- A regular expression (RE) in a programming language is a special text string used for describing a search pattern. It is extremely useful for extracting information from text such as code, files, log, spreadsheets, or even documents.
- Regular expressions can contain both special and ordinary characters. Most ordinary characters such as 'A', 'a', or '0' are the simplest regular expressions. These characters simply match themselves.

Regular Expression Syntax

RE

import re

- "re" module included with Python primarily used for string searching and manipulation Also used frequently for web page "Scraping" (extract large amount of data from websites)

Using regular expression methods

The "re" package provides several methods to actually perform queries on an input string. The method we going to see are

- re.match()
- re.search()
- re.findall()

Precompiled patterns

compile()

- re module provide compile () function to compile a pattern into RegexObject.

```
pattern = re.compile("ab")
```

Iterating over matches using `re.finditer`

finditer():

- You can use re.finditer to iterate over all matches in a string.
- This gives you (in comparison to re.findall extra information, such as information about the match location in the string (indexes):
- Returns an Iterator object which yields Match object for every Match

def finditer Found at: re

```
def finditer(pattern, string, flags=0):
```

- Return an iterator over all non-overlapping matches in the string. For each match, the iterator returns a Match object.

- Empty matches are included in the result."""

```
return _compile(pattern, flags).finditer(string)
```

matcher = pattern.finditer("abaababa")

On Match object we can call the following methods.

start()

Returns start index of the match

end()

Returns end+1 index of the match

group()

Returns the matched string

```
import re
count=0
pattern=re.compile("python")
matcher=pattern.finditer("python is a prog and python is High level")
for match in matcher:
    count+=1
    print(match.start(),"...",match.end(),"...",match.group())
    print("The number of occurrences: ",count)
```

```
0 ... 6 ... python
The number of occurrences:  1
21 ... 27 ... python
The number of occurrences:  2
```

Note: We can pass pattern directly as argument to finditer() function.

```
import re
count=0
matcher=re.finditer("python","python is a prog and python is High level")
for match in matcher:
    count+=1
    print(match.start(),"...",match.end(),"...",match.group())
    print("The number of occurrences: ",count)
```

```
0 ... 6 ... python
The number of occurrences:  1
21 ... 27 ... python
The number of occurrences:  2
```

Character classes:

We can use character classes to search a group of characters

Character classes:	Description
[abc]	Either 'a' or 'b' or 'c'
[^abc]	Any Character Except 'a' and 'b' and 'c'

```
import re
#RegEx pattern
pattern = '[abc]'
matcher=re.finditer(pattern,"abcdef")
for match in matcher:
    print(match.start(),"...",match.end(),"...",match.group())
```

```
0 ... 1 ... a
1 ... 2 ... b
2 ... 3 ... c
```

```
import re
#RegEx pattern
pattern = '[^abc]'
```

```
matcher=re.finditer(pattern,"abcdef")
for match in matcher:
    print(match.start(),"...",match.end(),"...",match.group())
```

```
3 ... 4 ... d
4 ... 5 ... e
5 ... 6 ... f
```

How to use range and union Regex character classes

Character classes:	Description
[a-z]	Any lower case alphabet symbol
[A-Z]	Any upper case alphabet symbol
[a-zA-Z]	Any alphabet symbol
[0-9]	Any digit from 0 to 9
[a-zA-Z0-9]	Any alphanumeric character
[^a-zA-Z0-9]	Any special character
[a-d[x-z]]	Any character that is a-d or x-z

```
import re
#RegEx pattern
pattern = '[a-z]'
matcher=re.finditer(pattern,"abcdeyAB12")
for match in matcher:
    print(match.start(),"...",match.end(),"...",match.group())
```

```
0 ... 1 ... a
1 ... 2 ... b
2 ... 3 ... c
3 ... 4 ... d
4 ... 5 ... e
5 ... 6 ... y
```

```
import re
#RegEx pattern
pattern = '[A-Z]'
matcher=re.finditer(pattern,"abcdeyABZ12")
for match in matcher:
    print(match.start(),"...",match.end(),"...",match.group())
```

```
6 ... 7 ... A
7 ... 8 ... B
8 ... 9 ... Z
```

```
import re
#RegEx pattern
pattern = '[a-zA-Z]'
matcher=re.finditer(pattern,"abcdeyABZ12")
for match in matcher:
    print(match.start(),"...",match.end(),"...",match.group())
```

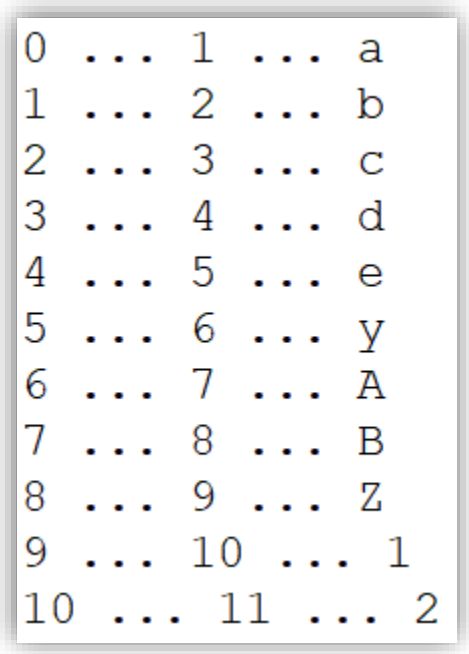
```
1 ... 2 ... b
2 ... 3 ... c
3 ... 4 ... d
4 ... 5 ... e
5 ... 6 ... y
6 ... 7 ... A
7 ... 8 ... B
8 ... 9 ... Z
```

```
import re
#RegEx pattern
pattern = '[0-9]'
matcher=re.finditer(pattern,"abcdeyABZ12")
for match in matcher:
    print(match.start(),"...",match.end(),"...",match.group())
```

```
9 ... 10 ... 1
10 ... 11 ... 2
```

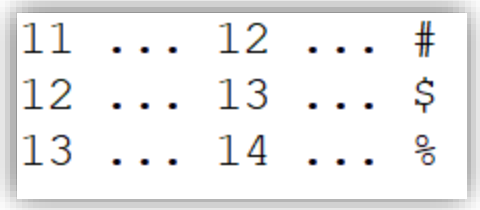
```
import re
#RegEx pattern
```

```
pattern = '[a-zA-Z0-9]'
matcher=re.finditer(pattern,"abcdeyABZ12")
for match in matcher:
    print(match.start(),"...",match.end(),"...",match.group())
```



```
0 ... 1 ... a
1 ... 2 ... b
2 ... 3 ... c
3 ... 4 ... d
4 ... 5 ... e
5 ... 6 ... y
6 ... 7 ... A
7 ... 8 ... B
8 ... 9 ... Z
9 ... 10 ... 1
10 ... 11 ... 2
```

```
import re
#RegEx pattern
pattern = '[^a-zA-Z0-9]'
matcher=re.finditer(pattern,"abcdeyABZ12#$%")
for match in matcher:
    print(match.start(),"...",match.end(),"...",match.group())
```



1. [abc]===>Either a or b or c
2. [^abc] ===>Except a and b and c
3. [a-z]==>Any Lower case alphabet symbol
4. [A-Z]===>Any upper case alphabet symbol
5. [a-zA-Z]==>Any alphabet symbol
6. [0-9] Any digit from 0 to 9
7. [a-zA-Z0-9]==>Any alphanumeric character
8. [^a-zA-Z0-9]==>Except alphanumeric characters(Special Characters)

Pre defined Character classes:

\s □ Space character

\S □ Any character except space character

\d □ Any digit from 0 to 9

\D Any character except digit

\w Any word character [a-zA-Z0-9]

\W Any character except word character (Special Characters)

. Any character including special characters

Qunatifiers:

- We can use quantifiers to specify the number of occurrences to match.

a Exactly one 'a'

a+ Atleast one 'a'

a* Any number of a's including zero number

a? Atmost one 'a' ie either zero number or one number

a{m} Exactly m number of a's

a{m,n} Minimum m number of a's and Maximum n number of a's