13. File IO

October 20, 2022

1 Introduction

- Files are named locations on disk to store related information. They are used to permanently store data in a non-volatile memory (e.g. hard disk).
- Since Random Access Memory (RAM) is volatile (which loses its data when the computer is turned off), we use files for future use of the data by permanently storing them.
- When we want to read from or write to a file, we need to open it first. When we are done, it needs to be closed so that the resources that are tied with the file are freed.
- Hence, in Python, a file operation takes place in the following order:
  1. Open a file
  2. Read or write (perform operation)
  3. Close the file

2 Opening a File

- Python has a built-in `open()` function to open a file.
- This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

[1]:
```
file = open("quotes.txt")
```

2.1 File opening Modes

- We can specify the mode while opening a file. In mode, we specify whether we want to read, write, append to the file, etc.

Mode	Description
r	Opens a file for reading. (default)
w	Opens a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
x	Opens a file for exclusive creation. If the file already exists, the operation fails.
a	Opens a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.

Mode	Description
t	Opens in text mode. (default)
b	Opens in binary mode.
+	Opens a file for updating (reading and writing)

- The default is reading in text mode. In this mode, we get strings when reading from the file.
- Binary mode returns bytes and this is the mode to be used when dealing with non-text files like images or executable files.

3 Closing a File

- When we are done with performing operations on the file, we need to properly close the file.
- Closing a file will free up the resources that were tied with the file.
- It is done using the `close()` method.

[2]:
```
file.close()
```

4 Reading Files

- To read a file in Python, it must br opened in reading `r` mode.

[3]:
```
file = open("quotes.txt","r")
```

4.1 `read(size)`

- There are various methods available for this purpose. We can use the `read(size)` method to read in the size number of data. If the size parameter is not specified, it reads and returns up to the end of the file.
- `read()` method returns a newline as `'\n'`.
- Once the end of the file is reached, we get an empty string on further reading.

[ ]:
```
file = open("quotes.txt", "r")
```

[4]:
```
file.read(5)  # read the first 5 data
```

[4]:
```
'The b'
```

[5]:
```
file.read(10) # read the next 10 data
```

[5]:
```
'est way to'
```

[6]:
```
file.read()
```

[6]:
```
" get started is to quit talking and begin doing.\nDon't let yesterday take up
too much of today.\nYou learn more from failure than from success. Don't let it
```

```
stop you.\nIt's not whether you get knocked down, it's whether you get up.\nWe
generate fears while we sit. We overcome them by action."
```

[7]:
```
file.read()
```

[7]:
```
''
```

4.2 `seek()` and `tell()`

- We can change our current file cursor (position) using the `seek()` method.
- Similarly, the `tell()` method returns our current position (in number of bytes).

[8]:
```
file.tell()  # get the current file position
```

[8]:
```
304
```

[9]:
```
file.seek(0)   # bring file cursor to initial position
```

[9]:
```
0
```

[10]:
```
file.tell()
```

[10]:
```
0
```

[11]:
```
file.read()
```

[11]:
```
"The best way to get started is to quit talking and begin doing.\nDon't let
yesterday take up too much of today.\nYou learn more from failure than from
success. Don't let it stop you.\nIt's not whether you get knocked down, it's
whether you get up.\nWe generate fears while we sit. We overcome them by
action."
```

4.3 `readline()`

- `readline()` method is used to read individual lines of a file.
- This method reads a file till the newline, including the newline character.

[12]:
```
file.tell()
```

[12]:
```
304
```

[13]:
```
file.seek(100)
```

[13]:
```
100
```

[14]:
```
file.readline()
```

[14]:
```
' of today.\n'
```

[15]:
```
file.readline()
```

[15]:
```
"You learn more from failure than from success. Don't let it stop you.\n"
```

[16]:
```
file.readline()
```

[16]:
```
"It's not whether you get knocked down, it's whether you get up.\n"
```

[17]:
```
file.readline()
```

[17]:
```
'We generate fears while we sit. We overcome them by action.'
```

[18]:
```
file.readline()
```

[18]:
```
''
```

[19]:
```
file.readline()
```

[19]:
```
''
```

4.4 `readlines()`

- `readlines()` method returns a list of remaining lines of the entire file.

[20]:
```
file.seek(0)
```

[20]:
```
0
```

[21]:
```
file.readlines()
```

[21]:
```
['The best way to get started is to quit talking and begin doing.\n',
 "Don't let yesterday take up too much of today.\n",
 "You learn more from failure than from success. Don't let it stop you.\n",
 "It's not whether you get knocked down, it's whether you get up.\n",
 'We generate fears while we sit. We overcome them by action.']
```

[23]:
```
file.seek(0)
file.readlines()
```

4.5 Using `for` loop

- We can read a file line-by-line using a `for` loop.

[27]:
```
file.seek(0)
```

[27]:
```
0
```

[28]:
```
for line in file:
    print(line)
```

```
The best way to get started is to quit talking and begin doing.

Don't let yesterday take up too much of today.

You learn more from failure than from success. Don't let it stop you.

It's not whether you get knocked down, it's whether you get up.

We generate fears while we sit. We overcome them by action.
```

[29]:
```
file.close()
```

5 Writing to Files

5.1 `write()`

- In order to write into a file in Python, we need to open it in write `w`, append `a` or exclusive creation `x` mode.
- Be careful with the `w` mode, as it will overwrite into the file if it already exists. Due to this, all the previous data are erased.
- Only string datatype can be written to a file.
- Writing a string to a file is done using the `write()` method. This method returns the number of characters written to the file.

[30]:
```
# This line will create a new file named test.txt in the current directory
# if it does not exist. If it does exist, it is overwritten.
file_w = open("test.txt", "w")
```

[31]:
```
file_w.write("My first file\n")
```

[31]: 14

[32]:
```
file_w.write("My second line in first file\n")
```

[32]: 29

[33]:
```
file_w.write("contains three lines\n")
```

[33]: 21

[34]:
```
file_w.close()
```

[37]:
```
file_w = open("test.txt", "w")
file_w.write("12  My first file\n")
```

```
file_w.write("13 My second line in first file\n")
file_w.write("15 contains three lines\n")
```

[37]: 24

5.2 flush()

- Flushes the write buffer of the file stream.

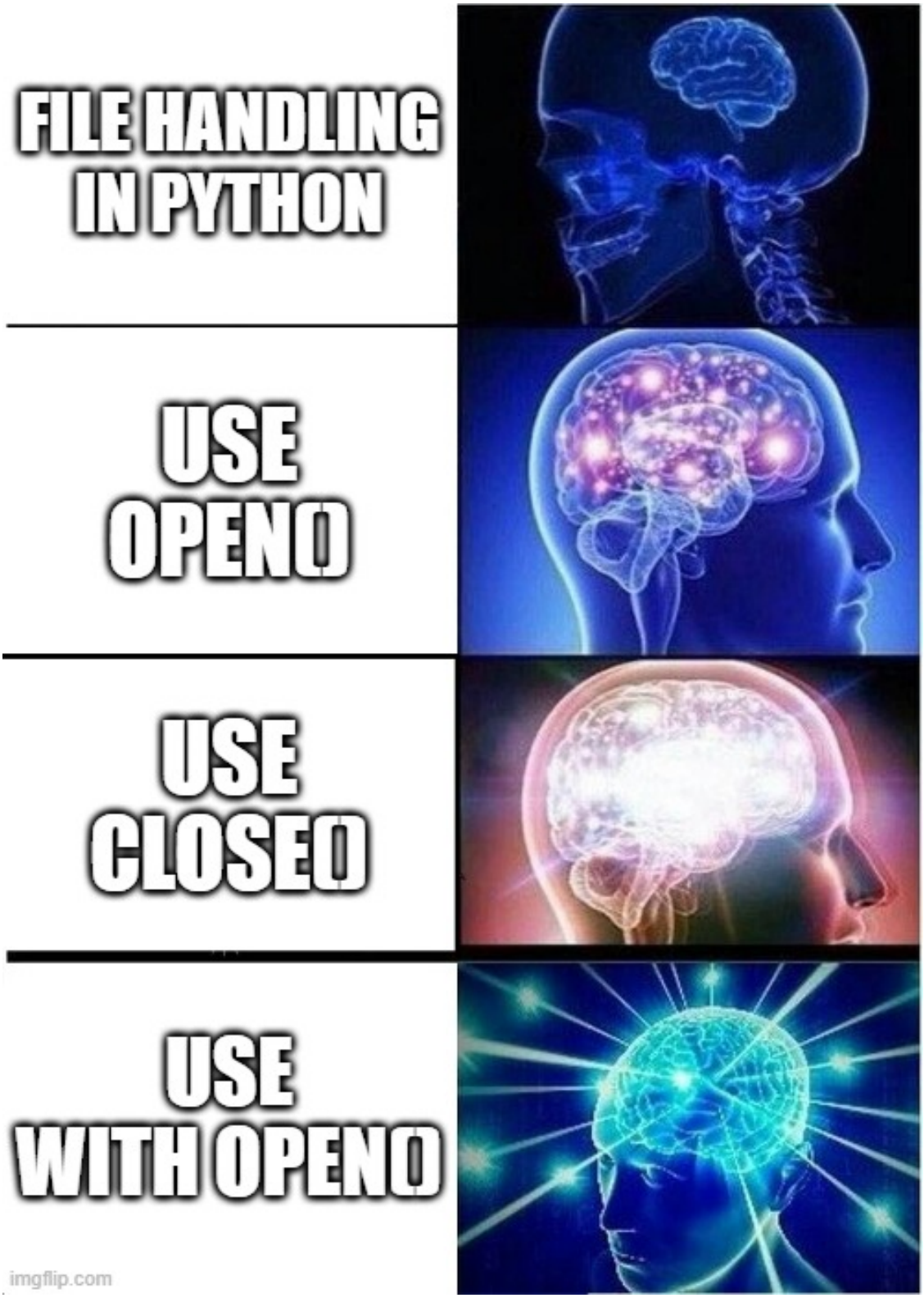
[38]:
```
file_w.flush()
```

[39]:
```
file_w.writable() # Returns True if the file stream can be written to.
```

[39]: True

[40]:
```
file_w.close()
```

5.3 `with` statement



- Another way to write a file is by using the `with` statement.

- This ensures that the file is closed when the block inside the with statement is exited.
- We don't need to explicitly call the `close()` method. It is done internally.

[41]:
```
with open("test.txt", "a") as f:
    f.write("My first file using with statement\n")
    f.write("My second line in first file\n")
    f.write("contains three lines")
```

5.4 `writelines(lines)`

- Writes a `list` of lines to the file.

[42]:
```
lines = ['The best way to get started is to quit talking and begin doing.\n',
 "Don't let yesterday take up too much of today.\n",
 "You learn more from failure than from success. Don't let it stop you.\n",
 "It's not whether you get knocked down, it's whether you get up.\n",
 'We generate fears while we sit. We overcome them by action.']
```

[43]:
```
quotes_file_2 = open("quotes2.txt","a")
quotes_file_2.writelines(lines)
quotes_file_2.close()
```

- Using `with` statement

[ ]:
```
with open("quotes2.txt","w") as f:
    f.writelines(lines)
```