Files in Python

BY ROHIT S. AGRAWAL

Files

- Files are named locations on disk to store related information. They are used to permanently store data in a non-volatile memory.
- Since 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. In Python, a file operation takes place in the following order:

  Open a file

  Read or write (perform operation)

  Close the file

Opening Files

- 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.

```
>>> f = open("test.txt")    # open file in current directory

>>> f = open("C:/Python38/test.txt")  # specifying full path
```

- We can specify the mode while opening a file. In mode, we specify whether we want to read, write or append to the file.
- Generally we use letter ‘r’ for read mode, ‘w’ for Write mode and ‘a’ for append mode.

Creating a new file

- The new file can be created by using one of the following access modes with the function open().
- x: It creates a new file with the specified name. It causes an error if file exists with the same name.
- a: It creates a new file with the specified name if no such file exists. It appends the content to the file if the file already exists with the specified name.
- w: It creates a new file with the specified name if no such file exists. It overwrites the existing file.

Reading Files

To read a file in Python, we must open the file in reading ‘r’ mode.

There are various methods available for this purpose. read(), readline(), readlines().

We can mention how many characters to be read in the read() method. If we don’t mention , it reads and returns up to the end of the file.

```
>>> f = open("test.txt","r")

>>> f.read(4)    # read the first 4 data

'This'

>>> f.read(4)    # read the next 4 data

' is '
```

Writing to Files

- In order to write into a file in Python, we need to open it in write ‘w’, append ‘a’ or exclusive creation ‘x’ mode.
- Writing a string or sequence of bytes is done using the ‘write()’ method.
- This method returns the number of characters written to the file.

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

Closing Files

- 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 available in Python.

```
f = open("test.txt", "r")

# perform file operations

f.close()
```

- This method is not entirely safe. If an exception occurs when we are performing some operation with the file, the code exits without closing the file.
- A safer way is to use a try...finally block.

Closing Files

- This way, we are guaranteeing that the file is properly closed even if an exception is raised that causes program flow to stop.
- The best way to close 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.
- with open("test.txt", "r") as f:
- # perform file operations

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).

```
>>> f.tell()    # get the current file position

56

>>> f.seek(0)   # bring file cursor to initial position

0

>>> print(f.read())  # read the entire file
```