

Computational Thinking and Programming - 2

File Handling

Introduction to files, types of files (Text file, Binary file, CSV file), relative and absolute paths

Text file: opening a text file, text file open modes (r, r+, w, w+, a, a+), closing a text file, opening a file using with clause, writing/appending data to a text file using write() and writelines(), reading from a text file using read(), readline() and readlines(), seek and tell methods, manipulation of data in a text file

Introduction

So far in our python programs, the standard input comes from the keyboard and the output goes to the monitor i.e. the data is not stored permanently. This entered data is present in the memory as long as the program is running.

FILE HANDLING is a mechanism which allows us to store the data entered through python program permanently onto a disk file and later on retrieve the data by reading it from the disk files.

What is a File in Python?

A file is a sequence/series of bytes stored one after the other on the disk/permanent storage.

File handling in Python enables us to create, update, read, and delete the files stored on the file system through our python program.

There are mainly three types of files in Python:

1. Text Files .txt
2. Binary Files .dat
3. CSV Files .csv

Types of Data Files in Python

A **text file** consists of human readable characters, which can be opened by any text editor. It can be stored in any storage device. Text files are also known as Flat files.

Binary files are made up of non-human readable characters and symbols, which require specific programs to access its contents.

A **CSV file (Comma Separated Values)** is a type of plain text file that uses specific structuring to arrange tabular data. Because it is a plain text file, it can contain only actual text data - in other words, printable ASCII or Unicode characters.

Text Files

Text files are the files that contain text exactly in the same manner as it has been typed by the user. When we open the file in a text editor, we can see the text exactly in the same manner.

A text file can be understood as a sequence of characters consisting of alphabets, numbers and other special symbols. When we open a text file using a text editor (e.g., Notepad), we see several lines of text.

However, the file contents are not stored in such a way internally. Rather, they are stored in sequence of bytes consisting of 0s and 1s. Each line of a text file is terminated by a special character, called the End of Line (EOL). For example, the default EOL character in Python is the newline (`\n`). However, other characters can be used to indicate EOL.

Relative and Absolute paths

Absolute File Path is specified by a drive label and/or folders along with leading forward slash (folder separated by/ in respective order). For example, **/New.Project/File.Txt** or **c:/Myproject/File.Txt**

An absolute path describes how to access a given file or folder/directory, starting from the root of the file system. A file path is also called pathname. It is the complete address of the file.

Relative File Path is specified by directly referring to the file/folder without a leading forward slash. For **File.Txt** a relative path is interpreted from the perspective of current working directory. If you use a relative file path from the wrong directory/ folder, then the path will refer to a different file than you intend, or it will refer to no file at all.

Writing into a text file

PYTHON PROG
File1.py



TEXT FILE
DIARY.TXT

```
# CREATE A TEXT FILE  
# CODE TO CREATE A TEXT FILE  
# WRITE CONTENT IN THE TEXT FILE
```

Good Morning
Today is a lovely day
Let us begin with Text files today

Reading from a text file

TEXT FILE
DIARY.TXT



PYTHON PROG
File1.py

Good Morning
Today is a lovely day
Let us begin with Text files today

```
# CODE TO Read from  
# A TEXT FILE
```

Operations performed in a data file

1. **Opening a file :** To work with any file, we have to open the file by specifying its filename and the mode.
2. **Performing read / write operation :** Once the file is opened now we can either read the data from the file or write the data onto the file.
3. **Closing the file :** After performing the read/write operation we must close the file and release the file for other application to use it.

Opening and closing a data file (Alternate 1)

Opening a file

<File object> = **open**(**<Filename>**, **<mode>**)

<File operation statements>



w - write
r - read
a - append

Closing a file

<File object>.close()

```
F = open("Diary.txt", 'r')  
lines = F.read()  
print(lines)  
F.close()
```

Relative Path Name

```
F = open("C:/Documents/Project/Diary.txt", 'r')
```

Opening and closing a data file (Alternate 2)

Opening a file

`with open (<Filename>, <mode>) as Fileobject:` 
`<File operation statements>`
w - write
r - read
a - append

The advantage of using with clause is that any file that is opened using this clause is closed automatically, once the control comes outside the with clause. In case the user forgets to close the file explicitly or if an exception occurs, the file is closed automatically

```
with open("Diary.txt", 'r') as F:  
    lines = F.read()  
    print(lines)
```

File Modes

Files can be opened for writing data, reading data or appending the data to the existing file.

File Mode	Description
'r'	Opens the file in read-only mode. In case the file is not found, an exception <code>FileNotFoundException</code> is raised.
'w'	Opens the file in write mode. If the file already exists, all the contents will be overwritten. If the file doesn't exist, then a new file will be created.
'a'	Opens the file in append mode. If the file doesn't exist, then a new file will be created.

Opening a data file

Python's `open()` function creates a file object which serves as the link to the file residing on the computer. The file object is also known as the **file handle**. It is the tool through which a Python program can work with files stored on the hardware. All the functions that are performed on the data file are performed through file-objects.

The **first parameter** to the `open()` function is the path to the file you would like to open.

The **second parameter** to the `open()` function corresponds to the **mode** which is typically 'r', 'w' or 'a'. If no second parameter is given, then **by default it** opens in **read ('r') mode**.

Closing a data file

An open file is closed by calling the `close()` method of its file object.

The `close()` function breaks the link of the file- object and the file on the disk. After `close()`, no task can be performed on that file through the file handle.

When using the `with open....` method of opening the file, one does not need to explicitly use the `close()` function to close the file.

To create a text file with input from the user:

```
F = open("Diary.txt", 'w') # write mode  
  
while True:  
  
    str = input("Line: ")  
    if len(str)==0:  
        break  
    F.write(str+'\n')  
  
F.close() # After the data has been written to  
         # the file, it has to be closed.
```

One line is taken as input from the user, it is then written to the file as a string. If the length of the string input is 0 i.e. user has not entered anything, the loop will terminate. That means we keep on writing the strings in the file as long as the user does not press enter

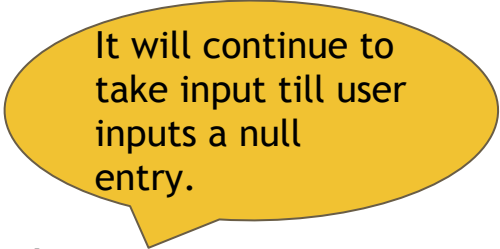
Writing/Appending to the file

The write() method takes a string as an argument and writes it to the text file.

```
F = open("Diary.txt", 'a')
while True:

    str = input("Line: ")
    if len(str)==0:
        break
    F.write(str+'\n')

F.close()
```



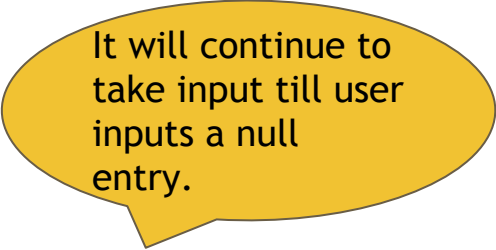
It will continue to take input till user inputs a null entry.

Line: Good Morning
Line: Today is a bright and sunny day!
Line: The weather is perfect
Line: Let us go out and play
Line:

Writing/Appending to the file

The write() method takes a string as an argument and writes it to the text file.

```
with open("Diary.txt", 'w') as F:
    while True:
        str = input("Line: ")
        if len(str)==0:
            break
        F.write(str+'\n')
```



It will continue to take input till user inputs a null entry.

Line: Good Morning

Line: Today is a bright and sunny day!

Line: The weather is perfect

Line: Let us go out and play

Line:

writelines() method

This method is used to write multiple strings to a file. We need to pass an iterable object like lists, tuple, etc. containing strings to the writelines() method.

```
with open("Diary.txt", 'w') as F:
    lines = ["Hello everyone\n", "Writing
             Multiline strings\n", "This is the
             third line"]
    F.writelines(lines)
with open("Diary.txt") as F:
    data = F.read()
print(data)
```

read() method - Reading a text file

There are three methods to read the contents from a text file.

Method 1 : Using the read() method

```
F = open('Diary.txt')  
line = F.read()  
print(line)  
F.close()
```

read() function will read the entire file and store(dump) it in a string **line**. The contents of the string are then printed on the screen.

After the data has been read from the file and processed, the file has to be closed

```
Good Morning  
Today is a bright and sunny day!  
The weather is perfect  
Let us go out and play
```

read() method

The read() method is also used to read a specified number of bytes of data from a data file. The syntax of read() method to read a specified number of bytes is:

```
file_object.read(n)
```

```
F = open('Diary.txt')  
line = F.read(5)  
print(line)  
line=F.read(10)  
print(line)  
F.close()
```

OUTPUT:

Good
Morning
To

Good Morning
Today is a bright and sunny
day!
The weather is perfect
Let us go out and play

readline() Method

Method 2 : Using the `readline()` method - reads one line of input

```
F=open("Diary.txt") # By default  
                    # read mode  
line1 = F.readline()  
print(line1)  
line2 = F.readline()  
print(line2)  
F.close()
```

OUTPUT:

Good Morning

Today is a bright and sunny day!

Readline() Method - Reading entire file

Method 2 : Using the readline() method

```
F = open("Diary.txt", "r")
line = F.readline()
while line:
    print(line)
    line = F.readline()
F.close()
```

OUTPUT:

```
Good Morning
Today is a bright and sunny day!
The weather is perfect
Let us go out and play
```

Readline() Method

`readline()` function can take in an integer as a parameter also but a `readline ()` function with an integer argument behaves in the same way as `read()` function with an integer argument. i.e.

`f.readline(n)` will read next `n` characters from the location of the file cursor in the file.

```
F=open("Diary.txt")  
line1 = F.readline(10)  
print(line1)  
F.close()
```


OUTPUT:

Good Morni

Readlines() Method

Method 3 : Using the `readlines()` method - reads the entire file and returns in a list.

```
F = open("Diary.txt", "r")
line = F.readlines()
print(line)
F.close()
```



`readlines()` helps to extract the entire content from the file and then store it as a list of strings where each line along with the newline character is stored as an element of the list

OUTPUT:

```
['Good Morning\n', 'Today is a bright and sunny day!\n', 'The weather is perfect\n', 'Let us go out and play\n']
```

Read a file using readlines()

Another way to read a file and display the contents of the file line by line using readlines()

```
with open("Diary.txt") as F:  
    lines = F.readlines()  
    for i in lines:  
        print(i,end=' ')  
F.close()
```

OUTPUT:

```
Good Morning  
Today is a bright and sunny day!  
The weather is perfect  
Let us go out and play
```

Python provides three types of read functions to read the data from a file : `read()`, `readline()` and `readlines()`.

Method	Syntax	Description
<code>read()</code>	<code><fileobject>.read([n])</code>	Reads at most next <code>n</code> bytes; if no <code>n</code> is specified, reads the entire file. Returns the read bytes in form of a string.
<code>readline()</code>	<code><fileobject>.readline([n])</code>	Reads a line of input; if <code>n</code> is specified, it reads at most <code>n</code> bytes. Returns the read bytes in form of a string; returns a blank string if no more bytes are left for reading in the file.
<code>readlines()</code>	<code><fileobject>.readlines()</code>	reads all lines and returns them in a list.

Program:

Write a function to read the data from a text file and then display each word terminated with a *.

```
def readtext(fname):  
    with open(fname) as F:  
        lines = F.readlines()  
        for line in lines:  
            for word in line.split():  
                print(word,end='*')  
            print()  
    F.close()  
readtext("Diary.txt")
```

OUTPUT:

```
Good*Morning*  
Today*is*a*bright*and*sunny*day!*  
The*weather*is*perfect*  
Let*us*go*out*and*play*
```

Write a menu driven program using functions to:

1. create a text file by taking input from the user
2. append data to the file
3. read and display the contents of the file using read()
4. read and display the contents of the file using readline()
5. read and display the contents of the file using readlines()

Program

Write a function to read the data from a text file and then count the number of vowels in the text file.

```
def readfile():
    F = open("Diary.txt")
    line=F.read()
    print(line)
    F.close()
def countvowel():
    with open("Diary.txt") as F:
        v=0
        while True:
            c=F.read(1)
            if c!="":
                if c in
"aeiouAEIOU":
                    v+=1
                else:
                    break
        print("Vowels : ", v)
readfile()
countvowel()
```

OUTPUT:

Good Morning
Today is a bright and sunny day!
The weather is perfect
Let us go out and play
Vowels : 26

Program

Write a function to read the data from a text file and then count the number of vowels in the text file.

```
def readfile():
    F = open("Diary.txt")
    line=F.read()
    print(line)
    F.close()
def countvowel():
    with open("Diary.txt") as F:
        v=0
        c=F.read()
        for i in c:
            if i in "aeiouAEIOU":
                v+=1
        print("Vowels : ", v)
readfile()
countvowel()
```

OUTPUT:

Good Morning
Today is a bright and sunny day!
The weather is perfect
Let us go out and play
Vowels : 26

Program:

Write a function to count the number of occurrences of the words 'me' or 'my' present in the text file "DIARY.TXT". If the "DIARY.TXT" contents are as follows: ©

My first book was Me and My Family.

It gave me chance to be Known to the world.

The output of the function should be: **Count of Me/My in file: 4**

```
def Me_My():
    F = open("Diary.txt")
    line=F.read()
    line=line.upper()
    ct=0
    for i in line.split():
        if i in ["ME","MY"]:
            ct=ct+1
    print("Count of Me/My in file:",ct)
    F.close()
CreateFile()
readfile()
Me_My()
```


Program:

```
def Me_My():  
    F = open("Diary.txt")  
    line=F.read()  
    line=line.upper()  
    ct=0  
    for i in line.split():  
        if i in ["ME","MY"]:  
            ct=ct+1  
    print("Count of Me/My in file:",ct)  
    F.close()  
CreateFile()  
Me_My()
```

OUTPUT:

Line: My first book was Me and My Family.
Line: It gave me chance to be Known to the world.
Line:
My first book was Me and My Family.
It gave me chance to be Known to the world.

Count of Me/My in the file are 4

Use try/except block (reading a file)

```
def Me_My(fname):  
    try:  
        F = open("Diary.txt")  
        line=F.read()  
        line=line.upper()  
        ct=0  
        for i in line.split():  
            if i in ["ME","MY"]:  
                ct=ct+1  
        print("Count of Me/My in file:",ct)  
        F.close()  
    except:  
        print("File ", fname, " does not exist")  
Me_My("abc.txt")
```

OUTPUT:

File abc.txt does not exist

Program:

Write a function to reverse and display each line of text from the text file.

```
def Reverse_Line():  
    F = open("Diary.txt")  
    lines=F.readlines()  
    for line in lines:  
        print(line[::-1],end=' ')  
    F.close()
```

Original File

Hello
How are you?
It is a bright day

Output

olleH
?uoy era woH
yad thgirb a si tl

Program:

Write a function to count the alphabets, digits and spaces in a text file.

```
def Count():  
    f = open("Diary.txt", "r")  
    lines = f.readlines()  
    a = d = s = 0  
    for line in lines:  
        for ch in line:  
            if ch.isalpha():  
                a+=1  
            elif ch.isdigit():  
                d+=1  
            elif ch == ' ':  
                s+=1  
    print("Alphabets = ", a)  
    print("Digits = ", d)  
    print("Spaces = ", s)  
    f.close()
```

CONTENTS OF THE FILE

Hello
How are you?
It is a bright day

OUTPUT:

Alphabets = 28
Digits = 0
Spaces = 6

Program:

Write a function to display alternate lines of a text file.

```
def Alternate_Lines():  
    f = open("Diary.txt", "r")  
    lines = f.readlines()  
    for i in range(0, len(lines), 2):  
        print(lines[i], end=' ')  
    f.close()
```

Original File

Hello
How are you?
It is a bright day

Output

Hello
It is a bright day

Program:

Write a program to display the contents of the file in reverse order (display the last line first).

```
def Reverse_Lines():  
    f = open("Diary.txt", "r")  
    lines = f.readlines()  
    for i in (lines[::-1]):  
        print(i, end=' ')  
    f.close()
```

Original File

Hello
How are you?
It is a bright day
Let us go and play

Output

Let us go and play
It is a bright day
How are you?
Hello

Program:

Write a function to display all the lines ending with 'o' or 'O' in a text file.

```
def Ending_With():  
    F = open("Diary.txt")  
    lines=F.readlines()  
    for line in lines:  
        if line[len(line)-1] in "oO":  
            print(line,end='')  
    F.close()
```

Original File

Hello
How are you?
It is a bright day

Output

Hello

Program:

Write the function definition for WORD4CHAR() to read the content of a text file FUN.TXT, and display all those words, which have 4 characters in it.

Example: If the content of the file Fun.TXT is as follows:

**When I was a small child, I used to play in the garden with my grand mom.
Those days were amazingly fun and I remember all the moments of that time.**

The function WORD4CHAR() should display the following:

When used play with days were that time

Program:

```
def Word4Char():  
    f = open("Diary.txt")  
    data = f.read()  
    data=data.split()  
    for word in data:  
        if word[-1] in ".,:;?":  
            word = word[:-1]  
        if len(word) == 4:  
            print(word,end= '  ')  
    f.close()
```

CONTENTS OF THE FILE:

Good Morning
Today is a bright and sunny day
Let us go and play
Stay safe

OUTPUT:

Good play Stay safe

Program:

Write the function to display the size of a text file after removing all the white spaces.

```
def RemoveSpaces():  
    F = open("Diary.txt")  
    data = F.read()  
    data.replace(" ", '')  
    data.replace("\n", "")  
    data.replace('\t', "")  
    length = len(data)  
    print("Number of characters are ", length)  
    F.close()
```

OUTPUT:

Number of
characters are 74

seek() and tell() methods

seek() method 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.

To change the file object's position, use **f.seek(offset, whence)**. The position is computed from adding *offset* to a reference point; the reference point is selected by the *whence* argument.

A **whence** value of **0** measures from the **beginning of the file**, **1** uses the **current file position**, and **2** uses the **end of the file** as the reference point. **whence** can be omitted and **defaults to 0**, using the **beginning of the file** as the reference point.

seek() method

Let us create another file called **Test.txt**

```
def CreateFile1():  
    with open("Test.txt", 'w') as F:  
        F.write("abcdefghi\n")  
        F.write("1234567890\n")  
        F.write("ABCDEFGHIJK\n")  
        F.write("1234567890\n")
```

seek() method

Let us create another file called **Test.txt**

```
def CreateFile1():  
    with open("Test.txt", 'w') as F:  
        F.write("abcdefghi\n")  
        F.write("1234567890\n")  
        F.write("ABCDEFGHIJK\n")  
        F.write("1234567890\n")
```

```
def DisplayFile1():  
    try:  
        with open("Test.txt") as F:  
            print("Contents are :")  
            for i in range(4):  
                F.seek(i)  
                print(F.read(i+1))  
    except:  
        print("File not found")
```

seek() method

```
def DisplayFile1():  
    try:  
        with open("Test.txt") as F:  
            print("Contents of the File are :")  
            for i in range(4):  
                F.seek(i)  
                print(F.read(i+1))  
    except:  
        print("File not found")
```

OUTPUT:

Contents of the File are :

a

bc bc

cde

defg

seek() and tell() method

```
def DisplayFile2():  
    try:  
        with open("Test.txt") as F:  
            print("Contents of the File are :")  
            for i in range(4):  
                F.seek(i)  
                print(F.read(i+1), F.tell())  
    except:  
        print("File not found")
```

OUTPUT:

```
Contents of the File are :  
a 1  
bc 3  
cde 5  
defg 7
```

The `tell()` function, returns the current position of the file pointer(cursor) in a stream.

seek() and tell() methods

Write a function to convert all uppercase characters to lowercase and all lowercase characters to their uppercase equivalent in a text file.

```
def upperlower():  
    f=open("Notes.txt", "r+")  
    while True:  
        pos=f.tell()  
        ch=f.read(1)  
        if ch.isupper():  
            ch=ch.lower()  
        elif ch.islower():  
            ch=ch.upper()  
        f.seek(pos)  
        f.write(ch)  
    f.close()
```


THANK YOU!

HAPPY LEARNING!!!