REGULAR EXPRESSIONS

The 14..

- \ Used to drop the special meaning of character following it
- [] Represent a character class
- ^ Matches the beginning
- $ Matches the end
- . Matches any character except newline
- ? Matches zero or one occurrence.
- | Means OR (Matches with any of the characters separated by it.
- * Any number of occurrences (including 0 occurrences)
- + One ore more occurrences
- {} Indicate number of occurrences of a preceding RE to match.
- () Enclose a group of REs

re.findall()

- NORMAL re.findall(“substring”,”string”)

- Return all non-overlapping matches of pattern in string, as a list of strings. The string is scanned left-to-right, and matches are returned in the order found
  - *import re*
  - *string = 'hi my main number is 9821 and other is 9892'*
  - *# A sample regular expression to find digits.*
  - *regex = '\d+'*
  - *match = re.findall(regex, string)*
  - *print(match)*

    re.findall("(\+\d{2})|(\d{5})",s)

re.finditer("is",data):

- Gives where all data found
- .span() method in loop
- Can return start to end

```
import re
# \d is equivalent to [0-9].
p = re.compile('\d')
print(p.findall(“india was free on 15 aug 1947 at mid night 12.00am"))
```

- [ ]
- Class match
- A-Z
- a-z
- 0-9

Of specific

[^ mean every thing apart from range]

Function compile()

- Regular expressions are compiled into pattern objects, which have methods for various operations such as searching for pattern matches or performing string substitutions.

```
import re

# compile() creates regular expression character class [a-e] which is equivalent to [abcde].

# class [abcde] will match with string with 'a', 'b', 'c', 'd', 'e'.
p = re.compile('[a-e]')

# findall() searches for the Regular Expression and return a list upon finding
print(p.findall("this is tony stark the iron man"))
Also can call .sub(“new string”,”Main String”)
```

Use of \

- Metacharacter blackslash ‘\’ has a very important role as it signals various sequences. If the blackslash is to be used without its special meaning as metacharacter, use’\\’
- \d Matches any decimal digit, this is equivalen to the set class [0-9].
  - *\d{specific digit}*
  - *\d{start-end} range of numbers*
  - *\d{min,max}*
- \D Matches any non-digit character.
- \s Matches any whitespace character.
- \S Matches any non-whitespace character
- \w Matches any alphanumeric character, this is equivalent to the class
- [a-zA-Z0-9_].
- \W Matches any non-alphanumeric character.

split()

- Split string by the occurrences of a character or a pattern, upon finding that pattern, the remaining characters from the string are returned as part of the resulting list.
- Syntax :
  - *re.split(pattern, string, maxsplit=0, flags=0)*

  - *from re import split*
  - *# '\W+' denotes Non-Alphanumeric Characters or group of characters*
  - *# Upon finding ',' or whitespace ' ', the split(), splits the string from that point*
  - *print(split('\W+', 'this are my final words'))*

sub()

- The ‘sub’ in the function stands for SubString, a certain regular expression pattern is searched in the given string(3rd parameter), and upon finding the substring pattern is replaced by repl(2nd parameter), count checks and maintains the number of times this occurs.
- Syntax:
  - *re.sub(pattern, repl, string, count=0, flags=0)*

```
# Regular Expression pattern 'ub' matches the string at "Subject" and "Uber".
# As the CASE has been ignored, using Flag, 'uber' should match twice with the string
# Upon matching, 'uber' is replaced by 'OLA'
print(re.sub('uber', 'OLA' , 'Subject has Uber booked already', flags = re.IGNORECASE))
```

escape()

- Return string with all non-alphanumerics backslashed, this is useful if you want to match an arbitrary literal string that may have regular expression metacharacters in it.

re.search() :

- This method either returns None (if the pattern doesn’t match), or a re.MatchObject that contains information about the matching part of the string. This method stops after the first match, so this is best suited for testing a regular expression more than extracting data.