- **Introduction File Handling**
- **File Operations**

File

File is the collection of data.

Helps us to store the data permanently. Which can be retrieved for future use.

Types of Files:

Text, CSV, Binary.

- 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).

File Operation in Python

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**

Text file

- Stores information in ASCII or unicode characters.
- Extension for text files is .txt
- Default mode of file

Open a file

1. Using open() function
2. Using with statement

Example

```
F = open(‘file.txt”, ‘r’)

F = open(“C:\\User\\file2.txt”, ‘r’)
```

Using with statement

Syntax:

```
With open(filename, fileMode) as FileObject:
            FileObject.write(“.........”)
```

File Access Modes

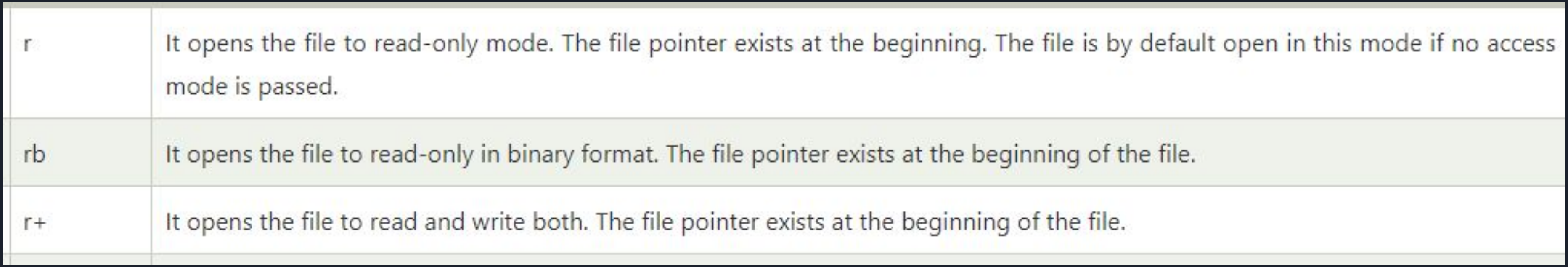
r $\rightarrow$ Opens file for read only . This is the Default Mode.

r+ $\rightarrow$ Reading and Write also

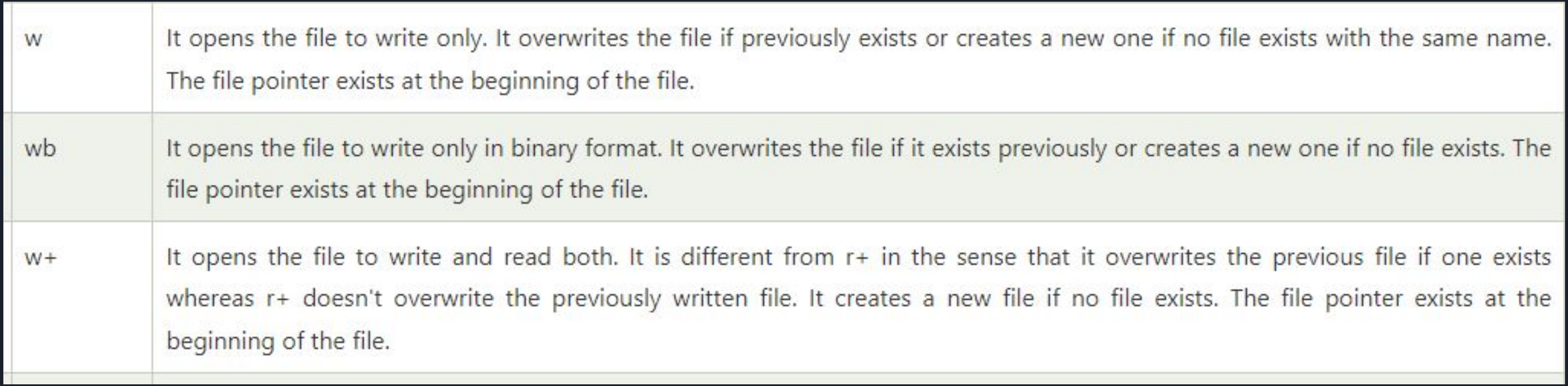
w $\rightarrow$ write to a file

w+ $\rightarrow$ Reading and Write also.

w+, a+ r+



r	It opens the file to read-only mode. The file pointer exists at the beginning. The file is by default open in this mode if no access mode is passed.
rb	It opens the file to read-only in binary format. The file pointer exists at the beginning of the file.
r+	It opens the file to read and write both. The file pointer exists at the beginning of the file.



w	It opens the file to write only. It overwrites the file if previously exists or creates a new one if no file exists with the same name. The file pointer exists at the beginning of the file.
wb	It opens the file to write only in binary format. It overwrites the file if it exists previously or creates a new one if no file exists. The file pointer exists at the beginning of the file.
w+	It opens the file to write and read both. It is different from r+ in the sense that it overwrites the previous file if one exists whereas r+ doesn't overwrite the previously written file. It creates a new file if no file exists. The file pointer exists at the beginning of the file.

Mode	Description
`r`	It opens an existing file to read-only mode. The file pointer exists at the beginning.
`rb`	It opens the file to read-only in binary format. The file pointer exists at the beginning.
`r+`	It opens the file to read and write both. The file pointer exists at the beginning.
`rb+`	It opens the file to read and write both in binary format. The file pointer exists at the beginning of the file.
`w`	It opens the file to write only. It overwrites the file if previously exists or creates a new one if no file exists with the same name.
`wb`	It opens the file to write only in binary format. It overwrites the file if it exists previously or creates a new one if no file exists.
`w+`	It opens the file to write and read data. It will override existing data.
`wb+`	It opens the file to write and read both in binary format
`a`	It opens the file in the append mode. It will not override existing data. It creates a new file if no file exists with the same name.
`ab`	It opens the file in the append mode in binary format.
`a+`	It opens a file to append and read both.
`ab+`	It opens a file to append and read both in binary format.

File access mode

Example

```
f = open("demofile.txt", "r")
print(f.read())
```

Read Only Parts of the File

By default the `read()` method returns the whole text, but you can also specify how many characters you want to return:

Example

Return the 5 first characters of the file:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

Read Lines

You can return one line by using the `readline()` method:

Example

Read one line of the file:

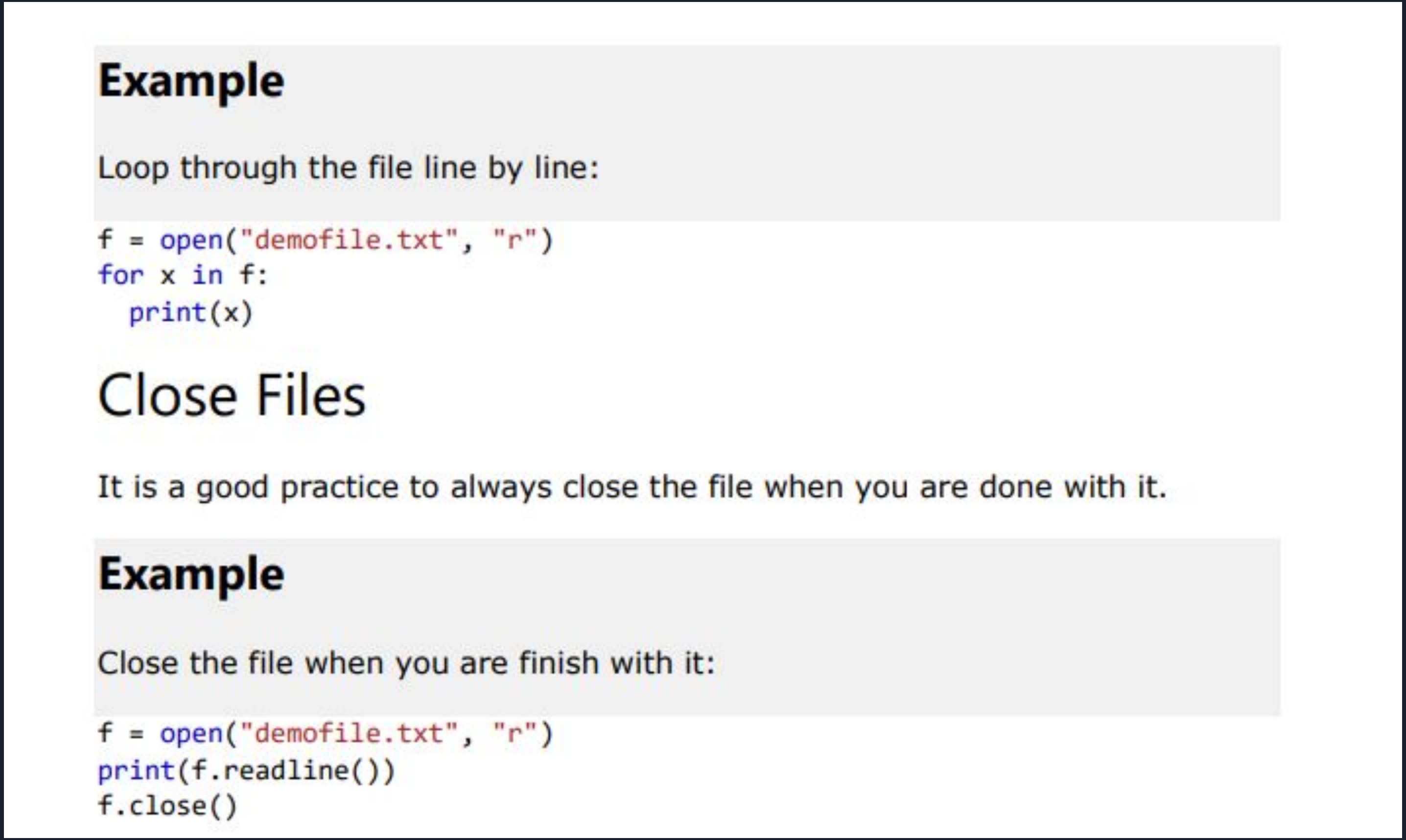
```
f = open("demofile.txt", "r")
print(f.readline())
```

By calling `readline()` two times, you can read the two first lines:

Example

Read two lines of the file:

```
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```



Example

Loop through the file line by line:

```
f = open("demofile.txt", "r")
for x in f:
  print(x)
```

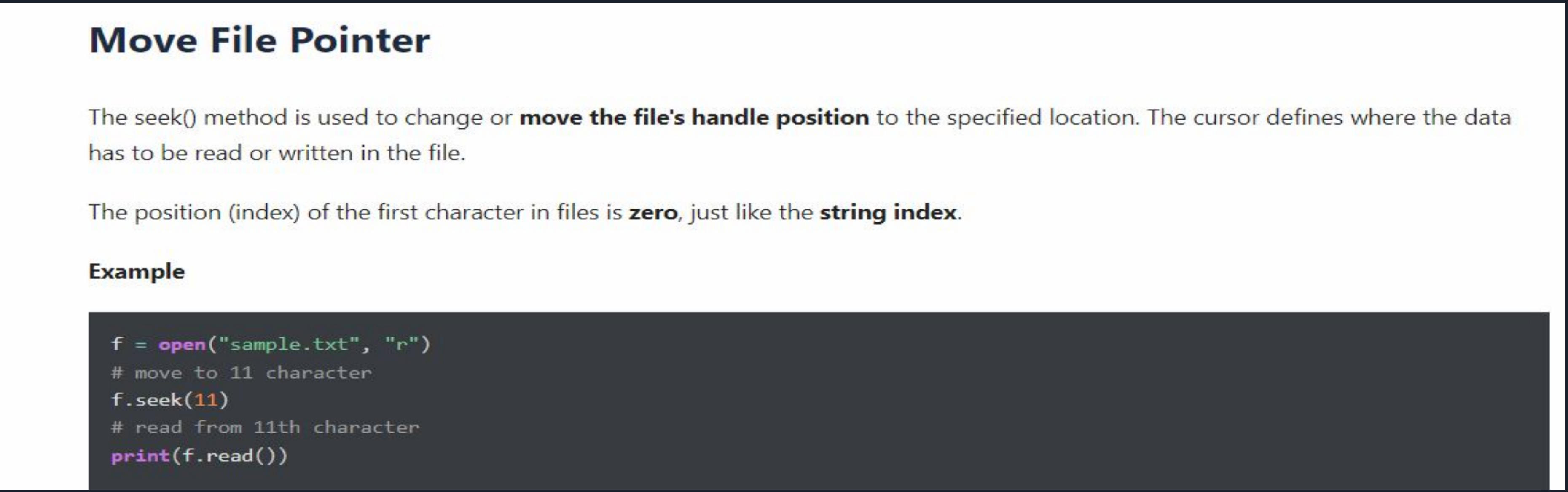
Close Files

It is a good practice to always close the file when you are done with it.

Example

Close the file when you are finish with it:

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```



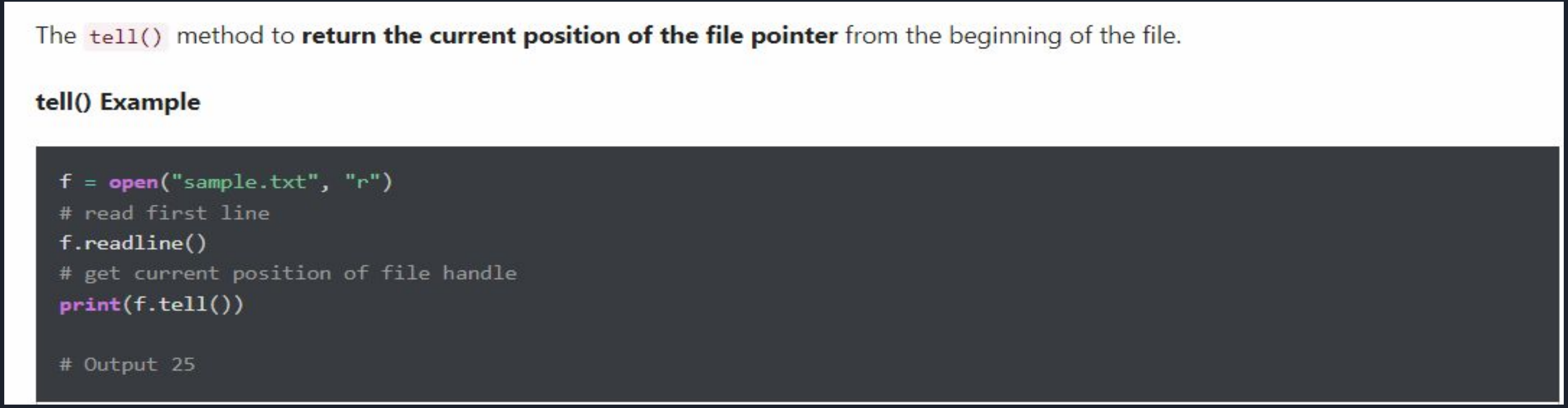
Move File Pointer

The seek() method is used to change or **move the file's handle position** to the specified location. The cursor defines where the data has to be read or written in the file.

The position (index) of the first character in files is **zero**, just like the **string index**.

Example

```
f = open("sample.txt", "r")
# move to 11 character
f.seek(11)
# read from 11th character
print(f.read())
```

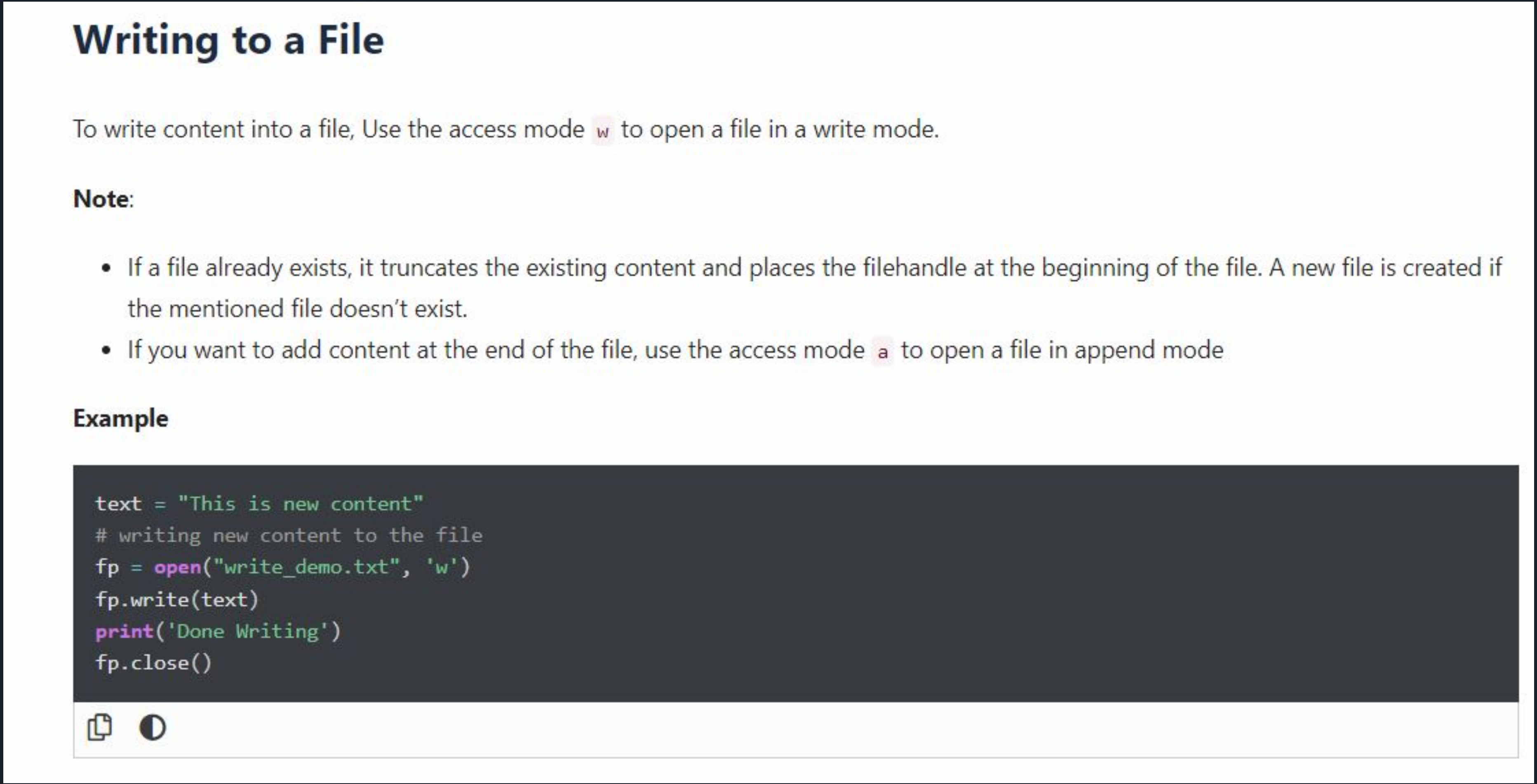


The `tell()` method to **return the current position of the file pointer** from the beginning of the file.

tell() Example

```
f = open("sample.txt", "r")
# read first line
f.readline()
# get current position of file handle
print(f.tell())

# Output 25
```



Writing to a File

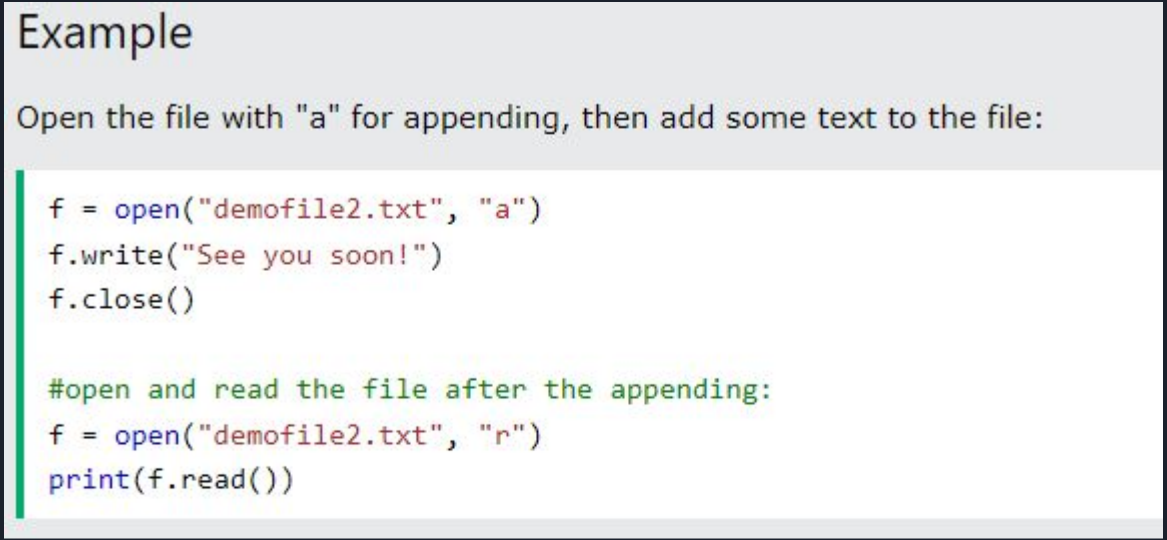
To write content into a file, Use the access mode `w` to open a file in a write mode.

Note:

- If a file already exists, it truncates the existing content and places the filehandle at the beginning of the file. A new file is created if the mentioned file doesn't exist.
- If you want to add content at the end of the file, use the access mode `a` to open a file in append mode

Example

```
text = "This is new content"
# writing new content to the file
fp = open("write_demo.txt", 'w')
fp.write(text)
print('Done Writing')
fp.close()
```

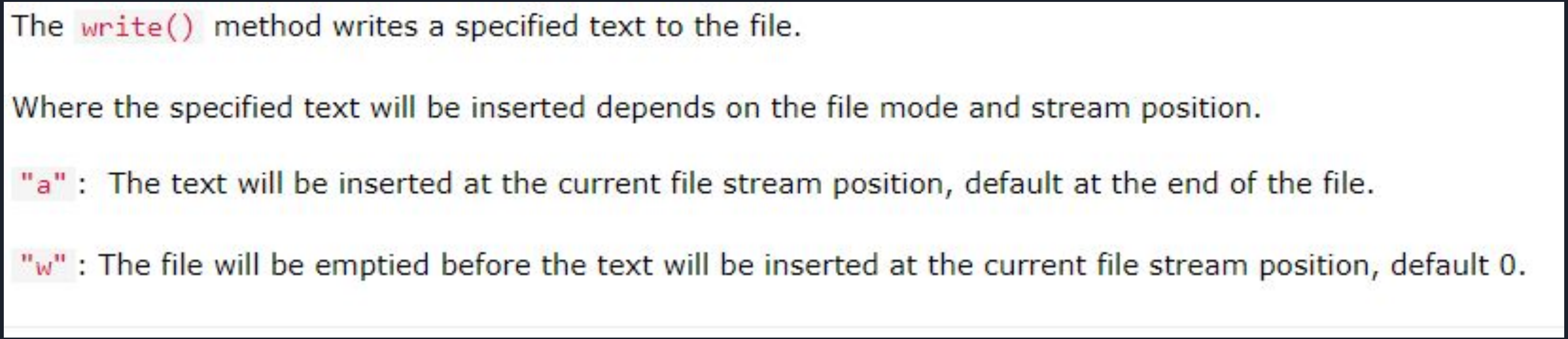


Example

Open the file with "a" for appending, then add some text to the file:

```python
f = open("demofile2.txt", "a")
f.write("See you soon!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

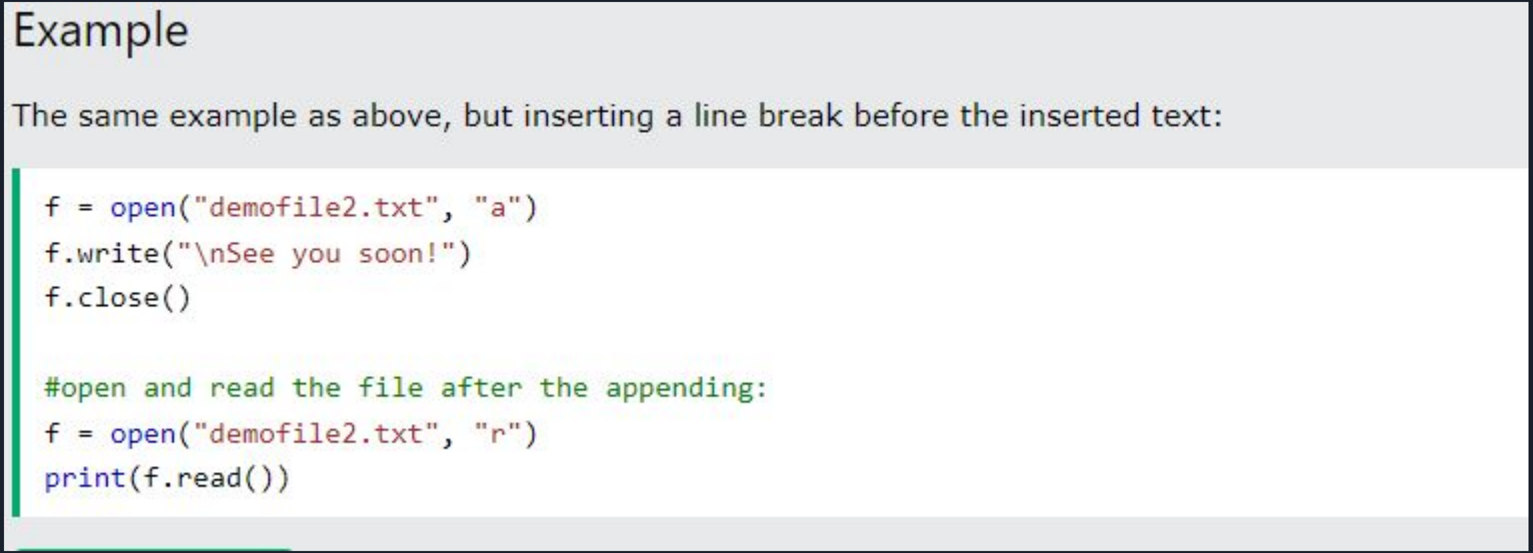


The `write()` method writes a specified text to the file.

Where the specified text will be inserted depends on the file mode and stream position.

`"a"`: The text will be inserted at the current file stream position, default at the end of the file.

`"w"`: The file will be emptied before the text will be inserted at the current file stream position, default 0.



Example

The same example as above, but inserting a line break before the inserted text:

```
f = open("demofile2.txt", "a")
f.write("\nSee you soon!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

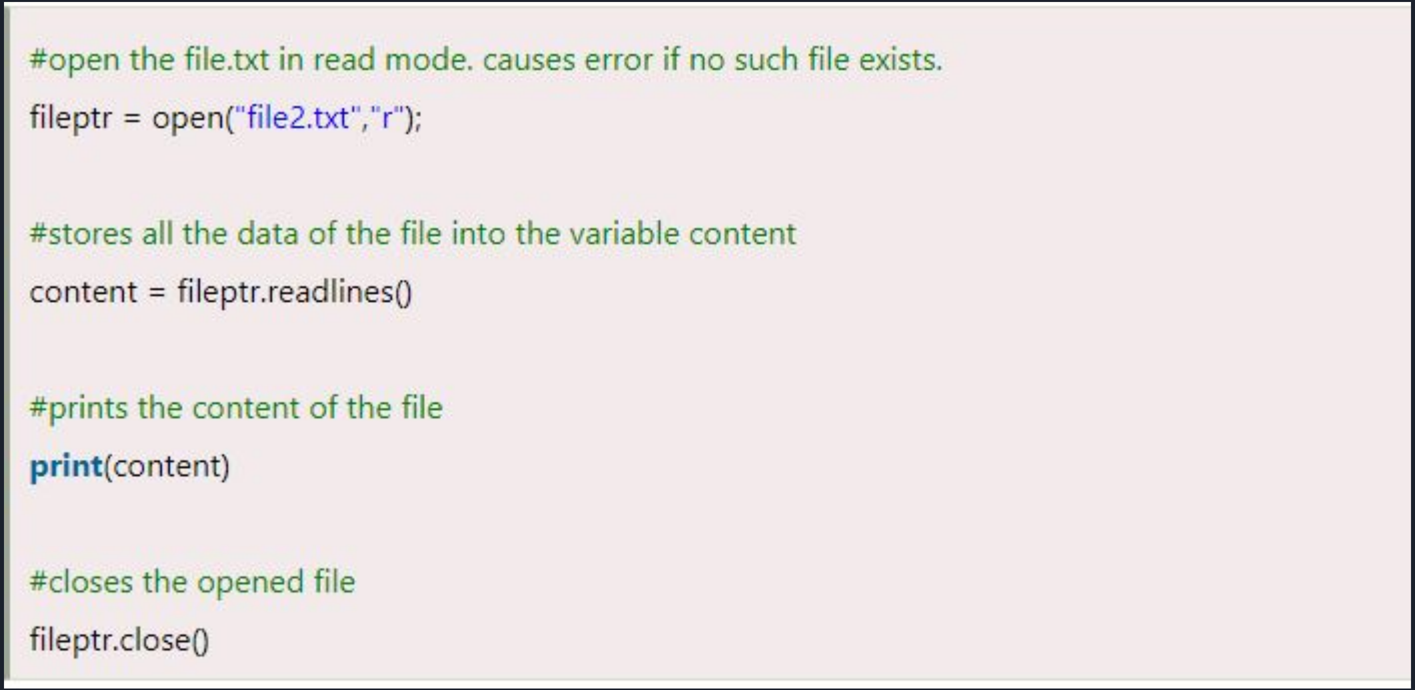
a : Append

It opens the file in the append mode. The file pointer exists at the end of the previously written file if exists any. It creates a new file if no file exists with the same name.

a+

It opens a file to append and read both. The file pointer remains at the end of the file if a file exists. It creates a new file if no file exists with the same name.

Example



```
#open the file.txt in read mode. causes error if no such file exists.
fileptr = open("file2.txt","r");

#stores all the data of the file into the variable content
content = fileptr.readlines()

#prints the content of the file
print(content)

#closes the opened file
fileptr.close()
```

Seek and tell()

- Seek()
- tell()

tell():

- In python programming, within file handling concept `tell()` function is used to get the actual position of file object.

- By file object we mean a cursor. And it’s cursor, who decides from where data has to be read or written in a file.

seek():

- In python programming, within file handling concept `seek()` function is used to shift/change the position of file object to required position.
- By file object we mean a cursor. And it’s cursor, who decides from where data has to be read or write in a file.