Regular Expression #1

- Regular Expression are useful for **defining patterns for a string** matching , it is useful for checking the conditions
- Regular Expressions or Regex is very powerful
- Suppose you want to match a pattern with a string we can use Regular Expressions
- Lets look at some patterns for strings

	Pattern	String	
1.	‘a’	‘a’	
2.	‘a | b ’	‘a’ or ‘b’	(Either a or b is True)
3.	‘abc’	‘abc’	(Exactly abc will give True)
4.	‘ [abc] ’	‘a’ or ‘b’ or ‘c’	(Either a ,b ,c will give True)

- Lets try this with an example
- To use regular Expressions we need to import re module

```
>>> from re import *
>>>
>>> match('a', 'a').group()
'a'
>>> match('a|b', 'a').group()
'a'
>>> match('a|b', 'b').group()
'b'
>>> match('abc', 'abc').group()
'abc'
>>> match('abc', 'abcd').group()
'abc'
>>> match('[abc]', 'abcd').group()
'a'
>>> match('[abc]', 'bcd').group()
'b'
>>>
```

- You can also include some characters like **+ , * , ? , {m}, {m, n}** in Regular Expression

Character	Description
+	1 or more repetitions
*	0 or more repetitions
?	0 or 1 repetitions
{ m }	Exactly m occurrences
{ m , n }	From m to n . m defaults to 0 . n to infinity

Pattern String

5. ’ [abc] + ’ a, ab, aaaa, ababab, cccc

```
>>> match('[abc]+', 'bcd').group()
'bc'
>>> match('[abc]+', 'ccccc').group()
'ccccc'
>>> match('[abc]+', 'ababcbcbbaba').group()
'ababcbcbbaba'
>>> match('[abc]{5}', 'ababcbcbbaba').group()
'ababc'
>>>
```