Files

Python uses file objects to interact with external files on computer. These file objects can be any sort of file you have on your computer for e.g. text files, emails, excel external docs.

Note: You need to install various libraries & modules to deal with certain file types

IPYTHON writing a file

Specific to Jupyter notebook

In: %%writefile test.txt
~~o/p: Hello~~
In: Hello this is a quick test file

o/p: Overwriting test.txt

PYTHON Opening a file

In: myfile = open('whoops.txt')

o/p: FileNotFoundError

To check your notebook location use pwd

O/p: 'C:\\Users\\ . ____ - - - . . . ____

Alternatively, to grab files from any location on your PC, just pass in the entire file path.

For windows you need to use double \ so python doesn't treat second \ as a ~~ex~~ escape character.

Windows:

myfile=open("C:\\Users\\ - - - - - - - - - -)

MacOS & Linux

myfile=open("/Users/yourname - - -)

Opening a file

In: my_file= open('test.txt')

Reading the file

In: my_file.read()

O/p: 'Hello, this is a quick test file.'

What happens when we try to read it again

In: my_file.read()

o/p: ''

⇒ This ~~hap~~ happens because you can imagine the reading 'cursor' is at the end of the file after reading it. So there is nothing ~~left~~ left for read

// Resetting the cursor at desired position:- Seek(index)

In: my_file.seek(0)
o/p: 0

// Now read again

In: my_file.read()

o/p: 'Hello, this is a quick test file.'

Readlines() - ~~method~~ returns a list of lines in file

In: my_file.seek(0)
my_file.readlines()

o/p: ['Hello, this is a quick test file.']

// When you have finished reading a file, it's always a good practice to close it.

In: my_file.close()

Reading, writing, Appending modes

→ 'r' → read only

→ 'w' → write only (will overwrite files or create new)

→ a → append only (will add on to files)

→ 'r+' → reading & writing

→ 'w+' → writing & reading (overwrites existing files or creates new one)

Writing to a file

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

In:
WRITE TO A FILE

In: my_file.write('This is a new line')

O/P: 18

In: my_file.seek(0)
my_file.read()

O/P: 'This is a new line'

In: my_file.close()

Appending to a file

→ 'a' → opens file & puts pointer at end

→ 'a+' → Let's us read & write to a file, If file does not exist one would be created

In: my_file = open('test.txt','a+')
~~my~~
my_file.write('\n This is text being appended to test.txt')

my_file.write('\n And another line here')

O/P: 28

In: my_file.seek(0)
print(my_file.read())

o/p: This is a new line
This is text being appended to test.txt
And another line here.

In: my_file.close()

Appending with %%writefile

In: %%writefile -a test.txt

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

o/p: Appending to test.txt

Note: Add a blank space if you want the 1st line to begin on its own line, as Jupyter won't recognize escape sequences such as \n

Iterating thro. a file

In:
```
%%writefile test.txt
First line
Second line
```

o/p:
```
Overwriting test.txt
```

- - - - - - - - - - - - - - -

In:
```
for line in open('test.txt'):
        print(line)
```

o/p:
```
First line
Second line
```

- - - - - - - - - - - - - - -

In the above code, we said that for every line in this text file, go ahead & print that line.

Note: →) We could have called the 'line' object anything

→) By not calling .read() on the file, the whole text file wasn't stored in memory

→) Notice the indent on second line for print, it's very imp. in python