Hyperiondev

Workshop - File IO

Lecture – Housekeeping

- ❑ The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- ❑ No question is daft or silly - **ask them!**
- ❑ There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- ❑ For all non-academic questions, please submit a query: www.hyperiondev.com/support
- ❑ Report a safeguarding incident: http://hyperiondev.com/safeguardreporting
- ❑ We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0q/

Objectives

1. File I/O:
   a. Opening files
   b. Access modes
   c. Reading files
   d. Writing files

File I/O

- ★ File I/O stands for file input/output
- ★ It is a process that reads data from an external file on the computer or outputs to another file.
- ★ Python has a built-in file type, which is the complex data type.
- ★ This means that Python can create variables of type “file”.

Opening a File

★ To read from a file, we must first open it.

★ To open a file, we use Python’s built-in open() function, which creates what is known as a file object.

★ To utilize the file object’s data, we store the file object in a variable.

★ Once we are done, we then close the file.

Opening Files

★ To use a file in our program, we store the file object in a variable as such :

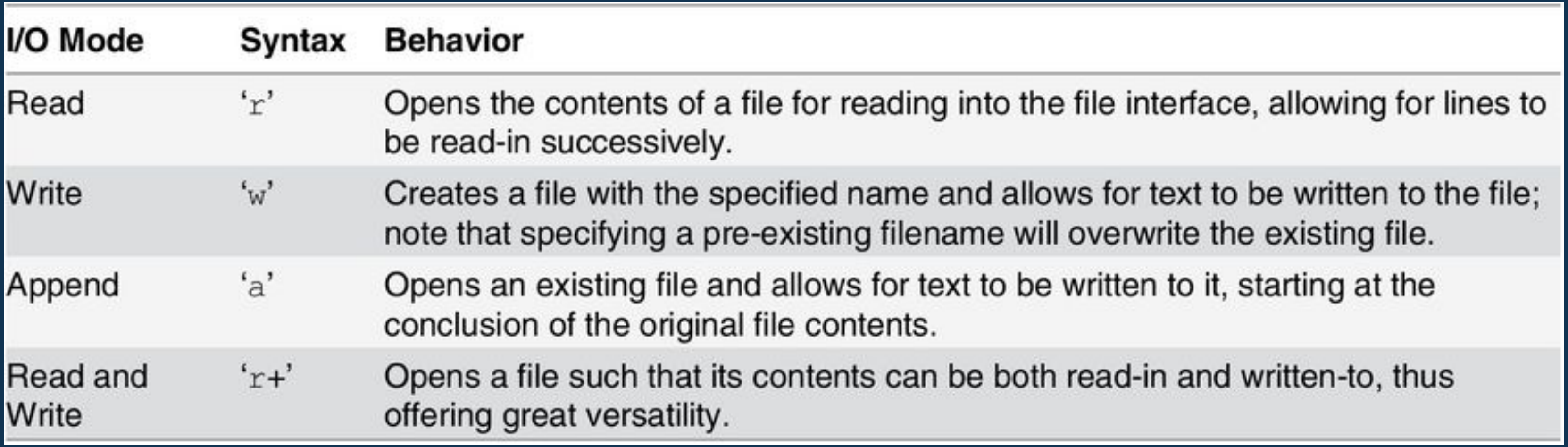
- **file = open(file_name , access_mode)**

OR

- **with open(file_name,access_mode) as file:**

★ **Access mode** : what the user can do when the file has been opened, such as reading (r), writing (w), appending(a) or reading and writing (r+).

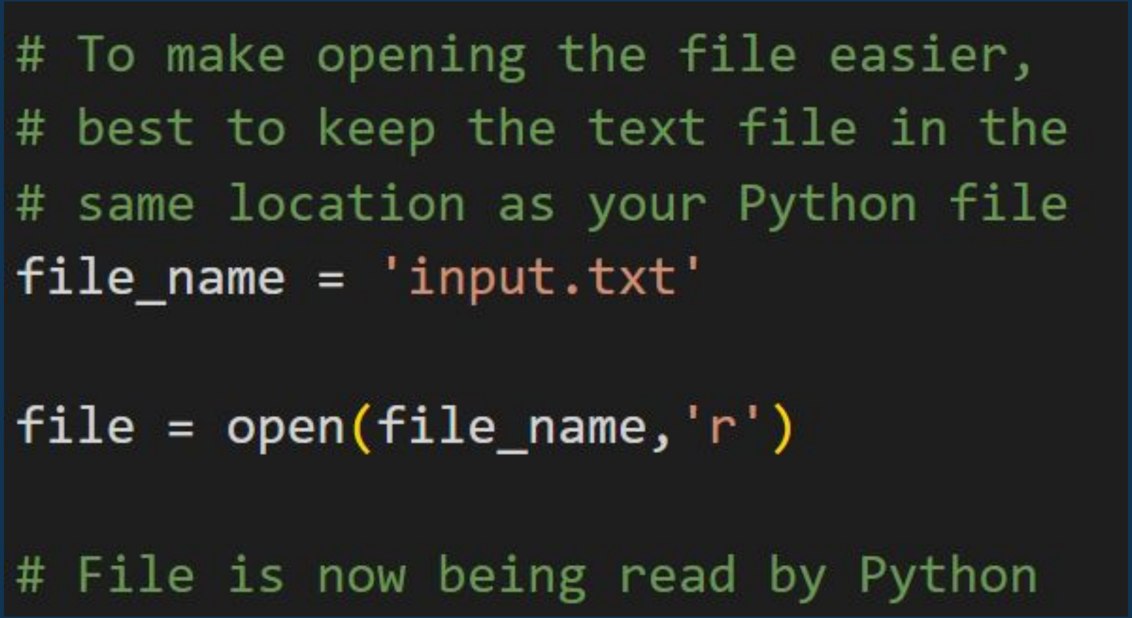
Access Modes



I/O Mode	Syntax	Behavior
Read	‘r’	Opens the contents of a file for reading into the file interface, allowing for lines to be read-in successively.
Write	‘w’	Creates a file with the specified name and allows for text to be written to the file; note that specifying a pre-existing filename will overwrite the existing file.
Append	‘a’	Opens an existing file and allows for text to be written to it, starting at the conclusion of the original file contents.
Read and Write	‘r+’	Opens a file such that its contents can be both read-in and written-to, thus offering great versatility.

Reference: https://www.researchgate.net/figure/Pythons-file-access-modes_fig6_303337342

Example 1: Opening



```
# To make opening the file easier,
# best to keep the text file in the
# same location as your Python file
file_name = 'input.txt'

file = open(file_name,'r')

# File is now being read by Python
```

Hyperiondev

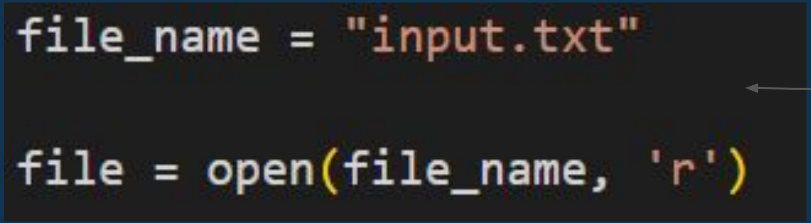
Reading Files

★ Files are opened in Python with the open() function. We know that open() will return a file object.

★ To then properly read the object, we will need to use the read method.

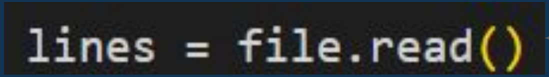
★ There are three methods :
   ○ .read()
   ○ .readline()
   ○ .readlines()

Read Example



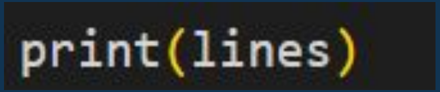
```python
file_name = "input.txt"

file = open(file_name, 'r')
```

Open file and allow its contents to be read by Python



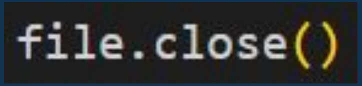
```python
lines = file.read()
```

.read() will simply read over all lines in our text file. The contents are saved in a variable called lines.



```python
print(lines)
```

To display the contents.



```python
file.close()
```

Remember to close the file.

Readline Example

```python
file_name = "input.txt"

file = open(file_name, 'r')
```

Open file and allow its contents to be read by Python

```python
lines = file.readline()
```

.readline() will simply read the first line in our text file. The contents are saved in a variable called lines.

```python
print(lines)
```

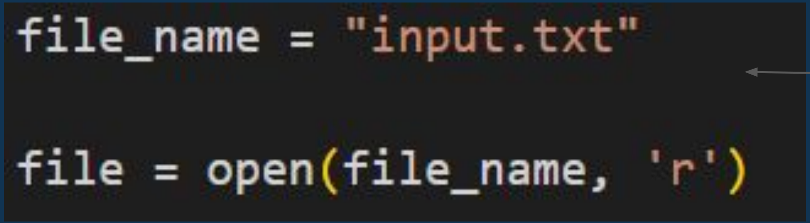
To display the contents.

```python
file.close()
```

Remember to close the file.

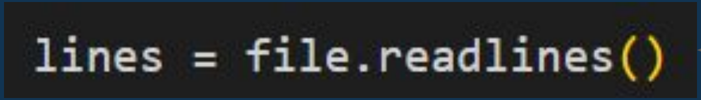
Hyperiondev

Readlines Example



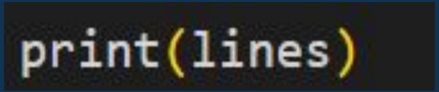
```
file_name = "input.txt"

file = open(file_name, 'r')
```

Open file and allow its contents to be read by Python



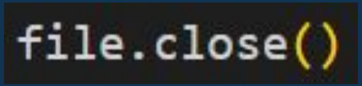
```
lines = file.readlines()
```

.readlines() will simply read over all each line individually within the text file. The contents are saved in a variable called lines.



```
print(lines)
```

To display the contents. Keep in mind that the output is actually a **list** (not a string)



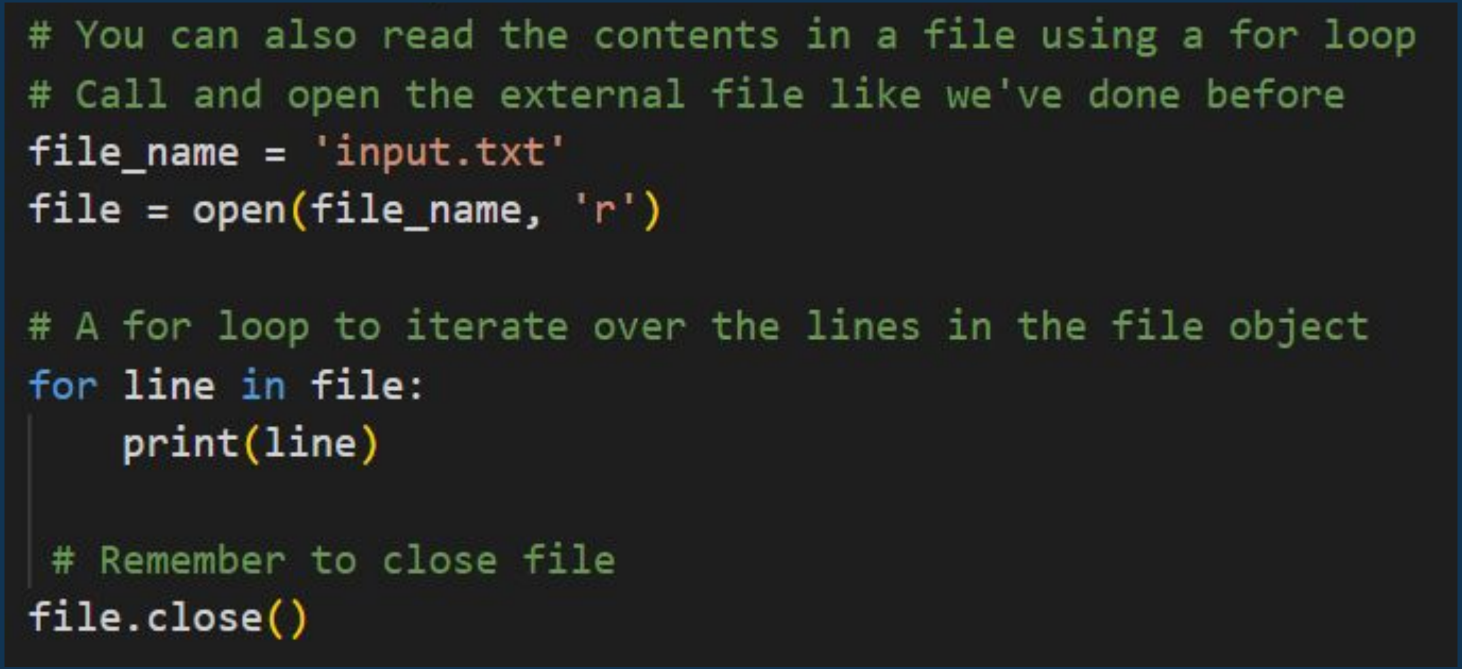
```
file.close()
```

Remember to close the file.

Closing a File

- ★ The close() method ensures system resources are not wasted in our programs.
- ★ It is always best practice to close files when you are finished working with them.
- ★ Remember that once a file is closed, it cannot be read again until is is re-opened.

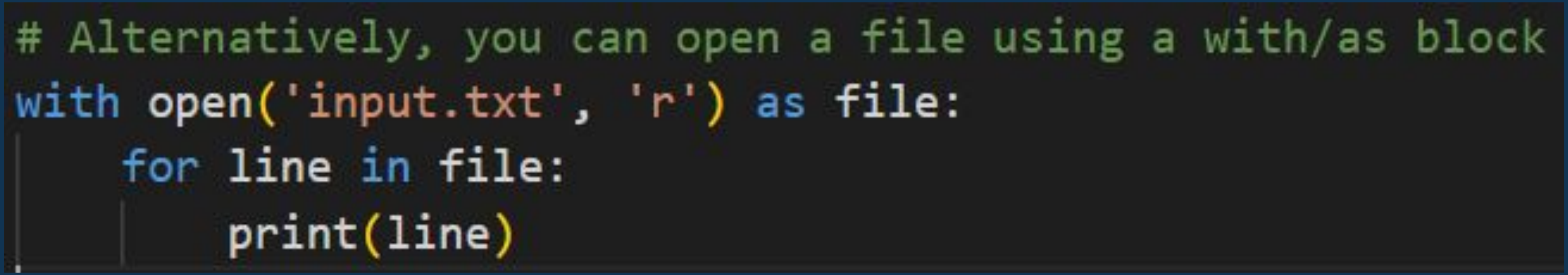
Example 2: Using a for loop to display contents



```python
# You can also read the contents in a file using a for loop
# Call and open the external file like we've done before
file_name = 'input.txt'
file = open(file_name, 'r')

# A for loop to iterate over the lines in the file object
for line in file:
    print(line)

 # Remember to close file
file.close()
```

Hyperiondev

With/as block to open files and display the contents



```
# Alternatively, you can open a file using a with/as block
with open('input.txt', 'r') as file:
    for line in file:
        print(line)
```

Hyperiondev

Writing to Files

- ★ Often, we will want to write data to a new file.
- ★ Usually after we have done a lot of computations or data processing and we would like to save the work and come back to it at another point.
- ★ Writing to a file has a simple multi-step process.

Prepping the file

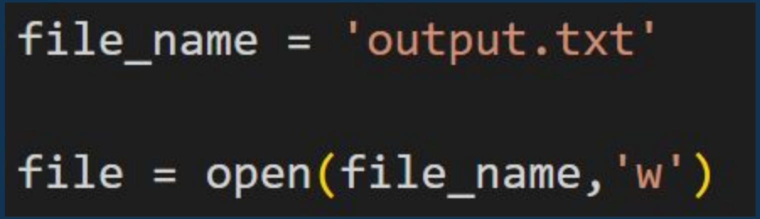
★ We already know how to open a file and store the file object in a variable.

★ Now the main difference between Input and Output is the access mode is different
  ○ Instead of reading from the file, we are now writing to the file using modes w or a

★ What comes next is then actually writing to the file, which we will take a look at now.

Hyperiondev

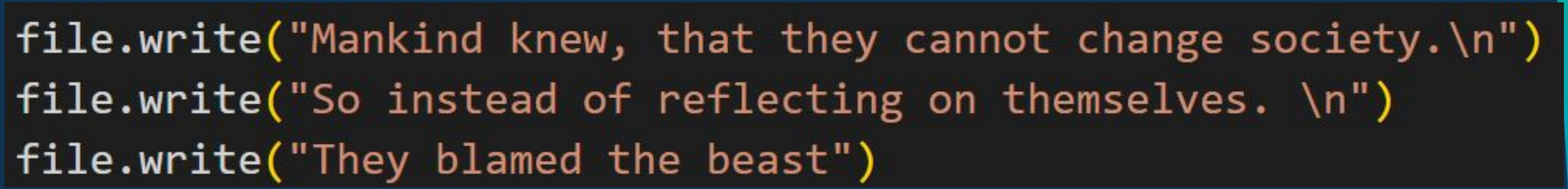
Example 3



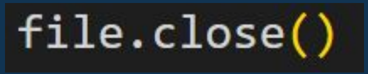
```python
file_name = 'output.txt'

file = open(file_name,'w')
```

Open file and allow contents to be written to it.

.write() will allow the sentences to be added to the text file.



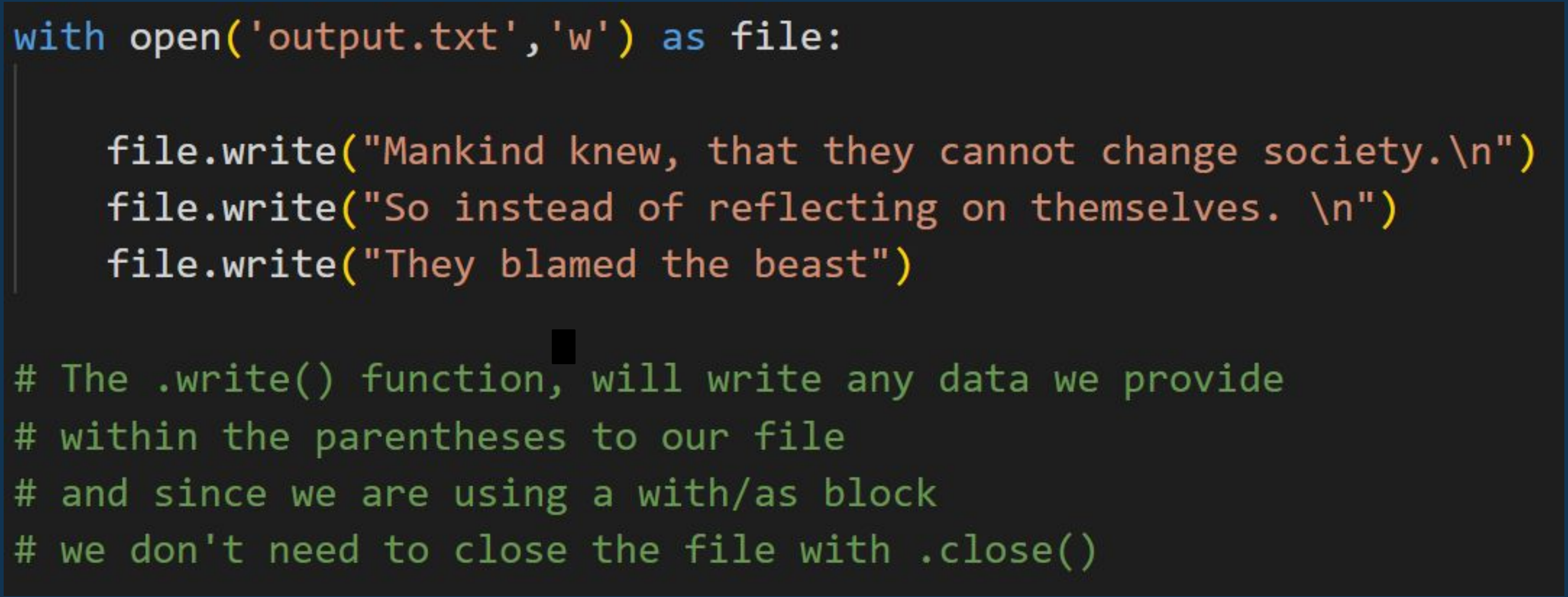
```python
file.write("Mankind knew, that they cannot change society.\n")
file.write("So instead of reflecting on themselves. \n")
file.write("They blamed the beast")
```



```python
file.close()
```

Remember to close the file.

Example 4: Using a with/as block



```python
with open('output.txt','w') as file:

    file.write("Mankind knew, that they cannot change society.\n")
    file.write("So instead of reflecting on themselves. \n")
    file.write("They blamed the beast")

# The .write() function, will write any data we provide
# within the parentheses to our file
# and since we are using a with/as block
# we don't need to close the file with .close()
```

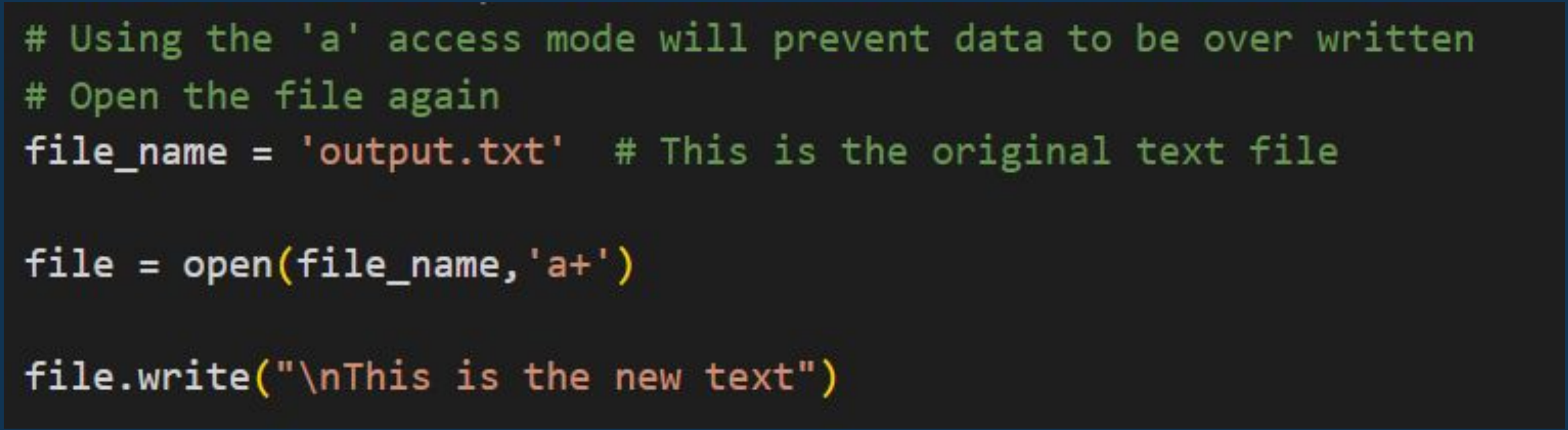
Things to Note

★ Remember that when the file is reopened and new data is written to the file, the previous data is then overwritten.

★ You can preserve the previous data by using the append (a) access mode. This will simply append the new data to the end of the file, instead of overwriting.

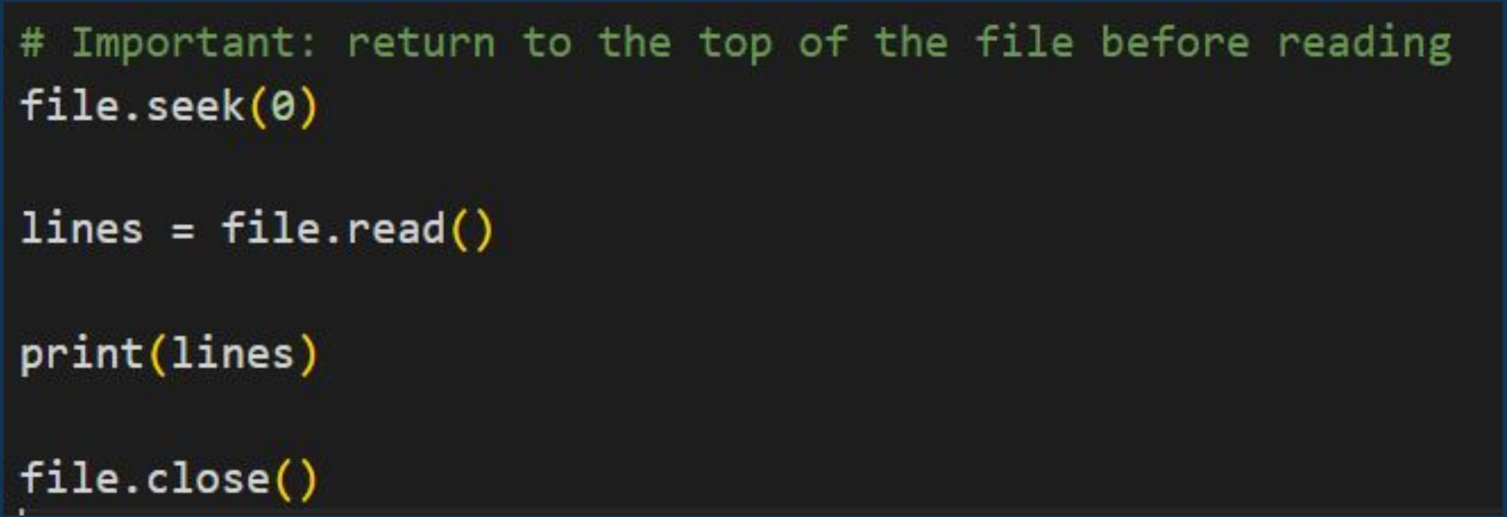
★ Always remember to close your file when you are done using it (unless you use a with/as block).

Example 5: Appending and reading



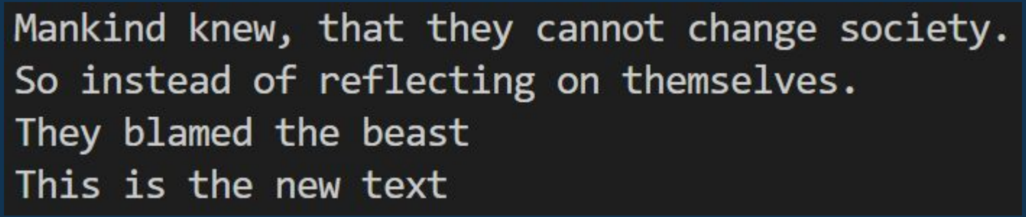
```
# Using the 'a' access mode will prevent data to be over written
# Open the file again
file_name = 'output.txt'  # This is the original text file

file = open(file_name,'a+')

file.write("\nThis is the new text")
```

Example 5 continued...



```
# Important: return to the top of the file before reading
file.seek(0)

lines = file.read()

print(lines)

file.close()
```

[output]



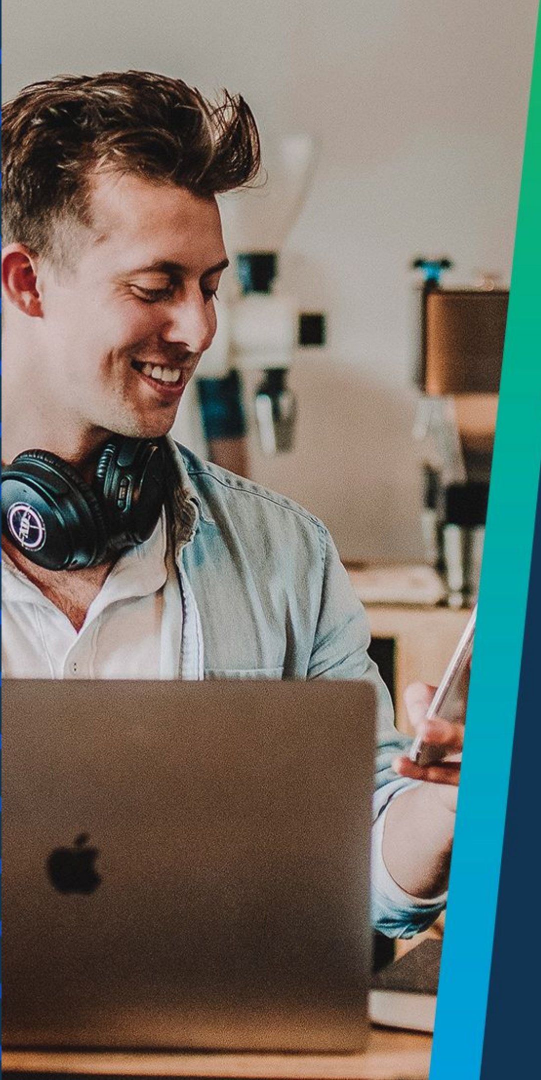
```
Mankind knew, that they cannot change society.
So instead of reflecting on themselves.
They blamed the beast
This is the new text
```

Hyperiondev

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any



Hyperiondev

Thank you for joining us

Stay hydrated
Avoid prolonged screen time
Take regular breaks
Have fun :)