**Reading and Writing to text files in Python**

Python provides built-in functions for creating, writing, and reading files. Two types of files can be handled in Python, normal text files and binary files (written in binary language, 0s, and 1s).

* **Text files:**In this type of file, Each line of text is terminated with a special character called EOL (End of Line), which is the new line character (‘\n’) in Python by default.
* **Binary files:**In this type of file, there is no terminator for a line, and the data is stored after converting it into machine-understandable binary language.

This article will focus on opening, closing, reading, and writing data in a text file. Here, we will also see how to get Python output in a text file.

**Table of Content**

* [Opening a Text File](https://www.geeksforgeeks.org/reading-writing-text-files-python/#opening-a-text-file-in-python)
* [Read Text File](https://www.geeksforgeeks.org/reading-writing-text-files-python/#reading-from-a-file-in-python)
* [Write to Text File](https://www.geeksforgeeks.org/reading-writing-text-files-python/#write-to-text-file-in-python)
* [Appending to a File](https://www.geeksforgeeks.org/reading-writing-text-files-python/#appending-to-a-file-in-python)
* [Closing a Text File](https://www.geeksforgeeks.org/reading-writing-text-files-python/#closing-a-text-file-in-python)

**Opening a Text File in Python**

It is done using the open() function. No module is required to be imported for this function.

File\_object = open(r"File\_Name","Access\_Mode")

**Example**: Here, file1 is created as an object for MyFile1 and file2 as object for MyFile2.

# Open function to open the file "MyFile1.txt"

# (same directory) in append mode and

file1 = open("MyFile1.txt","a")

# store its reference in the variable file1

# and "MyFile2.txt" in D:\Text in file2

file2 = open(r"D:\Text\MyFile2.txt","w+")

**Python Read Text File**

There are three ways to read txt file in Python:

Using read()

Using readline()

Using readlines()

**Reading From a File Using read()**

**read():** Returns the read bytes in form of a string. Reads n bytes, if no n specified, reads the entire file.

File\_object.read([n])

**Reading a Text File Using readline()**

**readline():** Reads a line of the file and returns in form of a string.For specified n, reads at most n bytes. However, does not reads more than one line, even if n exceeds the length of the line.

File\_object.readline([n])

**Reading a File Using readlines()**

**readlines():** Reads all the lines and return them as each line a string element in a list.

File\_object.readlines()

**Note:**‘\n’ is treated as a special character of two bytes.

In this example, a file named “myfile.txt” is created and opened in write mode ( "w" ). Data is written to the file using write and writelines methods. The file is then reopened in read and append mode ( "r+" ). Various read operations, including read , readline , readlines , and the use of seek , demonstrate different ways to retrieve data from the file. Finally, the file is closed.

file1 = open("myfile.txt", "w")

L = ["This is Delhi \n", "This is Paris \n", "This is London \n"]

# \n is placed to indicate EOL (End of Line)

file1.write("Hello \n")

file1.writelines(L)

file1.close() # to change file access modes

file1 = open("myfile.txt", "r+")

print("Output of Read function is ")

print(file1.read())

print()

# seek(n) takes the file handle to the nth

# byte from the beginning.

file1.seek(0)

print("Output of Readline function is ")

print(file1.readline())

print()

file1.seek(0)

# To show difference between read and readline

print("Output of Read(9) function is ")

print(file1.read(9))

print()

file1.seek(0)

print("Output of Readline(9) function is ")

print(file1.readline(9))

file1.seek(0)

# readlines function

print("Output of Readlines function is ")

print(file1.readlines())

print()

file1.close()

**Output:**

Output of Read function is   
Hello   
This is Delhi   
This is Paris   
This is London   
Output of Readline function is   
Hello   
Output of Read(9) function is   
Hello   
Th  
Output of Readline(9) function is   
Hello   
Output of Readlines function is   
['Hello \n', 'This is Delhi \n', 'This is Paris \n', 'This is London \n']

**Write to Text File in Python**

There are two ways to write in a file:

Using write()

Using writelines()

**Writing to a Python Text File Using write()**

**write():** Inserts the string str1 in a single line in the text file.

File\_object.write(str1)

file = open("Employees.txt", "w")

for i in range(3):

name = input("Enter the name of the employee: ")

file.write(name)

file.write("\n")

file.close()

print("Data is written into the file.")

**Output:**

Data is written into the file.

**Writing to a Text File Using writelines()**

**writelines():**For a list of string elements, each string is inserted in the text file.Used to insert multiple strings at a single time.

File\_object.writelines(L) for L = [str1, str2, str3]

file1 = open("Employees.txt", "w")

lst = []

for i in range(3):

name = input("Enter the name of the employee: ")

lst.append(name + '\n')

file1.writelines(lst)

file1.close()

print("Data is written into the file.")

**Output:**

Data is written into the file.

**Appending to a File in Python**

In this example, a file named “myfile.txt” is initially opened in write mode ( "w" ) to write lines of text. The file is then reopened in append mode ( "a" ), and “Today” is added to the existing content. The output after appending is displayed using readlines . Subsequently, the file is reopened in write mode, overwriting the content with “Tomorrow”. The final output after writing is displayed using readlines.

file1 = open("myfile.txt", "w")

L = ["This is Delhi \n", "This is Paris \n", "This is London \n"]

file1.writelines(L)

file1.close()

# Append-adds at last

file1 = open("myfile.txt", "a") # append mode

file1.write("Today \n")

file1.close()

file1 = open("myfile.txt", "r")

print("Output of Readlines after appending")

print(file1.readlines())

print()

file1.close()

# Write-Overwrites

file1 = open("myfile.txt", "w") # write mode

file1.write("Tomorrow \n")

file1.close()

file1 = open("myfile.txt", "r")

print("Output of Readlines after writing")

print(file1.readlines())

print()

file1.close()

**Output:**

Output of Readlines after appending  
['This is Delhi \n', 'This is Paris \n', 'This is London \n', 'Today \n']  
Output of Readlines after writing  
['Tomorrow \n']

**Closing a Text File in Python**

Python close() function closes the file and frees the memory space acquired by that file. It is used at the time when the file is no longer needed or if it is to be opened in a different file mode. File\_object.close()

# Opening and Closing a file "MyFile.txt"

# for object name file1.

file1 = open("MyFile.txt","a")

file1.close()

**Reading and Writing to text files in Python – FAQs**

**File Access Modes in Python**

*Access modes govern the type of operations possible in the opened file. It refers to how the file will be used once it’s opened. These modes also define the location of the* ***File Handle*** *in the file. The file handle is like a cursor, which defines from where the data has to be read or written in the file and we can get Python output in text file.*

*There are 6 access modes in Python:*

* ***Read Only (‘r’):*** *Open text file for reading. The handle is positioned at the beginning of the file. If the file does not exist, raises the I/O error. This is also the default mode in which a file is opened.*
* ***Read and Write (‘r+’):*** *Open the file for reading and writing. The handle is positioned at the beginning of the file. Raises I/O error if the file does not exist.*
* ***Write Only (‘w’):*** *Open the file for writing. For the existing files, the data is truncated and over-written. The handle is positioned at the beginning of the file. Creates the file if the file does not exist.*
* ***Write and Read (‘w+’)****: Open the file for reading and writing. For an existing file, data is truncated and over-written. The handle is positioned at the beginning of the file.*
* ***Append Only (‘a’)****: Open the file for writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.*
* ***Append and Read (‘a+’):*** *Open the file for reading and writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.*

**How to read a file and write to another file in Python?**

*You can achieve this by opening two files: one for reading and another for writing, and then using appropriate methods to read from one file and write to another.*

*# Read from ‘input.txt’ and write to ‘output.txt’  
with open(‘input.txt’, ‘r’) as file\_in:  
with open(‘output.txt’, ‘w’) as file\_out:  
for line in file\_in:  
file\_out.write(line)*

**How Files are Loaded into Primary Memory?**

*There are two kinds of memory in a computer i.e. Primary and Secondary memory every file that you saved or anyone saved is on secondary memory causing any data in primary memory to be deleted when the computer is powered off. So when you need to change any text file or just to work with them in Python you need to load that file into primary memory. Python interacts with files loaded in primary memory or main memory through* ***“file handlers”*** *( This is how your operating system gives access to Python to interact with the file you opened by searching the file in its memory if found it returns a file handler and then you can work with the file).*

**What is the difference between reading and writing files in Python?**

* ***Reading files:*** *Involves methods ( read() , readline() , readlines() ) to retrieve data from a file.*
* ***Writing files:*** *Involves methods ( write() , writelines() ) to store data into a file.*

**Which function is used to read data from a text file?**

*The read() method is commonly used to read data from a text file in Python.*

*with open(‘file.txt’, ‘r’) as file:  
data = file.read()  
print(data)*