Regular Expressions

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

It is a special sequence of characters that uses a search pattern to find a string or set of strings. It can detect the presence or absence of a text by matching with a particular pattern, and also can split a pattern into one or more sub-patterns.

RegEx Module

Python has a built-in package called `re`, which can be used to work with Regular Expressions.

Import the `re` module:

```
import re
```

The re module throws an exception if there is some error while using the regular expression.

RegEx Functions

The `re` module offers a set of functions that allows us to search a string for a match:

SN	Function	Description
1	match	This method matches the regex pattern in the string with the optional flag. It returns true if a match is found in the string otherwise it returns false.
2	search	This method returns the match object if there is a match found in the string. A Match Object is an object containing information

		about the search and the result.
3	findall	It returns a list that contains all the matches of a pattern in the string.
4	split	Returns a list in which the string has been split in each match.
5	sub	Replace one or many matches in the string.

Forming a regular expression

A regular expression can be formed by using the mix of meta-characters, special sequences, and sets.

Meta-Characters

Metacharacter is a character with the specified meaning.

Metacharacter	Description	Example
[]	It represents the set of characters.	"[a-z]"
\	It represents the special sequence.	"\r"
.	It signals that any character is present at some specific place.	"Ja.v."
^	It represents the pattern present at the beginning of the string.	"^Java"
$	It represents the pattern present at the end of the string.	"point"
*	It represents zero or more occurrences of a pattern in the string.	"hello*"
+	It represents one or more occurrences of a pattern in the string.	"hello+"

{}	The specified number of occurrences of a pattern the string.	"java{2}"
|	It represents either this or that character is present.	"java|point"
()	Capture and group	

Special Sequences(Character Classes)

Special sequences are the sequences containing \ followed by one of the characters.

Character	Description
\A	It returns a match if the specified characters are present at the beginning of the string.
\b	It returns a match if the specified characters are present at the beginning or the end of the string.
\B	It returns a match if the specified characters are present at the beginning of the string but not at the end.
\d	It returns a match if the string contains digits [0-9].
\D	It returns a match if the string doesn't contain the digits [0-9].
\s	It returns a match if the string contains any white space character.
\S	It returns a match if the string doesn't contain any white space character.
\w	It returns a match if the string contains any word characters.
\W	It returns a match if the string doesn't contain any word.
\Z	Returns a match if the specified characters are at the end of the string.

Sets

A set is a group of characters given inside a pair of square brackets. It represents the special meaning.

SN	Set	Description
1	[arn]	Returns a match if the string contains any of the specified characters in the set.
2	[a-n]	Returns a match if the string contains any of the characters between a to n.
3	[^arn]	Returns a match if the string contains the characters except a, r, and n.
4	[0123]	Returns a match if the string contains any of the specified digits.
5	[0-9]	Returns a match if the string contains any digit between 0 and 9.
6	[0-5][0-9]	Returns a match if the string contains any digit between 00 and 59.
10	[a-zA-Z]	Returns a match if the string contains any alphabet (lower-case or upper-case).

The findall() Function

The `findall()` function returns a list containing all matches.

Example

Print a list of all matches:

```
import re

txt = "The rain in Spain"
x = re.findall("ai", txt)
print(x)
```

```
Output      ['ai', 'ai']
```

Example:➔**Return an empty list if no match was found:**

```
import re

txt = "The rain in Spain"

#Check if "Portugal" is in the string:

x = re.findall("Portugal", txt)
print(x)

if (x):
  print("Yes, there is at least one match!")
else:
  print("No match")
```

```
[]
No match
```

The list contains the matches in the order they are found.

If no matches are found, an empty list is returned:

The search() Function

The `search()` function searches the string for a match, and returns a Match object if there is a match.

If there is more than one match, only the first occurrence of the match will be returned:

Example

Search for the first white-space character in the string:

```
import re
```

```
import re

txt = "The rain in Spain"
x = re.search("\s", txt)

print("The first white-space character is located in position:", x.start())
```

```
The first white-space character is located in position: 3
```

The split() Function

The `split()` function returns a list where the string has been split at each match:

```python
import re

#Split the string at every white-space character:

txt = "The rain in Spain"
x = re.split("\s", txt)
print(x)
```

```
['The', 'rain', 'in', 'Spain']
```

```python
import re

#Split the string at the first white-space character:

txt = "The rain in Spain"
x = re.split("\s", txt, 1)
print(x)
```

```
['The', 'rain in Spain']
```

The sub() Function

The `sub()` function replaces the matches with the text of your choice:

```python
import re

#Replace all white-space characters with the digit "9":

txt = "The rain in Spain"
x = re.sub("\s", "9", txt)
print(x)
```

```
The9rain9in9Spain
```

```python
import re

#Replace the first two occurrences of a white-space character with the digit 9:

txt = "The rain in Spain"
x = re.sub("\s", "9", txt, 2)
print(x)
```

```
The9rain9in Spain
```

Match Object

A Match Object is an object containing information about the search and the result.

Note: If there is no match, the value `None` will be returned, instead of the Match Object.

```
import re

#The search() function returns a Match object:

txt = "The rain in Spain"
x = re.search("ai", txt)
print(x)
```

```
<_sre.SRE_Match object; span=(5, 7), match='ai'>
```

The Match object has properties and methods used to retrieve information about the search, and the result:

`.span()` returns a tuple containing the start-, and end positions of the match.
`.string` returns the string passed into the function
`.group()` returns the part of the string where there was a match

Example

```
1. import re
2.
3. str = "How are you. How is everything"
4.
5. matches = re.search("How", str)
6.
7. print(matches.span())
8.
9. print(matches.group())
10.
11. print(matches.string)
```

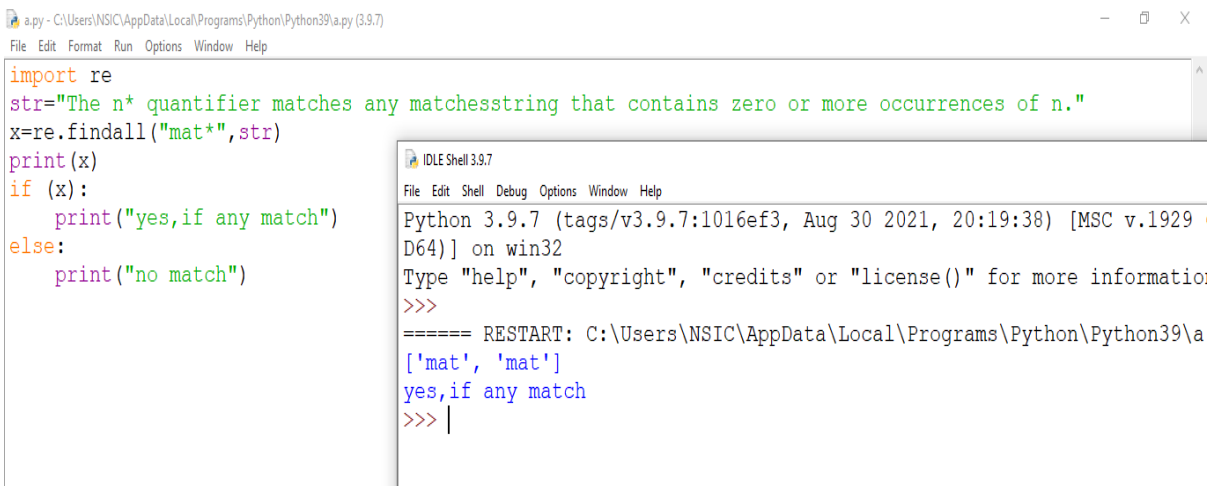
Output:

```
(0, 3)
How
How are you. How is everything
```

QUANTIFIERS:➔ In regular expressions, quantifiers match the preceding characters or character sets a number of times. The following table shows all the quantifiers and their meanings:

Quantifier	Name	Meaning
*	Asterisk	Match its preceding element zero or more times.

Quantifier	Name	Meaning
+	Plus	Match its preceding element one or more times.
?	Question Mark	Match its preceding element zero or one time.
{ *n* }	Curly Braces	Match its preceding element exactly `n` times.
{ *n* , }	Curly Braces	Match its preceding element at least `n` times.
{ *n* , *m* }	Curly Braces	Match its preceding element from `n` to `m` times.



Quantifier Table:
All the above types can be understood with this table which has the regular expression containing quantifiers and their examples.

Regex	Examples
/Ax*A/	**AA, AxA, AxxA, AxxxA, …..**
/Ax+A/	**AxA, AxxA, AxxxA, …..**
/Ax?A/	**AA, AxA**

Regex	Examples
/Ax{1, 3}A/	**AxA, AxxA, AxxxA**
/Ax{2, }A/	**AxxA, AxxxA, …..**
/Ax{4}A/	**AxxxxA**

The Dot Character:➔Dot Character represent any character (except newline character).

```
import re

txt = "hello planet"

#Search for a sequence that starts with "he", followed by two
(any) characters, and an "o":

x = re.findall("he..o", txt)
print(x)
```

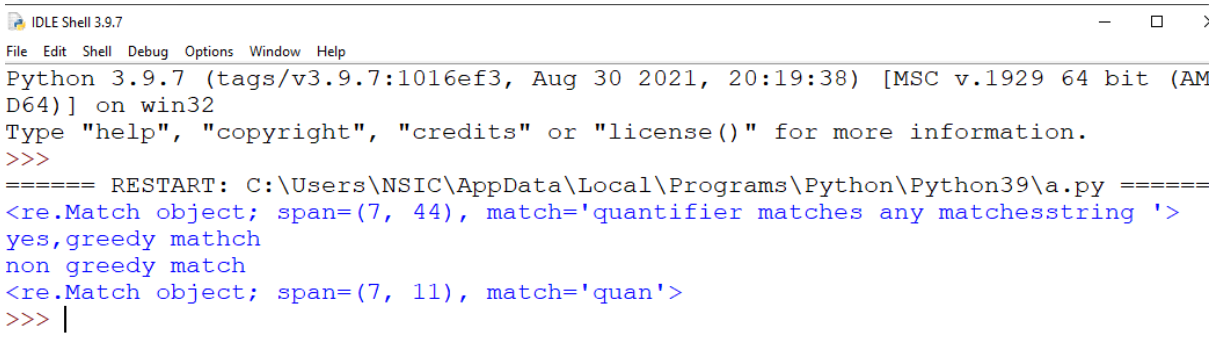
```
['hello']
```

The Greedy Matches:➔ The '*','+','?' quantifiers are all greedy ; they match as much text as possible.Adding ? after the quantifier makes it perform the match in non-greedy or minimal fashion.

a.py - C:\Users\NSIC\AppData\Local\Programs\Python\Python39\a.py (3.9.7)

File Edit Format Run Options Window Help

```
import re
str="The n* quantifier matches any matchesstring "
x=re.search("qua.*",str)
print(x)
if (x):
    print("yes,greedy mathch")
x=re.search("qua.?",str)
if x:
    print("non greedy match")
    print(x)
```



Compiling regular expressions:→
The re.compile() method

re.compile(pattern, repl, string):

We can combine a regular expression pattern into pattern objects, which can be used for pattern matching. It also helps to search a pattern again without rewriting it.

Example

```
import re

pattern=re.compile('TP')

result=pattern.findall('TP Tutorialspoint TP')

print result

result2=pattern.findall('TP is most popular tutorials site of India')

print result2
```

Output

```
['TP', 'TP']

['TP']
```