Big Data Analytics
Structuring with Regex

Muhammad Affan Alim

Regex for Structuring (Data Wrangling)

What is Feature Engineering - Introduction

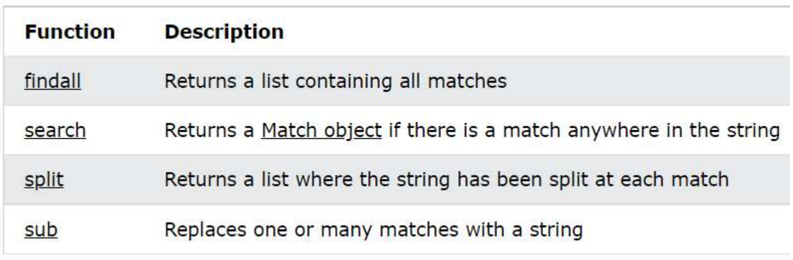
- The regular expression language is relatively small and restricted, so not all possible string processing tasks can be done using regular expressions

Match Characters

- Some characters are special metacharacters, and don’t match themselves. Instead, they signal that some out-of-the-ordinary thing should be matched, or they affect other portions of the RE by repeating them or changing their meaning. Much of this document is devoted to discussing various metacharacters and what they do.

Regex Function

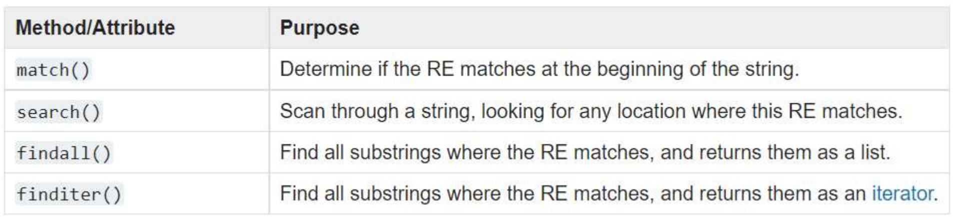
- The re module offers a set of functions that allows us to search a string for a match:



Function	Description
findall	Returns a list containing all matches
search	Returns a Match object if there is a match anywhere in the string
split	Returns a list where the string has been split at each match
sub	Replaces one or many matches with a string

Module Contents

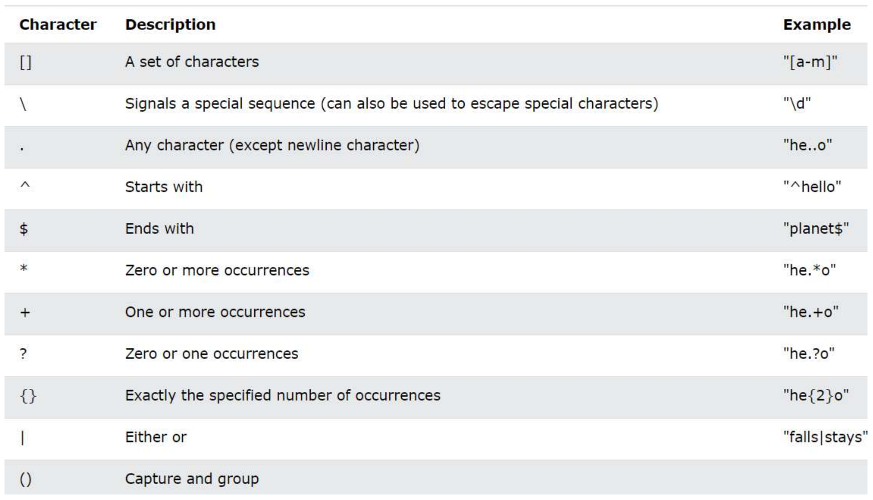
-



Method/Attribute	Purpose
`match()`	Determine if the RE matches at the beginning of the string.
`search()`	Scan through a string, looking for any location where this RE matches.
`findall()`	Find all substrings where the RE matches, and returns them as a list.
`finditer()`	Find all substrings where the RE matches, and returns them as an iterator.

Metacharacters

- Metacharacters are characters with a special meaning:



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 occurrences	"he.?o"
{}	Exactly the specified number of occurrences	"he{2}o"
|	Either or	"falls|stays"
()	Capture and group	

Metacharacters

- [ ]

```
>> import re
>> txt = "The rain in Spain"
>> #Find all lower case characters alphabetically between "a" and "m":
>> x = re.findall("[a-m]", txt)
>> print(x)
```

Metacharacters

- \

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

Metacharacters

- .

```
>> 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)
Output: ['hello']
```

Metacharacters

- ^

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

Metacharacters

- $

```
>> 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")
```

Metacharacters

- *

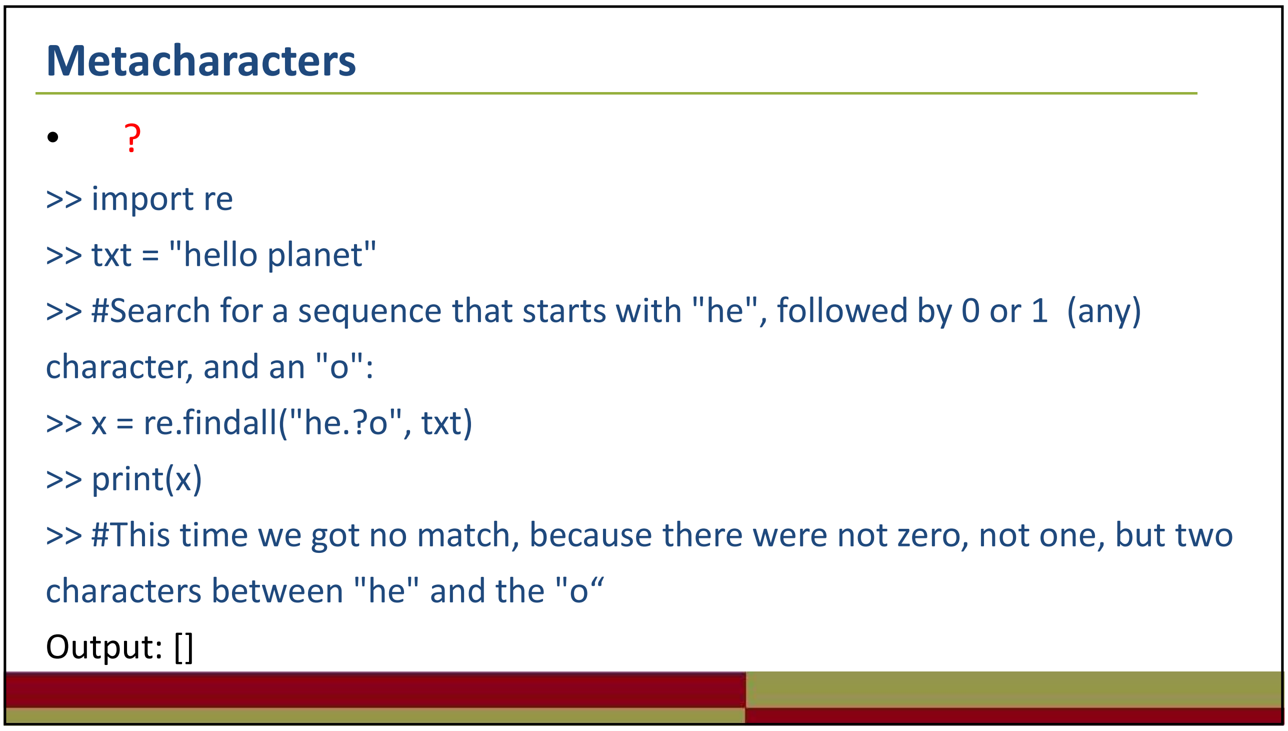
```
>> import re
>> txt = "hello planet"
>> #Search for a sequence that starts with "he", followed by 0 or more  (any)
>> characters, and an "o":
>> x = re.findall("he.*o", txt)
>> print(x)
```

Metacharacters

- +

```
>> 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)
```

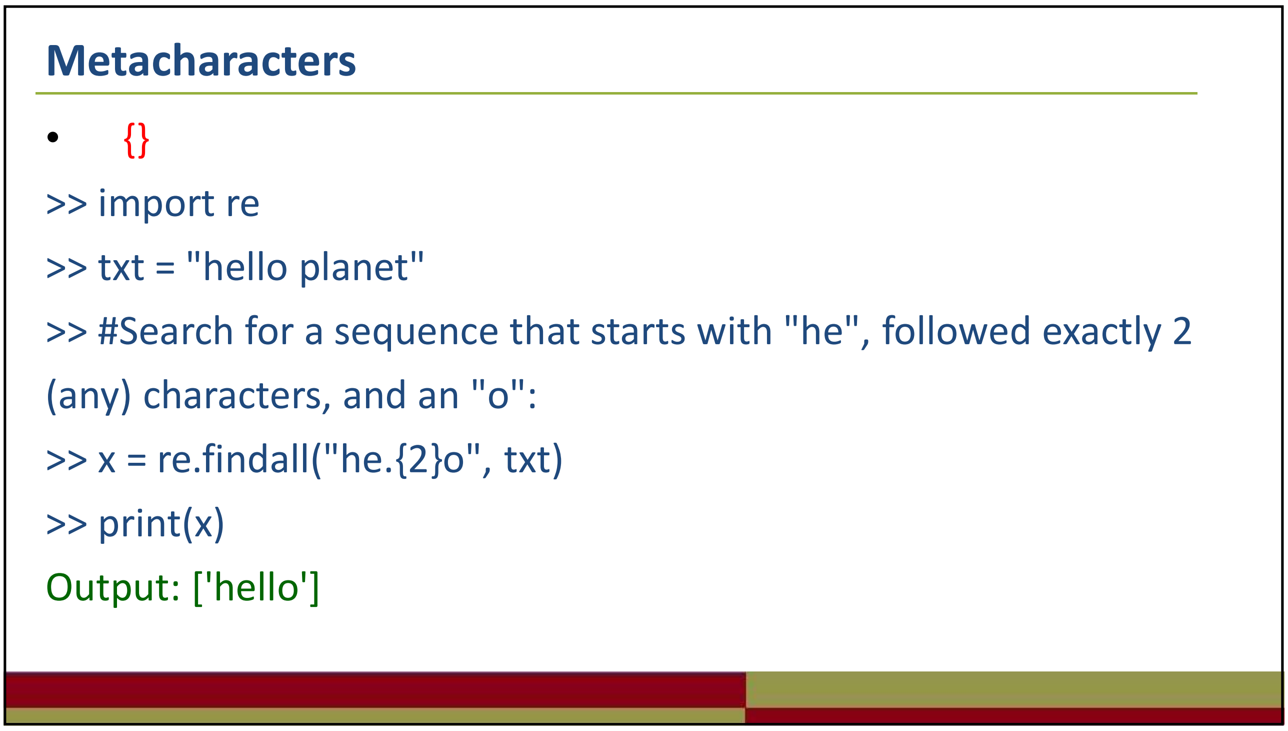
- Output: ['hello']



Metacharacters

- ?

```
>> 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)
>> #This time we got no match, because there were not zero, not one, but two
characters between "he" and the "o“
Output: []
```



Metacharacters

- {}

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

Metacharacters

- |

```
>> 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")
```

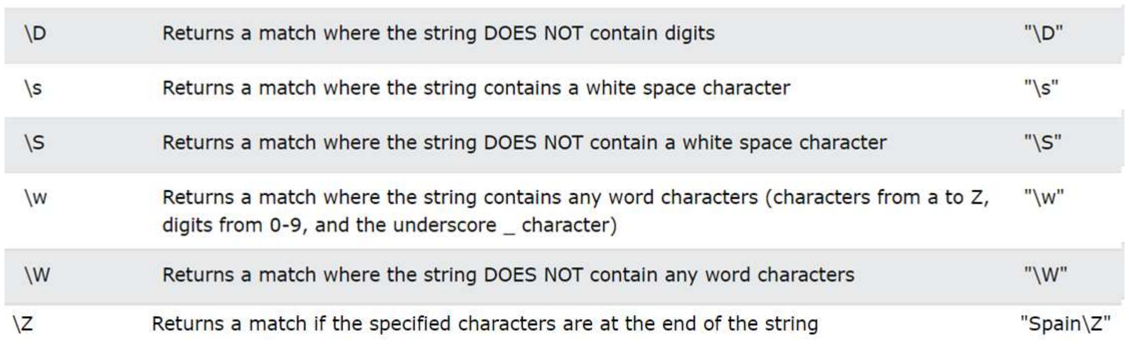
Special Sequences

- A special sequence is a \ followed by one of the characters in the list below, and has a special meaning:

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 is making sure that 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"

Special Sequences

- A special sequence is a \ followed by one of the characters in the list below, and has a special meaning:



\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"

Special Sequences

- \A

```
>> 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")
```

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

Special Sequences

- \b

```
>> import re
>> txt = "The rain in Spain"
>> #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")
```

Special Sequences

- \b

```
>> 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")
```

Special Sequences

- \B

```
>> import re
>> 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")
```

Special Sequences

- \d

```
>> 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")
```

Special Sequences

- \D

```
>> import re
>> txt = "The rain in 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")
```

Special Sequences

- \s

```
>> 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")
```

Special Sequences

- \S

```
>> 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")
```

Special Sequences

- \w

```
>> import re
>> txt = "The rain in Spain"
>> #Return a match at every 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")
```

Special Sequences

- \W

```
>> 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")
```

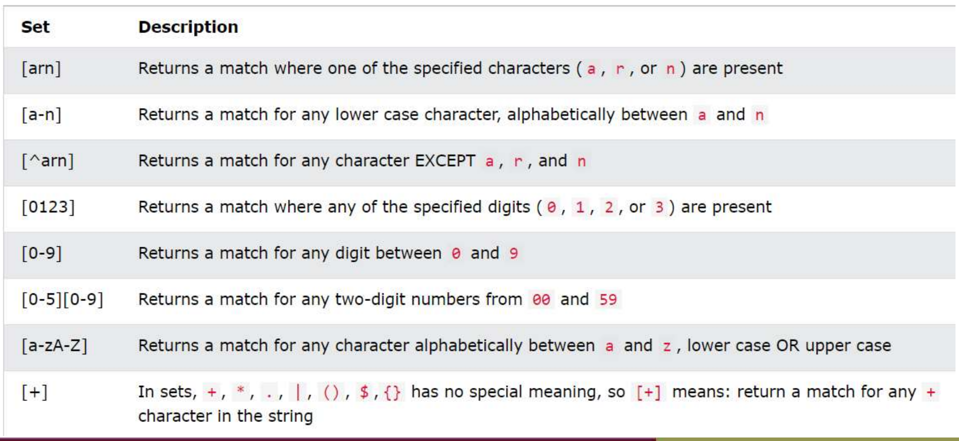
Special Sequences

- \Z

```
>> import re
>> txt = "The 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")
```

Set

- A set is a set of characters inside a pair of square brackets [] with a special meaning:



Set	Description
[arn]	Returns a match where one of the specified characters (a , r , or n) are present
[a-n]	Returns a match for any lower case character, alphabetically between a and n
[^arn]	Returns a match for any character EXCEPT a , r , and n
[0123]	Returns a match where any of the specified digits (0 , 1 , 2 , or 3) are present
[0-9]	Returns a match for any digit between 0 and 9
[0-5][0-9]	Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]	Returns a match for any character alphabetically between a and z , lower case OR upper case
[+]	In sets, + , * , . , | , () , $,{} has no special meaning, so [+] means: return a match for any + character in the string

Set

- [arn]

```
>> 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")
```

Output:

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

Set

- [a-n]

```
>> 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")
```

Output:

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

Set

- [^arn]

```
>> 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")
```

Output:

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

Set

- [123]

```
>> 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")
```

```
Output:
[]
No match
```

Set

- [0-9]

```
>> 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")
```

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

Set

- [0=5][0-9]

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

```
Output:
['t', 'i', 'm', 'e', 's', 'b', 'e', 'f', 'o', 'r', 'e', 'A', 'M']
Yes, there is at least one match!
```

Set

- [a-z][A-Z]

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

```
Output:
['t', 'i', 'm', 'e', 's', 'b', 'e', 'f', 'o', 'r', 'e', 'A', 'M']
Yes, there is at least one match!
```

Set

- +

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

```
Output:
[]
No match
```

Python RegEx

- Python has a module named **re** to work with regular expressions. To use it, we need to import the module.
- import re
- The module defines several functions and constants to work with RegEx.

Python RegEx

- **re.split()**
- The re.split method splits the string where there is a match and returns a list of strings where the splits have occurred.

Python RegEx

- **Example 1: re.split()**

```
>> import re
>> string = 'Twelve:12 Eighty nine:89.'
>> pattern = '\d+'
>> result = re.split(pattern, string)
>> print(result)
```

- # Output: ['Twelve:', ' Eighty nine:', '.']
- If the pattern is not found, re.split() returns a list containing the original string.

Python RegEx

- **re.sub()**
- The syntax of re.sub() is:

- re.sub(pattern, replace, string)
- The method returns a string where matched occurrences are replaced with the content of replace variable.

Python RegEx

- **Example 1: re.sub()**

```
>> # Program to remove all whitespaces
>> import re
>> # multiline string
>> string = 'abc 12\
>> de 23 \n f45 6'

>> # matches all whitespace characters
>> pattern = '\s+'
```

Python RegEx

You can pass count as a fourth parameter to the re.sub() method. If omitted, it results to 0. This will replace all occurrences.

```
>> import re

>> # multiline string
>> string = 'abc 12\de 23 \n f45 6'
```

Python RegEx

- **re.subn()**
- The re.subn() is similar to re.sub() except it returns a tuple of 2 items containing the new string and the number of substitutions made.
- **Example 4: re.subn()**

```
>> # Program to remove all whitespaces
>> import re
>> # multiline string
>> sstring = 'abc 12\de 23 \n f45 6'
```

Python RegEx

```
>> # matches all whitespace characters
>> pattern = '\s+'

>> # empty string
>> replace = ''

>> new_string = re.subn(pattern, replace, string)
>> print(new_string)

>> # Output: ('abc12de23f456', 4)
```

Python RegEx

- **re.search()**
- The re.search() method takes two arguments: a pattern and a string. The method looks for the first location where the RegEx pattern produces a match with the string.

- If the search is successful, re.search() returns a match object; if not, it returns None.

- match = re.search(pattern, str)

Python RegEx

- **Example 1: re.search()**

```
>> import re
>> string = "Python is fun"
>> # check if 'Python' is at the beginning
>> match = re.search('\APython', string)

>> if match:
>> print("pattern found inside the string")
>> else:
>> print("pattern not found")
```

- # Output: pattern found inside the string

Python RegEx

- **Match object**

- You can get methods and attributes of a match object using dir() function.
- Some of the commonly used methods and attributes of match objects are:
- match.group()
- The group() method returns the part of the string where there is a match.

Python RegEx

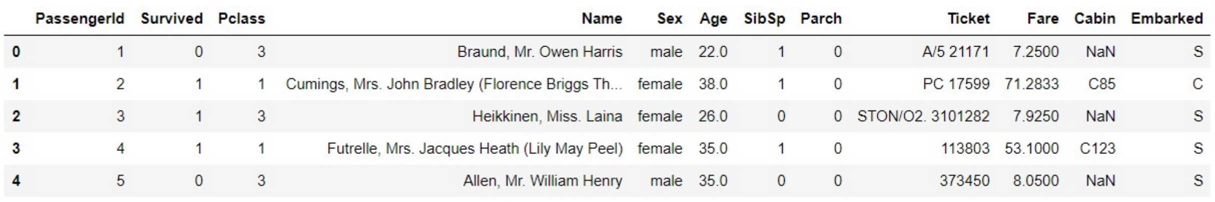
- **Example 2: Match object**

```
>> import re
>> string = '39801 356, 2102 1111'
>> # Three digit number followed by space followed by two digit number
>> pattern = '(\d{3}) (\d{2})'
>> # match variable contains a Match object.
>> match = re.search(pattern, string)
```

Example of Regex

Example - Regex

- Import pandas as pd
- Import numpy as np

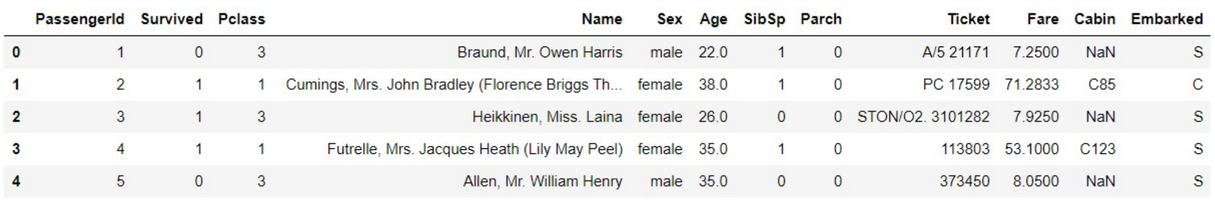


	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

- df = pd.read_csv(‘titanic_train.csv’)
- df[‘Title’] = df[‘Name’].str.extract(‘([A-Za-z]+\.)’,expand=False)
- df[‘Age’].fillna(df.groupby(‘title)[‘Age’].transform(‘mean’), inplace = True)

Example 1 - Regex

- Import pandas as pd
- Import numpy as np

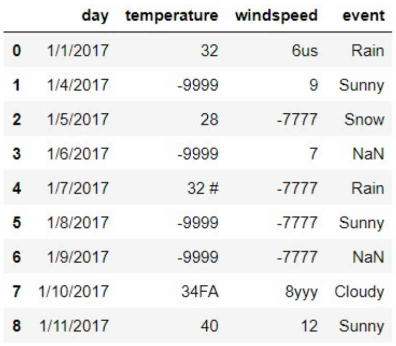


	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

- df = pd.read_csv(‘titanic_train.csv’)
- df[‘Title’] = df[‘Name’].str.extract(‘([A-Za-z]+\.)’,expand=False)
- df[‘Age’].fillna(df.groupby(‘title)[‘Age’].transform(‘mean’), inplace = True)

Example 2 - Regex

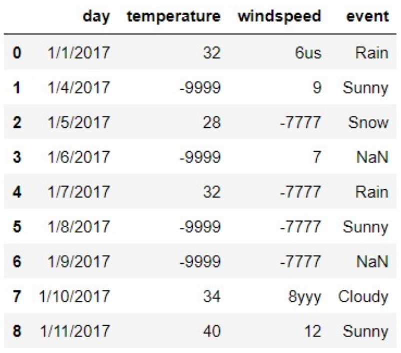
- **Remove the unnecessary characters from columns**
- import pandas as pd
- import numpy as np
- dfW = pd.read_csv('D:\\Teaching Subject\\Data Science\\Fall 2021\\Lectures\\Structuring and Regex Example\\weather_data.csv')



	day	temperature	windspeed	event
0	1/1/2017	32	6us	Rain
1	1/4/2017	-9999	9	Sunny
2	1/5/2017	28	-7777	Snow
3	1/6/2017	-9999	7	NaN
4	1/7/2017	32 #	-7777	Rain
5	1/8/2017	-9999	-7777	Sunny
6	1/9/2017	-9999	-7777	NaN
7	1/10/2017	34FA	8yyy	Cloudy
8	1/11/2017	40	12	Sunny

Example 2 - Regex

- dfW['temperature'].replace('[^0-9-]','',inplace=True,regex=True)
- **output**



	day	temperature	windspeed	event
0	1/1/2017	32	6us	Rain
1	1/4/2017	-9999	9	Sunny
2	1/5/2017	28	-7777	Snow
3	1/6/2017	-9999	7	NaN
4	1/7/2017	32	-7777	Rain
5	1/8/2017	-9999	-7777	Sunny
6	1/9/2017	-9999	-7777	NaN
7	1/10/2017	34	8yyy	Cloudy
8	1/11/2017	40	12	Sunny

Example 3 - Regex

- pakistan_intellectual_capital
- Practice of structuring problem. See in Jupytor notebook or PDF