# Python - File I/O Operations

Here, you will learn how to read and write to the physical file in Python.

In Python, a physical file must be mapped to a built-in file object with the help of built-in function `open()`.

Signature:
file object = open(file name[, access mode][, buffersize])

In the `open()` method, the first parameter is the name of a file including its path. The access mode parameter is an optional parameter which decides the purpose of opening a file, e.g. read, write, append, etc. Use access mode 'w' to write data in a file and 'r' to read data. The optional buffersize argument specifies the file's desired buffer size: 0 means unbuffered, 1 means line buffered and other positive values indicate the buffer size. A negative buffersize uses the default value. If a file cannot be opened, then OSError or its subtype is raised.

The following table lists the valid values of mode parameters.

| **Access Modes** | **Description** |
|---|---|
| r | Opens a file for reading only. |
| rb | Opens a file for reading only in binary format. |
| r+ | Opens a file for both reading and writing. |
| rb+ | Opens a file for both reading and writing in binary format. |
| w | Opens a file for writing only. |
| wb | Opens a file for writing only in binary format. |
| w+ | Opens a file for both writing and reading. |
| wb+ | Opens a file for both writing and reading in binary format. |
| a | Opens a file for appending. |
| ab | Opens a file for appending in binary format. |
| a+ | Opens a file for both appending and reading. |
| ab+ | Opens a file for both appending and reading in binary format. |

## Writing to a File

The following example writes to a physical file.

Example: Writing to File
```
f=open("D:\myfile.txt","w")
f.write("Hello! Learn Python on TutorialsTeacher.")
f.close()
```

In the above example, the `f=open("myfile.txt","w")` statement opens `myfile.txt` in write mode, the `open()` method returns the file object and assigns it to a variable `f`. "w" specifies that the file should be writable. Next, we have to put certain data in the file. The `f.write("Hello! Learn Python on TutorialsTeacher.")` stores a string in the file. In the end, `f.close()` closes the file object.

When you run the above code, you will find "myfile.txt" created on the D drive of your computer. You can see the contents by opening it with an editor, like Notepad.

Python provides the `writelines()` method to save the contents of a list object in a file. Since the newline character is not automatically written to the file, it must be provided as a part of the string.

Example: Write Lines to File
```
lines=["Hello world.\n", "Welcome to TutorialsTeacher.\n"]
f=open("D:\myfile.txt","w")
f.writelines(lines)
f.close()
```

## Reading from a File

Three different methods are provided to read data from file.

1. readline(): reads the characters starting from the current reading position up to a newline character.
2. read(chars): reads the specified number of characters starting from the current position.
3. readlines(): reads all lines until the end of file and returns a list object.

Example: Reading Lines
```
f=open("myfile.txt","r")
line=f.readline()
print(line)
f.close()
```

As you can see, we have to open the file in 'r' mode. The `readline()` method will return a first line, and then will point to the second line in the file.

To read all the lines from a file, use the while loop as shown below.

Example: Reading Lines
```
f=open("D:\myfile.txt","r")
line=f.readline()
while line!='':
    print(line)
    line=f.readline()
```

### File Iterator

The file object has an inbuilt iterator. The following program reads the given file line by line until StopIteration is raised, i.e. the end of the file is reached.

Example: File Iterator

```
f=open("myfile.txt","r")
while True:
        try:
            line=next(f)
            print (line)
        except StopIteration:
            break
f.close()
```

Use the for loop to read a file easily.

Example: Read File using the For Loop

```
f=open("myfile.txt","r")
for line in f:
    print(line)
f.close()
```

## Append Text to a File

The "w" mode will always treat the file as a new file. In other words, an existing file opened with "w" mode will lose its earlier contents. In order to add more data to existing file use the "a" or "a+" mode.

Example: Append and Read a File

```
f=open("D:\myfile.txt","a+")
f.write("Hello! Learn Python on TutorialsTeacher.")
line=f.readline()
f.close()
```

Opening a file with "w" mode or "a" mode can only be written into and cannot be read from. Similarly "r" mode allows reading only and not writing. In order to perform simultaneous read/append operations, use "a+" mode.

### seek() method

To read or write at a specific position, use the `seek()` function to set the current read/write position.

```
f.seek(offset, from)
```

Here, the *from* parameter takes the following values:

- 0 : offset calculated from the beginning
- 1 : offset calculated from the current position
- 2 : offset calculated from the end

Assuming that myfile.txt contains "Hello World" text, the following example demonstrates the `seek()` method.

Example: seek()
```
f=open("D:\myfile.txt","r+")
f.seek(6,0)
lines=f.readlines()
for line in lines:
    print(line)
f.close()
```

The output of the above example is "World".

## Reading and Writing to a Binary File

The `open()` function opens a file in text format by default. To open a file in binary format, add 'b' to the mode parameter. Hence the "rb" mode opens the file in binary format for reading, while the "wb" mode opens the file in binary format for writing. Unlike text mode files, binary files are not human readable. When opened using any text editor, the data is unrecognizable.

The following code stores a list of numbers in a binary file. The list is first converted in a byte array before writing. The built-in function bytearray() returns a byte representation of the object.

Example: Write to a Binary File
```
f=open("binfile.bin","wb")
num=[5, 10, 15, 20, 25]
arr=bytearray(num)
f.write(arr)
f.close()
```

To read the above binary file, the output of the `read()` method is casted to a list using the `list()` function.

Example: Reading a Binary File
```
f=open("binfile.bin","rb")
num=list(f.read())
print (num)
f.close()
```

All methods of file object is given below:

| Method | Description |
|---|---|
| file.close() | Closes the file. |
| file.flush() | Flushes the internal buffer. |
| next(file) | Returns the next line from the file each time it is called. |
| file.read([size]) | Reads at a specified number of bytes from the file. |
| file.readline() | Reads one entire line from the file. |
| file.readlines() | Reads until EOF and returns a list containing the lines. |

| | |
|---|---|
| file.seek(offset, from) | Sets the file's current position. |
| file.tell() | Returns the file's current position |
| file.write(str) | Writes a string to the file. There is no return value. |