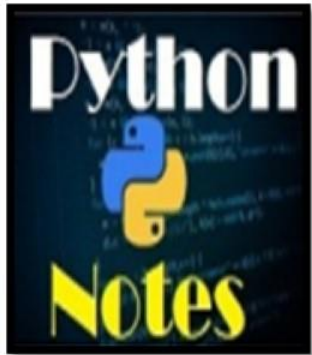


PYTHON : DATA FILE HANLING



File handling in Python

Opening files

01

Reading from files

02

Writing to files

03

04

Adding to files

Closing the files

05

By : Sangeeta M Chauhan , Gwalior

DATA FILES IN PYTHON

WHY DATA FILES ??????????????????

As we know whenever we enter data while running programs, it is not saved anywhere and we have to enter it again when we run the program again. So to store required data permanently on hard disk (Secondary Storage Device) we need to store it in File.

Note : File is a Stream or sequence of bytes /characters

Python Data Files can be of two types

- Text File (By default it creates file in **text Mode**)
- Binary File

Difference between Text and Binary Files

S.No.	Python Text File	Python Binary Files
1.	Consists Data in ASCII (Human readable form).	Consists Data in Binary form
1.	Suitable to store Unicode characters also	Suitable to store binary data such as images, video files , audio files etc.
1.	Each line in Text file is terminated with a Special character EOL(end of line)	There is no EOL character
1.	Operation on text files are slower than binary files as data is to be translated to binary	Operation on text files are faster as no translation required

Text files in Python

- Text files don't have any specific encoding and it can be opened in normal text editor itself.
- **Example:**
- **Web standards:** html, XML, CSS, JSON etc.
- **Source code:** c, app, js, py, java etc.
- **Documents:** txt, tex, RTF etc.
- **Tabular data:** csv, tsv etc.
- **Configuration:** ini, cfg, reg etc.

Binary files in Python

Most of the files that we see in our computer system are called binary files.

- **Examples:**
- **Document files:** .pdf, .doc, .xls etc.
- **Image files:** .png, .jpg, .gif, .bmp etc.
- **Video files:** .mp4, .3gp, .mkv, .avi etc.
- **Audio files:** .mp3, .wav, .mka, .aac etc.
- **Database files:** .mdb, .accde, .frm, .sqlite etc.
- **Archive files:** .zip, .rar, .iso, .7z etc.
- **Executable files:** .exe, .dll, .class etc.

All binary files follow a specific format. We can open some binary files in the normal text editor but we can't read the content present inside the file. That's because all the binary files will be encoded in the binary format, which can be understood only by a computer or machine.

For handling such binary files we need a specific type of software to open it.

Operations on File:

Whenever we worked with Data File in Python we have to follow sequence

- **Open/Create File**
- **Read from/Write to file**
- **Close File**

We can do following tasks/operations with python Data File.

- **Creation/Opening of an existing Data File**
- **Reading from file**
- **Writing to Data File**
- **Appending data (inserting Data at the end of the file)**
- **Inserting data (in between the file)**
- **Deleting Data from file**
- **Copying a file**
- **Modification/Updation in Data File**

Functions Used for File Handling

s.no.	Function Name	Syntax	Use
1	<code>open()</code>	<code>F_obj=open("File_name",mode)</code>	To Open or create a file in desired mode
2	<code>Close()</code>	<code>F_obj.close()</code>	To Close the file
3	<code>read()</code>	<code>F_obj.read()</code> or <code>F_obj.read(n)</code>	To read all or specified no. of characters from file
4	<code>readline()</code>	<code>F_obj.readline()</code> or <code>F_obj.readline(n)</code>	To read a single line or specified no. of characters from a line in a file
5	<code>readlines()</code>	<code>F_obj.readlines()</code>	To read all lines from a file and returns it in the form of list
6	<code>write()</code>	<code>F_obj.write(str)</code>	To write data (of string type) on to the file and return the numeric value (no. of characters written)
7	<code>writelines()</code>	<code>F_obj.writelines(LST)</code>	To Write Sequence (list/tuple etc) of strings in a file

Sr.No.	File Modes & Description
1	r Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
2	rb Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
3	r+ Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
4	rb+ Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.
5	w Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
6	wb Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.

7	w+ Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
8	wb+ Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
9	a Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
10	ab Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
11	a+ Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
12	ab+ Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

Read() Operation

```
print(".....Reading All Characters.....")
f=open("mydat.txt","r")
contents=f.read()
print(contents)
f.close()
```

Output

Python is one of those rare languages which is simple and powerful.

You will find it very easy to learn.

The official introduction to Python is:

Python is an easy to learn, powerful programming language.

It has efficient high-level data structures.

Read specific number of characters

```
print("\n\n.....Reading 70 Characters.....")  
f=open("mydat.txt","r")  
contents=f.read(70)  
print(contents)  
f.close()
```

Output

```
.....Reading 70 Characters.....  
Python is one of those rare languages which is simple and powerful.  
Yo
```

Reading single line

```
print("\n\n.....Reading a Single Line.....")
f=open("mydat.txt","r")
contents=f.readline()
print(contents)
f.close()
```

Output

```
.....Reading a Single Line.....  
Python is one of those rare languages which is simple and powerful.
```

Reading some characters from line

```
print("\n\n.....Reading 60 characters from line.....")
f=open("mydat.txt","r")
contents=f.readline(60)
print(contents)
f.close()
```


Output

```
.....Reading 60 characters from line.....  
Python is one of those rare languages which is simple and po
```

Read() Operation

```
print("\n\n.....Reading All Lines.....")
f=open("mydat.txt","r")
contents=f.readlines()
print(contents)
f.close()
```

Output

```
.....Reading All Lines.....
```

```
['Python is one of those rare languages which is simple and powerful.\n', 'You will find it very easy to learn.\n', 'The official introduction to Python is:\n', 'Python is an easy to learn, powerful programming language.\n', 'It has efficient high-level data structures.\n']
```

Write Operations

```
fh = open("Nw_File1.txt", "w")  
text = "Hello How are You "  
fh.write(text)
```

 Nw_File1.txt - C:\Users\Sangeeta Chauhan\AppData\Local\Programs\Python\Python36-32\Nw_File1.txt (3.6.5)

File Edit Format Run Options Window Help

Hello How are You

Write Operations

```
fh = open("Nw_File2.txt", "w")
text = ["First line ", "Second line", "Third line "]
fh.writelines(text)
fh.close()
```


 Nw_File2.txt - C:\Users\Sangeeta Chauhan\AppData\Local\Programs\Python\Python36-32\Nw_File2.txt (3.6.5)

File Edit Format Run Options Window Help

First line Second lineThird line

Write Operations

```
fh = open("Nw_File3.txt", "w")
text = ["First line\n ", "Second line\n", "Third line\n"]
fh.writelines(text)
fh.close()
```

 Nw_File3.txt - C:\Users\Sangeeta Chauhan\AppData\Local\Programs\Python\Python36-32\Nw_File3.txt (3.6.5)

File Edit Format Run Options Window Help

```
First line
Second line
Third line
```

Write Operations

```
fh = open("Nw_File4.txt", "w")
text = ("First line\n ", "Second line\n", "Third line \n")
fh.writelines(text)
fh.close()
```

Nw_File4.txt - C:\Users\Sangeeta Chauhan\AppData\Local\Programs\Python\Python36-32\Nw_File4.txt (3.6.5)

File Edit Format Run Options Window Help

First line

Second line

Third line

Write Operations

```
fh = open("Nw_File5.txt", "w")
text = {1:"First line\n ", 2:"Second line\n", 3:"Third line \n"}
fh.writelines(str(text))
fh.close()
```

Nw_File5.txt - C:\Users\Sangeeta Chauhan\AppData\Local\Programs\Python\Python36-32\Nw_File5.txt (3.6.5)

File Edit Format Run Options Window Help

```
{1: 'First line\n ', 2: 'Second line\n', 3: 'Third line \n'}
```




In the previous cases whenever we run the code again previously written contents will be overwritten. If we want to add contents after the previously added contents then we need to open file in append mode.

File Edit Format Run Options Window Help

```
f=open('StreamRecord.dat','a') # opening file in append mode
```

```
n=int(input('How many records y
```

```
for i in range(n):
```

```
    print('Record No:', i+1)
```

```
    rno=input('Enter Roll no')
```

```
    stream=input('Enter Opted s
```

```
    record=rno + " " + stream +
```

```
    f.write(record)
```

```
f.close()
```

```
RESTART: C:/Users/Sangeeta Chauhan/AppData/Local/Programs/Python/Python38-64/Scripts/python.exe leappend.py
```

```
How many records you want to enter2
```

```
Record No: 1
```

```
Enter Roll no 101
```

```
Enter Opted StreamScience
```

```
Record No: 2
```

```
Enter Roll no102
```

Reading the contents of above file

```
f=open('StreamRecord.dat','r')
rec=""
while rec:
    rec=f.readline()
    print(rec)
f.close()
```

output	
101	Science
102	Commerce
103	Commerce
104	COMmerce
105	Science
111	Science
112	Arts

File Edit Format Run Options Window Help

```
f=open('StreamRecord.dat','r')
```

```
rec=""
```

```
while rec:
```

```
    rec=f.readline()
```

```
    for d in rec.split():
```

```
        print(d)
```

```
f.close()
```

OUTPUT

```
>>>
```

```
RESTART: C
```

```
101
```

```
Science
```

```
102
```

```
Commerce
```

```
103
```

```
Commerce
```

```
104
```

```
Commerce
```

```
105
```

```
Science
```

```
111
```

```
Science
```

```
112
```

```
Arts
```

```
>>> |
```

Another way to read contents

```
f=open('StreamRecord.dat','r')
rec=""
rec=f.read()
print(rec)
f.close()
```

output

```
101 Science
102 Commerce
103 Commerce
104 Commerce
105 Science
111 Science
112 Arts
```

Seek and tell

- `Fileobject.tell()` : Python file method **tell()** returns the current position of the file read/write pointer within the file.

`fileObject.tell()`

- `Fileobject.tell()` : **seek()** is used to change current cursor position in **Python**. The **method seek()** sets the file's current position at the offset.

`fileObject.seek(offset[, whence])`

offset – This is the position of the read/write pointer within the file.

whence – 0 means absolute file positioning (default),
1 means seek relative to the current position and
2 means seek relative to the file's end.

seek_tell.py - C:/Users/Sangeeta Chauhan/AppData/Local/Programs/Python/Python36-32/seek_tell.py (3.6.5)

File Edit Format Run Options Window Help

```
f=open("ffile.txt","w")
f.write("Hello how are u")
f.write("I am fine here")
SZ=f.tell()
print("file size is ",SZ)
|
f.seek(20,0)
f.write(" \nPython is one of those rare languages\
which can claim to be both simple and powerful.\n \
You will find yourself pleasantly surprised to see how easy it is..")

SZ=f.tell()
print("NOw file size is ",SZ)
f.close()
```

```
RESTART: C:/Users/Sangeeta Chauhan/AppData/Local/Programs/Python/Python36-32/seek_tell.p
file size is 29
NOw file size is 177
>>> |
```

Opening file using With Open

With `open("file","mode")` as object

- With the "With" statement, you get better syntax and exceptions handling.
- In addition, it will automatically close the file. The with statement provides a way for ensuring that a clean-up is always used.

Example:

with `open("Myfile.txt")` as f:

Click to View



File handling programs

NOW CONTINUE WITH BINARY
FILE.....

How to Extract text from binary File

```
import PyPDF2

f_pdf=open("cs.pdf","rb")

f_txt=open("CS.txt","w")

pdf_con=PyPDF2.PdfFileReader(f_pdf)

print ("Total NO. of Pages in file are ", pdf_con.numPages)

pg_Obj=pdf_con.getPage(0)

contents=pg_Obj.extractText()

f_txt.write(contents)

f_txt.close()
```

For this program
we have to install -
→
pip install PyPDF2

Thank
you!

