File Handling

Sidheswar Routray
Department of Computer Science & Engineering
School of Technology

What is File?

- **File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory . It is a collection of bytes.**

Operation in File

- **Open a file**
- **Read or write (perform operation)**
- **Close the file**

Open a file

- File_pointer = open(<file_name/path>, <access_mode>)

File mode

1	**r** **Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.**
2	**rb** **Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.**
3	**r+** **Opens a file for both reading and writing. The file pointer placed at the beginning of the file.**
4	**w** **Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.**
5	**w+** **Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.**
6	**a** **Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.**
7	**a+** **Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.**

Open() function

>>> **f = open("test.txt")** # open file in current directory, default mode is ‘r’ or ‘rt’

- This function returns a file object, also called a handle, as it is used to read or modify the file.
- The default is reading in text mode. In this mode, we get strings when reading from the file.

>>> **f = open("C:/Users/SR/Documents/README.txt")** #open file at given location

- Open() function gives an error if file doesn’t exist.

>>>**f = open("test.txt",'w')** # write in text mode

>>> **f= open("img.bmp",'r+b')** # read and write in binary mode

Close() function

- When we are done with operations to the file, we need to properly close the file and It is done using Python **close()** method.
- Closing a file will free up the resources that were tied with the file.
- Python has a **garbage collector** to clean up unreferenced objects but, we must not rely on it to close the file. For Example:

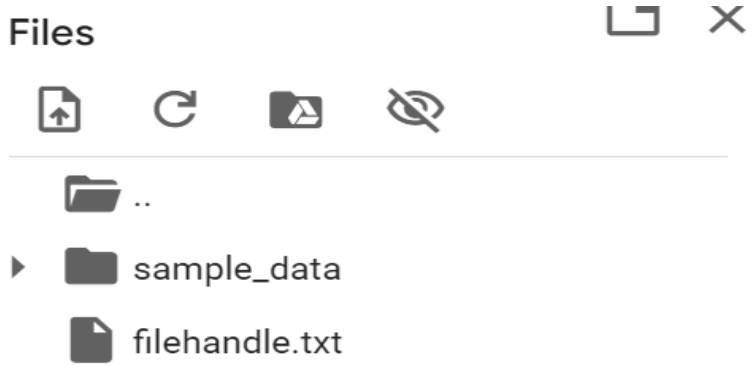
```
f = open("test.txt")
f.close()
```

Reading from file

- fp = open(" filehandle.txt", "r")
- print(fp.read())
- print(fp.read(5))
- print(fp.readline())
- fp.close()

filehandle.txt

Hello! Welcome to filehandle.txt
Reading the file is very easy.
Now you can read a file from your hard disk
Good Luck!




```
fp = open("filehandle.txt", "r")
print(fp.read())
print(fp.read(5))
print(fp.readline())
fp.close()
```

```
Hello! Welcome to filehandle.txt
Reading the file is very easy.
Now you can read a file from your hard disk
Good Luck!
```

Write into file

```
with open("test.txt",'w') as fp:
    fp.write("my first file\n")
    fp.write("This file\n\n")
    fp.write("contains three lines\n")
    fp.close()
```

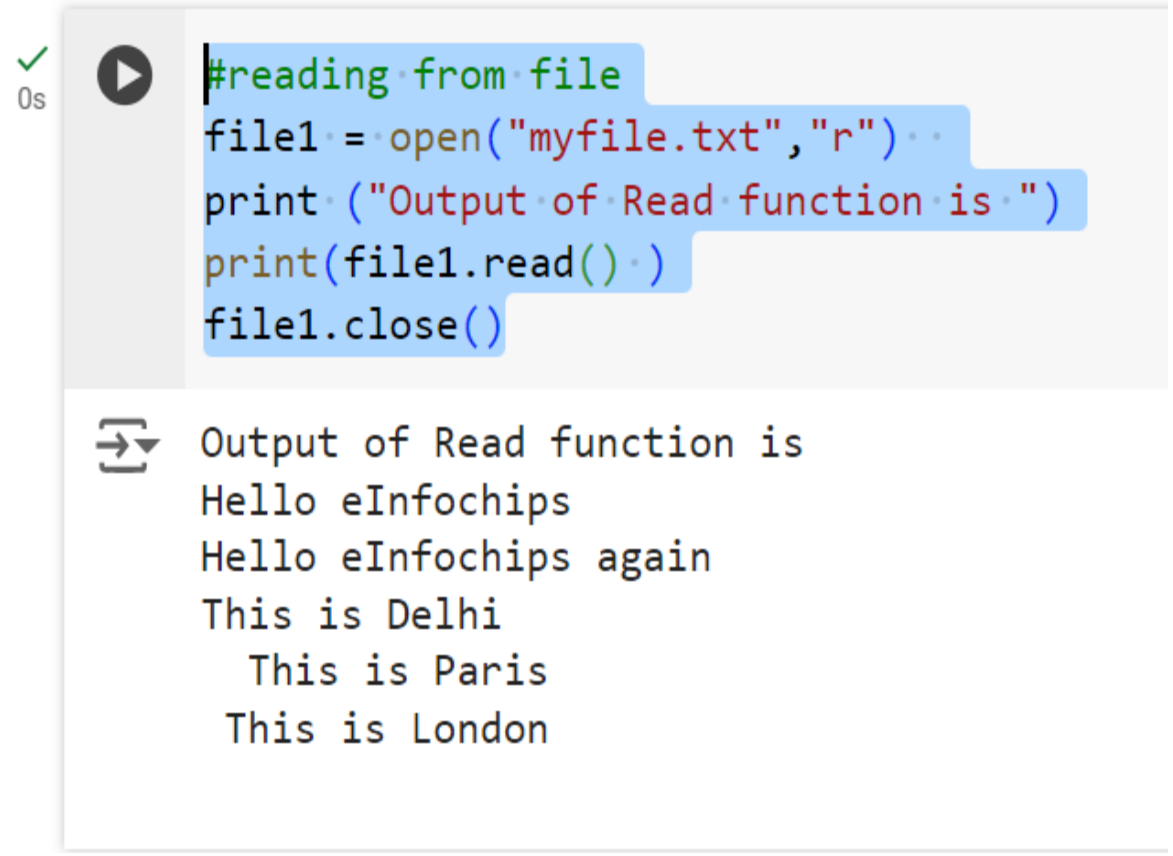
```
fp = open("test.txt",'w')
fp.write("my first file\n")
fp.write("This file\n\n")
fp.write("contains three lines\n")
fp.close()
```

- # write mode creates file if it desn't exists
- file1=open("myfile.txt","w")
- file1.write("Hello eInfochips\n")
- file1.close()

#append line in file
file1=open("myfile.txt","a")
file1.write("Hello eInfochips again\n")
file1.close()

#append multiple lines
file1=open("myfile.txt","a")
L = ["This is Delhi\n ","This is Paris \n ","This is London \n"]
file1.writelines(L)
file1.close()

#reading from file
file1 = open("myfile.txt","r")
print ("Output of Read function is ")
print(file1.read())
file1.close()



[6] 0s

```
# read line from file from the pointer currently exists , reads only one line
file1 = open("myfile.txt","r")
print(file1.readline())
```

```
Hello eInfochips
```

[9] 0s

```
print(file1.readline())
```

```
 This is Paris
```

How to read a specific line

- **# open the sample file used**
- file = open('filehandle.txt',"r")

 # read the content of the file opened

- content = file.readlines()

 # read 3rd line from the file

- print("3rd line")
- print(content[2])

 # print first 3 lines of file

- print("first three lines")
- print(content[0:3])

```
3rd line
Now you can read a file from your hard disk


first three lines
['Hello! Welcome to filehandle.txt\n', 'Reading the file is very easy.\n', 'Now you can read a file from your hard disk\n']
```

Few important function

fp.seek(0)

fp.tell()

Seek() method

- In Python, **seek()** function is used to change the position of the File Handle to a given specific position.
- File handle is like a cursor, which defines from where the data has to be read or written in the file.
- **syntax:** f.seek(offset, from_where), where f is file pointer

Parameters:

❖Offset: It refers to the new position of the file pointer within the file.

❖from_where: It indicates the reference position from where the bytes are to be moved.

The reference point is selected by the *from_where* argument. It accepts three values:

0: sets the reference point at beginning of the file

1: sets the reference point at current file position

2: sets the reference point at the end of the file

fp.tell(): The tell() method of python tells us the current position within the file

Cntd…

```
f = open("a.txt", 'w')
line = 'Welcome to python.mykvs.in\nRegularly visit
python.mykvs.in'
f.write(line)
f.close()
f = open("a.txt", 'rb+')
print(f.tell())
print(f.read(7)) # read seven characters
print(f.tell())
print(f.read())
print(f.tell())
f.seek(9,0) # moves to 9 position from begining
print(f.read(5))
f.seek(4, 1) # moves to 4 position from current location
print(f.read(5))
f.seek(-5, 2) # Go to the 5th byte before the end
print(f.read(5))
f.close()
```

OUTPUT

```
0
b'Welcome'
7
b' to python.mykvs.in\r\nRe gularly visit
python.mykvs.in‘
59
b'o pyt'
b'mykvs'
b'vs.in
```

Built-in File methods

- Built-in methods for file **other than** open(), close(), read(), readline(),readlines(), write(), writelines(), seek() are given below:

Method	Description
detach()	Separate the underlying binary buffer from the TextIOBase and return it.
fileno()	Return an integer number (file descriptor) of the file.
flush()	Flush the write buffer of the file stream.
isatty()	Return True if the file stream is interactive.
readable()	Returns True if the file stream can be read from.
seek(offset, from=SEEK_SET)	Change the file position to offset bytes, in reference to from (start, current, end).
seekable()	Returns True if the file stream supports random access.
tell()	Returns the current file location.
truncate(size=None)	Resize the file stream to size bytes. If size is not specified, resize to current location.
writable()	Returns True if the file stream can be written to.

Problem

- Write a program to copy contain of file X.txt to Y.txt

```
fpx=open('hello.txt','r')
fpy=open('new.txt','w+')
for line in fpx:
    fpy.write(line)
fpy.seek(0)
print(fpy.read())
fpx.close()
fpy.close()
```

Write a code to read file and give you summary of the file.

Number of words

Number of character (with space)

Number of character (without space)

Frequency of character

Number of paragraph