


```

1 #re.search(pattern, string) - searches for the first occurrence of pattern in string and returns a match object.
2 #If no match is found, it returns None.
3 import re
4
5 string = "The quick brown fox jumps over the lazy dog."
6 pattern = "brown"
7
8 match = re.search(pattern, string)
9 if match:#match==True
10     print("Match found:", match.group())
11 else:
12     print("Match not found.")
13

```

 Match not found.

```

1 #re.match(pattern, string) - similar to re.search(),
2 #but it only searches for the pattern at the beginning of the string.
3 #If no match is found, it returns None.
4 import re
5
6 string = "The quick brown fox jumps over the lazy dog."
7 pattern = "The"
8
9 match = re.match(pattern, string)
10 if match:
11     print("Match found:", match.group())
12 else:
13     print("Match not found.")
14

```

Match not found.

```

1 #re.findall(pattern, string) - searches for all non-overlapping occurrences of pattern in string and returns a list of strings.
2 #If no match is found, it returns an empty list.
3 import re
4
5 string = "The quick brown fox jumps over the lazy dog."
6 pattern = "[aeiou]"
7
8 matches = re.findall(pattern, string)#returns list
9 if matches:
10     print("Matches found:", matches)
11 else:
12     print("Matches not found.")
13 print("Total vowels are:",len(matches))

```

Matches found: ['e', 'u', 'i', 'o', 'o', 'u', 'o', 'e', 'e', 'a', 'o']
Total vowels are: 11

```

1 #re.sub(pattern, repl, string) - searches for all non-overlapping occurrences of pattern in string and replaces them with repl.
2 #Returns the modified string.
3 import re

```

```

4
5 string = "The quick brown fox jumps over the lazy dog."
6 pattern = "brown"
7
8 new_string = re.sub(pattern, "red", string)
9 print(new_string)
10

```

The quick red fox jumps over the lazy dog.

```

1 #re.split(pattern, string) - splits string into a list of substrings using pattern as the delimiter.
2 #Returns the list.
3 import re
4
5 string = "The quick      brown fox      jumps over the lazy dog."
6 pattern = "\s+"
7
8 split_string = re.split(pattern, string)
9 print(split_string)
10

```

```
['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog.']
```

Match any character: . (dot) Example: a.b matches "a b", "a1b", "a!b", etc.

Match the start of a string: ^ Example: ^hello matches "hello world", but not "world hello"

Match the end of a string: *Example* : world matches "hello world", but not "world hello"

Match any single digit: \d Example: \d matches any digit from 0 to 9

Match any non-digit character: \D Example: \D matches any character that is not a digit

Match any whitespace character: \s Example: \s matches spaces, tabs, and newlines

Match any non-whitespace character: \S Example: \S matches any character that is not whitespace

Match any word character (letters, digits, or underscore): \w Example: \w+ matches one or more word characters (letters, digits, or underscore)

Match any non-word character: \W Example: \W matches any character that is not a word character

Match zero or one occurrence of the previous character: ? Example: colou?r matches "color" and "colour"

Match zero or more occurrences of the previous character: * Example: ab*c matches "ac", "abc", "abbc", etc.

Match one or more occurrences of the previous character: + Example: ab+c matches "abc", "abbc", "abbbc", etc.

Match a specific number of occurrences of the previous character: {m} Example: a{3} matches "aaa"

Match a range of occurrences of the previous character: {m,n} Example: a{2,4} matches "aa", "aaa", and "aaaa"

Match any one of a set of characters: [] Example: [abc] matches "a", "b", or "c"

Match any character except those in a set: [^] Example: [^abc] matches any character except "a", "b", or "c"

Match the same characters as a previously captured group: \1, \2, etc. Example: (\w) is \1 matches "a is a", "b is b", etc.

These are just some of the most common regular expressions used in Python. There are many more advanced regular expressions available as well.

