READING A FILE

08 November 2020 19:27

Reading text files in Python

Python provides inbuilt functions for creating, writing and reading files. There are two types of files that 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.

FILE ACCESS MODES

Access modes govern the type of operations possible in the opened file. It refers to how the file will be used once its opened. These modes also define the location of the **File Handle** in the file. File handle is like a cursor, which defines from where the data has to be read or written in the file. There are 6 access modes in python.

1. **Read Only ('r') :** Open text file for reading. The handle is positioned at the beginning of the file. If the file does not exists, raises I/O error. This is also the default mode in which file is opened.

2. **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 exists.

3. **Write Only ('w') :** Open the file for writing. For existing file, 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 exists.

4. **Write and Read ('w+')** : Open the file for reading and writing. For existing file, data is truncated and over-written. The handle is positioned at the beginning of the file.

5. **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.

6. **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.

Opening a File

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")
```

The file should exist in the same directory as the python program file else, full address of the file should be written on place of filename.

Note: The **r** is placed before filename to prevent the characters in filename string to be treated as special character. For example, if there is \temp in the file address, then \t is treated as the tab character and error is raised of invalid address. The r makes the string raw, that is, it tells that the string is without any special characters. The r can be ignored if the file is in same directory and address is not being placed.

```
# Open function to open the file "MyFile1.txt"
# (same directory) in append mode and
file1 = open("MyFile.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+")
```

Here, file1 is created as object for MyFile1 and file2 as object for MyFile2

Closing a file

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.

```
# Opening and Closing a file "MyFile.txt"
# for object name file1.
file1 = open("MyFile.txt", "a")
file1.close()
```

Writing to a file

There are two ways to write in a file.

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

2. **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]`

Reading from a file

There are three ways to read data from a text file.

1. **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])`

2. **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])`

3. **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

```
# Program to show various ways to read and
# write data in a file.
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
# bite 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']
```

Appending to a file

```
# Python program to illustrate
# Append vs write mode
file1 = open("myfile.txt", "w")
L = ["This is Delhi \n", "This is Paris \n", "This is London \n"]
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']
```