given string ends with ‘a’ or not

```
from re import *
matcher=finditer("a+",'aaabbabaa')
count=0
for i in matcher:
    count+=1
    print(i.start(),'-',i.group())
print(count)
```

```
0 - aaa
5 - a
7 - aa
3
```

```
from re import *
matcher=finditer("a*",'aaabbabaa')
count=0
for i in matcher:
    count+=1
    print(i.start(),'-',i.group())
print(count)
```

```
0 - aaa
3 -
4 -
5 - a
6 -
7 - aa
9 -
7
```

```
from re import *
matcher=finditer("a{1,3}",'aaabbabaaaaa')
count=0
for i in matcher:
    count+=1
    print(i.start(),'-',i.group())
print(count)
```

```
0 - aaa
5 - a
7 - aaa
10 - aa
4
```

Functions in re module

1. match()
2. fullmatch()
3. search()
4. findall()
5. finditer()
6. sub()
7. subn()
8. split()
9. compile()

1. match() : used to check whether the given string is in the beginning of target string or not.

- if match found returns match object else None will be return

Syntax:

re.match(pattern,TargetString)

Example:

```
from re import *
    pattern=input("enter your pattern")
    matched=match(pattern,"sai Kiran")
    if matched:
        print(pattern," is found in beginning of the target string")
    else:
        print("not found")
```

2. fullmatch(): both the strings are matched or not.

```
from re import *
pattern=input("enter your pattern")
matched=fullmatch(pattern,"sai kiran")
if matched:
    print(pattern," is found in beginning of the target string")
else:
    print("not found")
```

```
enter your patternsai kiran
sai kiran  is found in beginning of the target
string
```

3. search() : searches for the pattern in a target string.

- If pattern present returns match object which indicates first occurrence of the matched string else return None.

- Syntax: re.search(pattern,targetpattern)

```
from re import *
pattern=input("enter your pattern")
matched=search(pattern," i am studying B.Tech First year my name sai kiran everybody calls me sai")
if matched:
   print(pattern,"found at location",matched.start(),'and ends at',matched.end())
else:
   print("not found")
```

```
enter your patternsai
sai found at location 41 and ends at 44
```

4. findall(): finds all the occurrences of the matched pattern.

returns list object contains all the occurrences.

```
from re import *
pattern=input("enter your pattern")
matched=findall(pattern," i am studying B.Tech First year my name sai kiran everybody calls me sai")
print(matched)
```

Output:

['sai','sai']

```
from re import *
pattern="[a-zA-Z]+ \d+"
matched=findall(pattern,"LXI 2020,VDI 2021,VDI 2018, maruthi models")
for i in matched:
    print(i,end=" ")
```

5. sub(): substitute or replace a string

Syntax:

sub(source1,replacestring,target string)

All the occurrences of source1 replace with replacestring in target string.

```
# Every alphabet replace with '#' symbol
from re import *
string1=sub('[a-z]','#','a1b2c3d4')
print(string1)
```

Output:
#1#2#3#4

6. subn(): same as sub() function returns no of replacements done

-Returns tuple which consists of modified string and no of replacements

```
from re import *
string1=subn('[a-z]','#','a1b2c3d4')
print(string1)
```

Output:

7. split(): splits the given string based on the delimiter.
returns list of all the tokens

```
from re import *
string1=split(',','sunny,bunny,munny,anny,chinni,chunni')
print(string1)
```

Output:
['sunny', 'bunny', 'munny', 'anny', 'chinni', 'chunni']

programs

-Collect phone numbers from website cbit.ac.in using regular expressions

-import re,urllib
import urllib.request
u=urllib.request.urlopen(**"https://www.cbit.ac.in/"**)
text=u.read()
numbers=re.findall(**"[0-9-]{7}[0-9-]+"**,str(text),re.I)
for n in numbers:
    print(n)

-Write a program to validate email id and phone number

-Write a program to validate password.

-Write a program to check whether the given car number is registered in telangana or not.

- input is enter car registation number: TS08ED1612
- output is car registered in telangana.


CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY
1979


CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY
1979