😊 Files in Python

1. Python uses file objects to interact with external files on your computer. These file objects can be any file you have on your computer, whether it be an audio file, a text file, emails, Excel documents, etc. Note: You will probably need to install certain libraries or modules to interact with those various file types, but they are easily available.
2. Python has a built-in open function that allows us to open and play with basic file types. First, we will need a file, though.
3. Below, you can see that in the first cell, we created a file using the **%%writefile test.txt** command, and then we wrote some text in the file as well, but you can only do this in the Jupyter Notebook; you cannot perform this activity in any other editor.
4. Now, if you are using any other editor other than Jupyter notebook, then you can create a simple file, write some text, and save it with the rest of the .py files.

```
[1]: %%writefile test.txt
     Hello, this is a quick test file.

     Overwriting test.txt
```

5. First, let's open the test.txt file, which is in the same directory as this notebook. We'll deal with files that are in the same directory as the notebook or .py script you're using for the time being.
6. Below you can see that we ran the command to open our file, but accidentally we wrote the wrong name, and instantly we got an error saying the file was not found.
7. Also, you can use the **pwd** command to know your present working directory in your Jupyter Notebook.

```
[7]: myfile = open('hello.txt')

---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[7], line 1
----> 1 myfile = open('hello.txt')

File ~\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:286, in _modified_open(file, *args, **kwargs)
    279 if file in {0, 1, 2}:
    280     raise ValueError(
    281         f"IPython won't let you open fd={file} by default "
    282         "as it is likely to crash IPython. If you know what you are doing, "
    283         "you can use builtins' open."
    284     )
--> 286 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'hello.txt'
```

```
[5]: pwd

[5]: 'C:\\Users\\PULKIT'
```

8. Now we tried to open the correct file, and here you can see that we get no error. Then we tried to read the contents of our file, and when we tried to read the contents again, we could see that we got an empty string, and why is that? Because you can imagine the reading "cursor" is at the end of the file after having read it. So, there is nothing left

to read. We can reset the "cursor" like this: **my_file.seek0()** command it will bring the cursor to zero and then we can read our file again as you can see below.

```
[2]: # Open the text.txt we made earlier
     my_file = open('test.txt')

[3]: # We can now read the file
     my_file.read()

[3]: 'Hello, this is a quick test file.'

[4]: # But what happens if we try to read it again?
     my_file.read()

[4]: ''
```

This happens because you can imagine the reading "cursor" is at the end of the file after having read it. So there is nothing left to read. We can reset the "cursor" like this:

```
[5]: # Seek to the start of file (index 0)
     my_file.seek(0)

[5]: 0

[6]: # Now read again
     my_file.read()

[6]: 'Hello, this is a quick test file.'
```

9. You can read a file line by line using the readlines method. Using this method, you can get the contents line by line. If you have multiple lines in the files.

```
[7]: # Readlines returns a list of the lines in the file
     my_file.seek(0)
     my_file.readlines()

[7]: ['Hello, this is a quick test file.']
```

When you have finished using a file, it is always good practice to close it.

```
[8]: my_file.close()
```

10. By default, the **open()** function will only allow us to read the file. We need to pass the argument **'w'** to write over the file.
11. Opening a file with 'w' or 'w+' truncates the original, meaning that anything that was in the original file **is deleted**!

```
[9]: # Add a second argument to the function, 'w' which stands for write.
     # Passing 'w+' lets us read and write to the file

     my_file = open('test.txt','w+')
```

12. The **write()** method writes the string **"This is a new line"** to the file. This string has **18 characters**, including spaces.
13. The **write()** method returns the number of characters it successfully writes, which in this case is 18. That's why you see 18 as the output.
14. The **seek(0)** moves the file pointer back to the beginning of the file.

15. The **read()** method reads the entire file content, returning '**This is a new line**', which matches what was written.

```
[10]: # Write to the file
      my_file.write('This is a new line')

[10]: 18

[11]: # Read the file
      my_file.seek(0)
      my_file.read()

[11]: 'This is a new line'

[12]: my_file.close()  # always do this when you're done with a file
```

16. Passing the argument **'a'** opens the file and puts the pointer at the end, so anything written is appended. Like **'w+', 'a+'** lets us read and write to a file. If the file does not exist, one will be created.

```
[13]: my_file = open('test.txt','a+')
      my_file.write('\nThis is text being appended to test.txt')
      my_file.write('\nAnd another line here.')

[13]: 23

[14]: my_file.seek(0)
      print(my_file.read())

      This is a new line
      This is text being appended to test.txt
      And another line here.

[15]: my_file.close()
```

17. The code below demonstrates how to iterate through a file line by line in Python using a for loop.
18. First, we overwrote a file, then using the for loop, we opened the file. Here, the loop variable is arbitrarily named **asdf**, but the behaviour remains the same.

[9]:
```
%%writefile test.txt
First Line
Second Line
```

```
Overwriting test.txt
```

[11]:
```
for line in open('test.txt'):
    print(line)
```

```
First Line

Second Line
```

[13]:
```
# Pertaining to the first point above
for asdf in open('test.txt'):
    print(asdf)
```

```
First Line

Second Line
```