Regular Expression:

Regular expressions are used to perform search-related tasks in Python. In this tutorial, our primary focus should be on understanding because we are going to cover a concept that has a wide range of uses. To work with regular expressions, we have to import a built-in module in Python called **'re'.**

⇨ import re

The module defines several functions and constants to work with RegEx. The "re" module is composed of five functions known as:

- **findall**: It finds all searches for matches and prints resultant in the form of a list.
- **search**: It works the same as a findall, but the resultant is a matched object if any is found.
- **split**: The split function splits the string from every matched into two new strings.
- **sub**: The sub-function works exactly like a replace function in notepad or MS Word. It replaces the original word with a word of our choice.
- **finditer**: The finditer yields an iterator as a resultant with all the objects that match the one we sent it) finditer supports more attributes than any other function defined above. It also provides more details related to the matched object. So, most of the examples we are going to see next will contain a finditer function in them.

So, you must be wondering that all of the searchings can easily be done using a simple loop with some conditions so, what is the purpose of the "re" module. Well "re" module is used for complex searching, using Metacharacters and special sequences.
Metacharacters have special meaning in Python, and they are used with "re" modules to search for keywords and objects more technically and efficiently. We will see the working of a few Meta Characters in this tutorial so you can get an idea. I will provide you with a list of these characters and their working at the end of this tutorial.

Use of "^":-

We use the" ^" symbol to check whether the string is starting from the keyword we wrote after ^ or not. For example, if a string starts from CodeWithHarry and we are searching the keyword using ^CodeWithHarry with finditer, it will return us whether

our string is starting from the searched keyword or not. The same is the case for $ sign. It will check whether our string is ending with the specific keyword or not.

Use of "|":-

We can also use a unique character "|" to use more than one condition, so if we use it for the above case, then it will check whether the string starts or ends with CodeWithHarry.Now we will move on to special sequences. We will see a few special sequences in this tutorial, and you can have a look at the list of these sequences at the end of the tutorial description for further practice.

- **\A:** the resultant is a match if the input characters are at the beginning of the string
- **\b** the resultant is a match whether the input characters are at the beginning or the end of a word
- **\d** the resultant is a match if the string contains any digits
- **\s** the resultant is a match if the string contains a white space character

There are many metacharacters supported by the re module. Some characters with their working are the following:

- '**.**': Matches any single character except newline
- '**$**': Anchors a match at the end of a string
- '*****': Matches zero or more repetitions
- '**+**':Matches one or more repetitions
- '**{}**': Matches an explicitly specified number of repetitions
- '**[]**': Specifies a character class

To explore more about the re module, check the https://docs.python.org/3/library/re.html python documentation.

Re.txt file as described in the video!

Meta Characters
[] A set of characters
\ Signals a special sequence (can also be used to escape special characters)
. Any character (except newline character)
^ Starts with
$ Ends with
* Zero or more occurrences
+ One or more occurrences

{} Exactly the specified number of occurrences
| Either or
() Capture and group
Special Sequences
\A Returns a match if the specified characters are at the beginning of the string
\b Returns a match where the specified characters are at the beginning or at the end of a word 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

\d Returns a match where the string contains digits (numbers from 0-9)
\D Returns a match where the string DOES NOT contain digits
\s Returns a match where the string contains a white space character
\S Returns a match where the string DOES NOT contain a white space character
\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 Returns a match where the string DOES NOT contain any word characters
\Z Returns a match if the specified characters are at the end of the string

```
import re

mystr = '''Tata Limited

Dr. David Landsman, executive director

18, Grosvenor Place

London SW1X 7HSc

Phone: +44 (20) 7235 8281

Fax: +44 (20) 7235 8727

Email: tata@tata.co.uk

Website: www.europe.tata.com

Directions: View map
```

Tata Sons, North America

1700 North Moore St, Suite 1520

Arlington, VA 22209-1911

USA

Phone: +1 (703) 243 9787

Fax: +1 (703) 243 9791

66-66

455-4545

Email: northamerica@tata.com

Website: www.northamerica.tata.com

Directions: View map fass

harry bhai lekin

bahut hi badia aadmi haiaiinaiiiiiiiiiiii'''

```
# findall, search, split, sub, finditer
# patt = re.compile(r'fass')
# patt = re.compile(r'.adm')
# patt = re.compile(r'^Tata')
# patt = re.compile(r'iin$')
# patt = re.compile(r'ai{2}')
# patt = re.compile(r'(ai){1}')
# patt = re.compile(r'ai{1}|Fax')
```

```
# Special Sequences
# patt = re.compile(r'Fax\b')
# patt = re.compile(r'27\b')
patt = re.compile(r'\d{5}-\d{4}')
# Task
# Given a string with a lot of indian phone numbers starting from +91

matches = patt.finditer(mystr)
for match in matches:
    print(match)


"""


"""
```