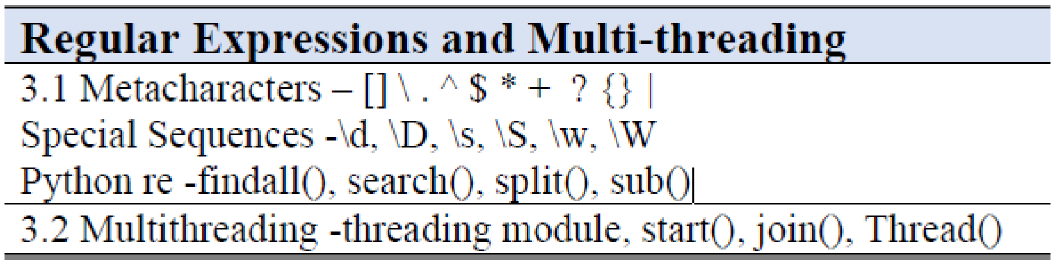
Regular Expression

Regular Expression or Regex is a vital aspect of Python programming or any other programming language. It’s used to match text strings by characters, words, and patterns. It also uses a set of characters to create a search pattern. ¶

- The language accepted by finite automata can be easily described by simple expressions called Regular Expressions. It is the most effective way to represent any language.
- The languages accepted by some regular expression are referred to as Regular languages.
- A regular expression can also be described as a sequence of pattern that defines a string.
- Regular expressions are used to match character combinations in strings. String searching algorithm used this pattern to find the operations on a string.



Regular Expressions and Multi-threading

3.1 Metacharacters – [] \ . ^ $ * + ? {} |
Special Sequences -\d, \D, \s, \S, \w, \W
Python re -findall(), search(), split(), sub()

3.2 Multithreading -threading module, start(), join(), Thread()

Metacharacters

- Metacharacters are regular expression building blocks. The following table highlights different metacharacters in Python RegEx, along with their descriptions

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"planet$"
*	Zero or more occurrences	"he.*o"
+	One or more occurrences	"he.+o"
?	Zero or one occurrence	"he.?o"
{}	Exactly the specified number of occurrences	"he.{2}o"
|	Either or	"falls|stays"
()	Capture and group	

[] Square Brackets

It is a character class with a set of characters we want to match.

- For example, the character class [abc] will match any single character a, b, or c. You can specify any range of characters as required. For example:
- [0, 3] is the same as [0123]
- [a-c] is the same as [abc].

You can invert the character class using the caret(^) symbol. For example:

- [^0-3] means any number except 0, 1, 2, or 3
- [^a-c] means any character except a, b, or c

In [1]:
```python
import re
txt = "The rain in Spain"
x = re.findall("[a-m]", txt)
print(x)
```

```
['h', 'e', 'a', 'i', 'i', 'a', 'i']
```

In [4]:
```python
import re
txt = "The rain in Spain"
x = re.findall("[^a-m]", txt)
print(x)
```

```
['T', ' ', 'r', 'n', ' ', 'n', ' ', 'S', 'p', 'n']
```

In [5]:
```python
import re
txt = "The rain in Spain"
x = re.findall("[^a-m,' ']", txt)
print(x)
```

```
['T', 'r', 'n', 'n', 'S', 'p', 'n']
```

In [7]:
```python
import re
txt = "The rain in Spain4 "
x = re.findall("[^1-4,' ']", txt)
print(x)
```

```
['T', 'h', 'e', 'r', 'a', 'i', 'n', 'i', 'n', 'S', 'p', 'a', 'i', 'n']
```

\ Backslash

- The backslash () makes sure the character is not treated in a special way. It is used to escape the metacharacters.
- For example, if you want to search for the dot(.) in the string, the searched dot will be treated as a special character. So, you need to use the backslash() just before the dot(.)

In [8]:
```python
import re
txt = "That will be 59 dollars"
#Find all digit characters:
x = re.findall("\d", txt)
print(x)
```

```
['5', '9']
```

In [14]:
```
import re
txt = "That will be .59 dollars."
#Find all digit characters:
x = re.findall("\.", txt)
print(x)
```

```
['.', '.']
```

.- Dot

- Using the dot symbol, you can match only a single character, except the newline character.
- a.b will check for the string containing any character at the place of the dot such as acb, acbd, abbb, etc
- .. will check if the string contains at least two characters

In [20]:
```
import re
txt = "hello planet hem9o he  o"
#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', 'hem9o', 'he  o']
```

^ Caret (Start With)

- Caret (^) symbol allows you to match the beginning of the string. It checks whether the string starts with the specific character(s) or not. For example:
- ^g will check if the string starts with g, such as girl, globe, gym, g, etc.
- ^ge will check if the string starts with ge, such as gem, gel, etc.

In [24]:
```
import re
txt = "hello54 hello56 planet"
#Check if the string starts with 'hello':
x = re.findall("^hello", txt)
if x:
    print("Yes, the string starts with 'hello'")
    print(x)
else:
    print("No match")
```

```
Yes, the string starts with 'hello'
['hello']
```

In [25]:
```
import re
txt = "hello54 hello65 planet"
#Check if the string starts with 'hello':
x = re.findall("^hello6", txt)
if x:
    print("Yes, the string starts with 'hello'")
    print(x)
else:
    print("No match")
```

```
No match
```

$ Dollar (end with)

- The dollar($) symbol allows you to match the end of the string. It checks whether the string ends with the given character(s) or not.
- For example:
- s$ will check for the string that ends with a such as sis, ends, s, etc.

- ks$ will check for the string that ends with ks such as seeks, disks, ks, etc.

In [26]:
```python
import re

txt = "hello planet"
#Check if the string ends with 'planet':
x = re.findall("planet$", txt)
if x:
    print("Yes, the string ends with 'planet'")
else:
    print("No match")
```

```
Yes, the string ends with 'planet'
```

In [27]:
```python
import re

txt = "hello planet5"
#Check if the string ends with 'planet':
x = re.findall("t$", txt)
if x:
    print("Yes, the string ends with 'planet'")
else:
    print("No match")
```

```
No match
```

In [28]:
```python
import re

txt = "hello planet"
#Check if the string ends with 'planet':
x = re.findall("t$", txt)
if x:
    print("Yes, the string ends with 'planet'")
else:
    print("No match")
```

```
Yes, the string ends with 'planet'
```

* Star

- This symbol will match zero or more occurrences of the regular expression precedingthe * symbol. For example:
- ab*c will be matched for the string ac, abc, abbbc, dabc, etc. but will not be matched for abdc because b is not followed by c.

In [1]:
```python
import re
txt = "hello planet"
#Search for a sequence that starts with "he", followed by 0 or more (any)
#characters, and an "o":
x = re.findall("he.*o", txt)
print(x)
```

```
['hello']
```

+ Plus

- This symbol will match one or more occurrences of the regular expression preceding the + symbol. For example:
- ab+c will be matched for the string abc, abbc, dabc, but will not be matched for ac, abdc because there is no b in ac and b is not followed by c in abdc.

In [34]:
```python
import re
txt = "hello planet"
#Search for a sequence that starts with "he", followed by 1 or more (any)
#characters, and an "o":
x = re.findall("he.+o", txt)
print(x)
```
```
['hello']
```

In [35]:
```python
import re
txt = "heo planet"
#Search for a sequence that starts with "he", followed by 1 or more (any)
#characters, and an "o":
x = re.findall("he.+o", txt)
print(x)
```
```
[]
```

In [36]:
```python
import re
txt = "heo planet"
#Search for a sequence that starts with "he", followed by 1 or more (any)
#characters, and an "o":
x = re.findall("he.*o", txt)
print(x)
```
```
['heo']
```

? Question mark

- This symbol will check if the string before the question mark occurs at least once, or not at all.
- For example:
- ab?c will match strings ac, acb, and dabc, but will not be matched for abbc because there are two bs. Similarly, it will not be matched for abdc because b is not followed by c.

In [39]:
```python
import re
txt = "hello planet"
#Search for a sequence that starts with "he", followed by 0 or 1 (any)
#character, and an "o":
x = re.findall("he.?o", txt)
print(x)
```
```
[]
```

In [40]:
```python
import re
txt = "helo planet"
#Search for a sequence that starts with "he", followed by 0 or 1 (any)
#character, and an "o":
x = re.findall("he.?o", txt)
print(x)
```
```
['helo']
```

{m,n}- Braces

- Braces match any repetitions preceding RegEx from m to n inclusive. For example:
- a{2, 4} will be matched for the string aaab, baaaac, gaad, but will not be matched for strings like abc, bc because there is only one a or no a in both the cases.

In [42]:
```python
import re
txt = "hello planet"
#Search for a sequence that starts with "he", followed excactly 2 (any)
#characters, and an "o":
x = re.findall("he.{2}o", txt)
print(x)
```
```
['hello']
```

In [43]:
```python
import re
txt = "hello planet"
#Search for a sequence that starts with "he", followed excactly 2 (any)
#characters, and an "o":
x = re.findall("he.{1}o", txt)
print(x)
```
```
[]
```

In [45]:
```python
import re
txt = "helllo planet"
#Search for a sequence that starts with "he", followed excactly 2 (any)
#characters, and an "o":
x = re.findall("he.{3}o", txt)
print(x)
```
```
['helllo']
```

| - OR

- The Or symbolt checks whether the pattern before or after the “or” symbol is present in the string or not. For example:
- a|b will match any string that contains a or b such as acd, bcd, abcd, etc.

In [47]:
```python
import re
txt = "The rain in Spain falls mainly in the plain!"
#Check if the string contains either "falls" or "stays":
x = re.findall("falls|stays", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```
```
['falls']
Yes, there is at least one match!
```

In [48]:
```python
import re
txt = "The rain in Spain falls mainly in the plain stays!"
#Check if the string contains either "falls" or "stays":
x = re.findall("falls|stays", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```
```
['falls', 'stays']
Yes, there is at least one match!
```

()- Group

- Group symbol is used to group sub-patterns.

For example:

- (a|b)cd will match for strings like acd, abcd, gacd, etc.

In [52]:
```python
import re

email = "john@example.com"
pattern = r"([a-z]+)@([a-z]+)\.com"

# Apply the pattern and extract the capture groups
match = re.match(pattern, email)
print(match.group())
if match:
    username = match.group(1)
    domain = match.group(2)
    print("Username:", username)
    print("Domain:", domain)
else:
    print("Email address is not valid")
```

```
john@example.com
Username: john
Domain: example
```

Special Sequences

Character	Description	Example
\A	Returns a match if the specified characters are at the beginning of the string	"\AThe"
\b	Returns a match where the specified characters are at the beginning or at the end of a word (the "r" in the beginning ensures the string is being treated as a "raw string")	r"\bain" r"ain\b"
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\Bain" r"ain\B"
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"
\Z	Returns a match if the specified characters are at the end of the string	"Spain\Z"

\A

Returns a match if the specified characters are at the beginning of the string

In [53]:
```python
import re
txt = "The rain in Spain"
#Check if the string starts with "The":
x = re.findall("\AThe", txt)
print(x)
if x:
    print("Yes, there is a match!")
else:
    print("No match")
```

```
['The']
Yes, there is a match!
```

In [55]:
```python
import re
txt = "The rain in Spain"
#Check if the string starts with "The":
x = re.findall("\Ara", txt)
print(x)
if x:
    print("Yes, there is a match!")
else:
    print("No match")
```

```
[]
No match
```

\b

Returns a match where the specified characters are at the beginning or at the end of a word

(the "r" in the beginning ensures the string is being treated as a "raw string")

In [56]:
```python
import re
txt = "The rain in Spain"
#Check if "ain" is present at the beginning of a WORD:
x = re.findall(r"\bain", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
[]
No match
```

In [57]:
```python
#Check if "ain" is present at the end of a WORD:
x = re.findall(r"ain\b", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
     print("No match")
```

```
['ain', 'ain']
Yes, there is at least one match!
```

In [58]:
```python
import re
txt = "The ainr in Spain"
#Check if "ain" is present at the beginning of a WORD:
x = re.findall(r"\bain", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['ain']
Yes, there is at least one match!
```

\B

Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word

(the "r" in the beginning is making sure that the string is being treated as a "raw string")

In [61]:
```python
txt = "The rain in Spain"
#Check if "ain" is present, but NOT at the beginning of a word:
x = re.findall(r"ain\B", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
[]
No match
```

In [62]:
```python
txt = "The rain in Spain"
#Check if "ain" is present, but NOT at the beginning of a word:
x = re.findall(r"\Bain", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['ain', 'ain']
Yes, there is at least one match!
```

\d

- Returns a match where the string contains digits (numbers from 0-9)

In [65]:
```python
import re
txt = "The rain in Spain"
#Check if the string contains any digits (numbers from 0-9):
x = re.findall("\d", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
[]
No match
```

In [66]:
```python
import re
txt = "The rain in Spain3"
#Check if the string contains any digits (numbers from 0-9):
x = re.findall("\d", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['3']
Yes, there is at least one match!
```

\D

Returns a match where the string DOES NOT contain digits

In [67]:
```python
import re
txt = "The rain in 33 Spain"
#Return a match at every no-digit character:
x = re.findall("\D", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['T', 'h', 'e', ' ', 'r', 'a', 'i', 'n', ' ', 'i', 'n', ' ', ' ', 'S', 'p', 'a', 'i', 'n']
Yes, there is at least one match!
```

\s

Returns a match where the string contains a white space character

In [68]:
```python
import re
txt = "The rain in Spain"
#Return a match at every white-space character:
x = re.findall("\s", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
[' ', ' ', ' ']
Yes, there is at least one match!
```

\S

Returns a match where the string DOES NOT contain a white space character

In [69]:
```python
import re
txt = "The rain in Spain"
#Return a match at every NON white-space character:
x = re.findall("\S", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['T', 'h', 'e', 'r', 'a', 'i', 'n', 'i', 'n', 'S', 'p', 'a', 'i', 'n']
Yes, there is at least one match!
```

\w

Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)

In [72]:
```
import re
txt = "The rain in Spain 35 _"
#Return a match at evry word character (characters from a to Z, digits from
# 0-9, and the underscore _ character):
x = re.findall("\w", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['T', 'h', 'e', 'r', 'a', 'i', 'n', 'i', 'n', 'S', 'p', 'a', 'i', 'n', '3', '5', '_']
Yes, there is at least one match!
```

\W

Returns a match where the string DOES NOT contain any word characters

In [74]:
```
import re
txt = "The rain in % Spain"
#Return a match at every NON word character (characters NOT between a and
#Z. Like "!", "?" white-space etc.):
x = re.findall("\W", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
[' ', ' ', ' ', '%', ' ']
Yes, there is at least one match!
```

\Z

Returns a match if the specified characters are at the end of the string

In [75]:
```
import re
txt = "Te rain in Spain"
#Check if the string ends with "Spain":
x = re.findall("Spain\Z", txt)
print(x)
if x:
    print("Yes, there is a match!")
else:
    print("No match")
```

```
['Spain']
Yes, there is a match!
```

Sets

- This is a set of characters enclosed in square brackets [] with a special meaning.

[arn]

This will return a match where one of the specified characters (a, r, or n) are present.

In [76]:
```python
import re
txt = "The rain in Spain"
#Check if the string has any a, r, or n characters:
x = re.findall("[arn]", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['r', 'a', 'n', 'n', 'a', 'n']
Yes, there is at least one match!
```

[a-n]

This will return a match for any lower case character, alphabetically between a and n.

In [77]:
```python
import re
txt = "The rain in Spain"
#Check if the string has any characters between a and n:
x = re.findall("[a-n]", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['h', 'e', 'a', 'i', 'n', 'i', 'n', 'a', 'i', 'n']
Yes, there is at least one match!
```

[^arn]

This will return a match for any character EXCEPT a, r, and n.

In [78]:
```python
import re
txt = "The rain in Spain"
#Check if the string has other characters than a, r, or n:
x = re.findall("[^arn]", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['T', 'h', 'e', ' ', 'i', ' ', 'i', ' ', 'S', 'p', 'i']
Yes, there is at least one match!
```

[0123]

This will return a match where any of the specified digits (0, 1, 2, or 3) are present.

In [79]:

```python
import re
txt = "The rain in Spain"
#Check if the string has any 0, 1, 2, or 3 digits:
x = re.findall("[0123]", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
[]
No match
```

In [80]:

```python
import re
txt = "The rain3 in Spain"
#Check if the string has any 0, 1, 2, or 3 digits:
x = re.findall("[0123]", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['3']
Yes, there is at least one match!
```

[0-9]

This will return a match for any digit between 0 and 9.

In [81]:

```python
import re
txt = "8 times before 11:45 AM"
#Check if the string has any digits:
x = re.findall("[0-9]", txt)
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['8', '1', '1', '4', '5']
Yes, there is at least one match!
```

[0-5][0-9]

This will return a match for any two-digit numbers from 00 and 59.

In [83]:

```python
import re
txt = "8 times before 11:45 AM"
#Check if the string has any two-digit numbers, from 00 to 59:
x = re.findall("[0-5][0-9]", txt)
print(x)
```

```
['11', '45']
```

[a-zA-Z]

This will return a match for any character alphabetically between a and z, lower case OR upper case.

In [85]:
```
import re
txt = "8 times before 11:45 AM"
#Check if the string has any characters from a to z lower case, and A to Z
#upper case:
x = re.findall("[a-zA-Z]", txt)
print(x)
```

```
['t', 'i', 'm', 'e', 's', 'b', 'e', 'f', 'o', 'r', 'e', 'A', 'M']
```

[+]

In sets, +, *, ., |, (), $,{} has no special meaning. So, [+] means: return a match for any + character in the string.

In [88]:
```
import re
txt = "8 times before+ 11:45 AM"
#Check if the string has any + characters:
x = re.findall("[+]", txt)
print(x)

```

```
['+']
```

Regex Module in Python

Python comes with a module named ‘re’. You must have noticed in the above example where we have imported the module ‘re’.

findall() function

This is a built-in function of the ‘re;’ module that handles the regular expression.

Syntax:

re.findall(pattern, string, flags=0)

Pattern is the regular expression.

- String is the input string provided by the user.
- Flags are used to modify the standard pattern behavior.

Each string is evaluated from left to right and finds all the matches of the pattern within the string. However, the result depends on the pattern.

- If the pattern has no capturing groups, the findall() function returns a list of strings that match the whole pattern.
- If the pattern has one capturing group, the findall() function returns a list of strings that match the group.
- If the pattern has multiple capturing groups, the findall() function returns the tuples of strings that match the groups.
- It’s important to note that the non-capturing groups do not affect the form of the return result.

In [89]:
```
import re
s = "black, blue and brown"
pattern = r'bl\w+'
matches = re.findall(pattern,s)
print(matches)
```

```
['black', 'blue']
```

• Pattern with a single group

In [90]:
```python
import re
s = "black, blue and brown"
pattern = r'bl(\w+)'
matches = re.findall(pattern,s)
print(matches)
```

```
['ack', 'ue']
```

• Pattern with multiple groups

In [91]:
```python
import re
s = "black, blue and brown"
pattern = r'(bl(\w+))'
matches = re.findall(pattern,s)
print(matches)
```

```
[('black', 'ack'), ('blue', 'ue')]
```

• Using regular expression flag

In [92]:
```python
import re
s = "Black, blue and brown"
pattern = r'(bl(\w+))'
matches = re.findall(pattern, s, re.IGNORECASE)
print(matches)
```

```
[('Black', 'ack'), ('blue', 'ue')]
```

Search() Function

- The search() function scans the string from left to right and finds the first location where the pattern produces a match. It returns a Match object if the search was successful or None otherwise.
- re.search(pattern, string, flags=0)

● Pattern is the regular expression.

● String is the input string provided by the user.

● Flags are used to modify the standard pattern behavior of the pattern.

• Finding the first match

In [2]:
```python
import re
s = 'Python 3 was released on Dec 3, 2008'
pattern = '\d+'
match = re.search(pattern, s)
if match is not None:
    print(match.group())
else:
    print('No match found')
```

```
3
```

• Finding the first word matching the pattern

In [4]:
```python
import re
s = 'CPython, IronPython, or Cython'
pattern = r'\b((\w+)thon)\b'
match = re.search(pattern, s)
if match is not None:
    print(match.groups())
```

```
('CPython', 'CPy')
```

The pattern r'\b((\w+)thon)\b' has two capturing groups:

- `(\w+) - captures the characters at the beginning of the word.`
- `((\w+)thon) - captures the whole word.`

Sub() Function

This function of the re module allows you to handle the regular expression.

Syntax:

```
re.sub(pattern, repl, string, count=0, flags=0)
```

- `Pattern is a regular expression or Pattern object.`
- `Repl is the replacement.`
- `String is the input string provided by the user.`
- `Count parameter specifies the maximum number of matches that the sub() function should replace. If you pass zero or skip it, the sub()function will replace all the matches.`
- `Fags is one or more RegEx flags to modify the standard pattern behavior.`

It will search for the pattern in the string and replace the matched strings with the replacement (repl). If the sub() function couldn't find a match, it will return the original string. Otherwise, the sub()function returns the string after replacing the matches.

• To turn the phone number (212)-456-7890 into 2124567890

In [99]:
```python

import re
phone_no = '(212)-456-7890'
pattern = '\D'
result = re.sub(pattern, '',phone_no)
print(result)
```

```
2124567890
```

• To replace the leftmost non-overlapping occurrences of a pattern

In [101]:
```python
import re
pattern = '00'
s = '00000'
result = re.sub(pattern,'',s)
print(result)
```

```
0
```

• Backreference example

In [102]:
```python
import re
s = 'Make the World a *Better Place*'
pattern = r'\*(.*?)\*'
replacement = r'<b>\1<\\b>'
html = re.sub(pattern, replacement, s)
print(html)
```

```
Make the World a <b>Better Place<\b>
```

Split() Function

```
    It splits a string by the matches of a regular expression.

    Syntax:

        split(pattern, string, maxsplit=0, flags=0)

        • Pattern is the regular expression.

        • String is the input string provided by the user.

        • Flag is optional and by default is zero. It accepts one or more RegEx flags. The fl
    ags parameter changes how the RegEx engine matches the pattern.

        • maxsplit determines at most the splits occur. Generally, if the maxsplit is one, th
    e
```

resulting list will have two elements. If the maxsplit is two, the resulting list will have three elements, and so on.

• To split the words in a sentence

In [106]:
```python
import re
s = 'A! B. C D'
pattern = r'\W+'
l = re.split(pattern, s)
print(l)
```

```
['A', 'B', 'C', 'D']
```

• With a capturing group

In [107]:
```python
import re
s = 'A! B. C D'
pattern = r'(\W+)'
l = re.split(pattern, s, 2)
print(l)
```

```
['A', '! ', 'B', '. ', 'C D']
```

practice

Matching a phone number:

In [43]:
```python
import re

phone_pattern = r'\d{3}-\d{3}-\d{4}'
phone_string = 'Call me at 555-123-4567'

match = re.search(phone_pattern, phone_string)
if match:
    print('Phone number found:', match.group())
else:
    print('Phone number not found')

```

```
Phone number found: 555-123-4567
```

In [44]:
```python
import re

phone_pattern = r'\d{3}-\d{3}-\d{4}'
phone_string = 'Call me at 555-123-4567'

match = re.findall(phone_pattern, phone_string)
match
```

Out[44]:
```
['555-123-4567']
```

Matching an email address:

In [48]:
```python
import re

email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
email_string = 'My email is john32_.doe@example.com'

match = re.search(email_pattern, email_string)
if match:
    print('Email address found:', match.group())
else:
    print('Email address not found')

```

```
Email address found: john32_+.doe@example.com
```

Matching a URL:

In [49]:

```python
import re

url_pattern = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))
url_string = 'Check out this site: https://www.example.com'

match = re.search(url_pattern, url_string)
if match:
    print('URL found:', match.group())
else:
    print('URL not found')

```

```
URL found: https://www.example.com (https://www.example.com)
```

Matching a date:

In [50]:

```python
import re

date_pattern = r'\d{1,2}[-/]\d{1,2}[-/]\d{4}'
date_string = 'The meeting is on 12/31/2022'

match = re.search(date_pattern, date_string)
if match:
    print('Date found:', match.group())
else:
    print('Date not found')

```

```
Date found: 12/31/2022
```

Matching a time:

In [51]:

```python
import re

time_pattern = r'\d{1,2}:\d{2}([ap]m)?'
time_string = 'The appointment is at 2:30pm'

match = re.search(time_pattern, time_string)
if match:
    print('Time found:', match.group())
else:
    print('Time not found')

```

```
Time found: 2:30pm
```

Matching a social security number:

In [52]:

```python
import re

ssn_pattern = r'\d{3}-\d{2}-\d{4}'
ssn_string = 'My SSN is 123-45-6789'

match = re.search(ssn_pattern, ssn_string)
if match:
    print('SSN found:', match.group())
else:
    print('SSN not found')

```

```
SSN found: 123-45-6789
```

Matching an IP address:

In [53]:
```python
import re

ip_pattern = r'\b(?:\d{1,3}\.){3}\d{1,3}\b'
ip_string = 'The server IP address is 192.168.0.1'

match = re.search(ip_pattern, ip_string)
if match:
    print('IP address found:', match.group())
else:
    print('IP address not found')

```

```
IP address found: 192.168.0.1
```

Matching a credit card number:

In [54]:
```python
import re

cc_pattern = r'\b(?:\d{4}[ -]?){3}(?:\d{4})\b'
cc_string = 'My credit card number is 1234-5678-9012-3456'

match = re.search(cc_pattern, cc_string)
if match:
    print('Credit card number found:', match.group())
else:
    print('Credit card number not found')

```

```
Credit card number found: 1234-5678-9012-3456
```

Matching a hex color code:

In [55]:
```python
import re

color_pattern = r'#[a-fA-F0-9]{6}\b'
color_string = 'The background color is #FFA500'

match = re.search(color_pattern, color_string)
if match:
    print('Color code found:', match.group())
else:
    print('Color code not found')

```

```
Color code found: #FFA500
```

Search for the word "apple" in a string:

In [56]:
```python
import re

string = 'I like apples'
pattern = r'apple'

match = re.search(pattern, string)
if match:
    print('Word found')
else:
    print('Word not found')

```

```
Word found
```

Write a regular expression to match any word that starts with 'a' and ends with 'e'.

In [5]:
```python
import re

pattern = r'\ba\w*e\b'
string = 'The apple tree is very old and has no leaves.'

match = re.findall(pattern, string)
print(match)

```

```
['apple']
```

Search for a string of digits in a sentence:

In [57]:
```python
import re

string = 'I have 3 cats and 2 dogs'
pattern = r'\d+'

match = re.search(pattern, string)
if match:
    print('Digits found:', match.group())
else:
    print('Digits not found')

```

```
Digits found: 3
```

Search for a sequence of characters containing "a", "b", and "c" in any order:

In [61]:
```python
import re

string = 'The quick brown fox jumps over the lazy dog abc bca cba'
pattern = r'^(?=.*a)(?=.*b)(?=.*c)[abc]+$'

match = re.search(pattern, string)
if match:
    print('Sequence found:', match.group())
else:
    print('Sequence not found')

```

```
Sequence not found
```

In [63]:
```python
import re

string = 'abc'
pattern = r'^(?=.*a)(?=.*b)(?=.*c)[abc]+$'

match = re.search(pattern, string)
if match:
    print('Sequence found:', match.group())
else:
    print('Sequence not found')
```

```
Sequence found: abc
```

Split a string of numbers and commas into a list of integers:

In [ ]:
```python
import re

string = '1,2,3,4,5'
numbers = [int(n) for n in re.split(r',', string)]

print(numbers)

```

Split a string of words into a list of words with at least three characters:

In [67]:
```python
import re

string = 'The quick brown fox jumps over ka the lazy dog'
words = re.split(r'\b\w{3}\b', string)

print(words)
```

```
['', ' quick brown ', ' jumps over ka ', ' lazy ', '']
```

Split a string into a list of sentences:

In [68]:
```python
import re

string = 'The first sentence. The second sentence? The third sentence!'
sentences = re.split(r'[.?!]', string)

print(sentences)

```

```
['The first sentence', ' The second sentence', ' The third sentence', '']
```

Replace all occurrences of "apple" with "orange" in a string:

In [69]:
```python
import re

string = 'I like apples'
new_string = re.sub(r'apple', 'orange', string)

print(new_string)

```
```
I like oranges
```

Replace all digits in a string with "x":

In [70]:
```python
import re

string = 'I have 3 cats and 2 dogs'
new_string = re.sub(r'\d', 'x', string)

print(new_string)
```
```
I have x cats and x dogs
```

Replace all occurrences of "cat" or "dog" with "pet" in a string:

In [71]:
```python
import re

string = 'I have 3 cats and 2 dogs'
new_string = re.sub(r'cat|dog', 'pet', string)

print(new_string)

```
```
I have 3 pets and 2 pets
```

Remove all non-alphanumeric characters from a string

In [72]:
```python
import re

string = 'The quick brown fox jumps over the lazy dog!'
new_string = re.sub(r'\W+', '', string)

print(new_string)

```
```
Thequickbrownfoxjumpsoverthelazydog
```

Write a regular expression to match any string that contains only letters (both uppercase and lowercase) and spaces.

In [ ]:
```python

```

In [13]:
```python
import re

pattern = r'^[A-Za-z\s]+$'
string =  'This Is Valid String'

match = re.search(pattern, string)
print(match)
print(bool(match))

```

```
<re.Match object; span=(0, 20), match='This Is Valid String'>
True
```

In [10]:
```python
import re

pattern = r'^[A-Za-z\s]+$'
string = "Hello World"

match = re.search(pattern, string)
print(match)
print(bool(match))
```

```
<re.Match object; span=(0, 11), match='Hello World'>
True
```

Write a regular expression to match any word that starts with a capital letter.

In [14]:
```python
import re

pattern = r'\b[A-Z]\w*\b'
string = 'The Quick brown fox jumps over the Lazy dog.'

match = re.findall(pattern, string)
print(match)
```

```
['The', 'Quick', 'Lazy']
```

Write a regular expression to match any string that contains at least one digit.

In [16]:
```python
import re

pattern = r'\d+'
string = 'There are 7 days in a week.'

match = re.findall(pattern, string)
print(match)
print(bool(match))
```

```
['7']
True
```

Write a regular expression to match any string that starts with 'Hello' and ends with 'world'.

In [18]:
```python
import re

pattern = r'^Hello.*world$'
string = 'Hello, how are you? I hope you are doing well. Hello world'

match = re.findall(pattern, string)
print(match)
```

```
['Hello, how are you? I hope you are doing well. Hello world']
```

Write a regular expression to match any string that contains a sequence of two or more vowels.

In [21]:
```python
import re

pattern = r'[aeiou]{2}'
string = 'The quick brown fox jumps over the lazy dog.'

match = re.findall(pattern, string)
print(match)

```

```
['ui']
```

Write a regular expression to match any string that contains a sequence of two or more consonants

In [22]:
```python
import re

pattern = r'[^aeiou\s]{2}'
string = 'The quick brown fox jumps over the lazy dog.'

match = re.findall(pattern, string)
print(match)
```

```
['Th', 'ck', 'br', 'wn', 'mp', 'th', 'zy', 'g.']
```

Write a regular expression to match any string that starts with a number and ends with a letter.

In [26]:
```python
import re

pattern = r'^\d.*[a-zA-Z]$'
string = '7apples , 3bananas, and 5oranges'

match = re.findall(pattern, string)
print(match)
```

```
['7apples , 3 bananas, and 5 oranges']
```

Write a regular expression to match any string that contains exactly 3 digits.

In [1]:
```python
import re

pattern = r'\d{3}'
string = 'The price of the product is 1234 dollars.'

match = re.findall(pattern, string)
print(match[0])

```
```
123
```

Write a regular expression to match any string that contains a sequence of two or more spaces.

In [29]:
```python
import re

pattern = r'\s{2,}'
string = 'The  quick  brown  fox  jumps  over  the  lazy  dog.'

match = re.findall(pattern, string)
print(match)

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

Write a regular expression to match any string that contains a sequence of four consecutive digits.

In [[illegible]8]:
```python
import re

pattern = r'\d{4}'
string = 'The year 2022 is goi2021ng to be a great year!'

match = re.findall(pattern, string)
print(match)
```
```
['2022', '2021']
```

Write a regular expression to match any string that contains a sequence of at least three consecutive vowels.

In [[illegible]0]:
```python
import re
pattern = r'[aeiou]{3,}'
string = 'The quiack brown fox jumps over the lazy dog.'
match = re.findall(pattern, string)
print(match)

```
```
['uia']
```

Write a regular expression to match any string that contains a sequence of at least two consecutive words that start with the same letter.

In [42]:
```python
import re

pattern = r'\b(\w)\w*\b\s+\b\1\w*\b'
string = 'The quick qbrown fox jumps over the lazy dog.'

match = re.findall(pattern, string)
print(match)

```
```
['q']
```