
python

UNIT – IV

Topics to be covered:

- Introduction to Files,
- File Handling,
- Working with File Structure,
- Directories,
- Handling Directories

- Sometimes, it is not enough to only display the data on the console.
- The data to be displayed may be very large, and only a limited amount of data can be displayed on the console, and since the memory is volatile, it is impossible to recover the programmatically generated data again and again.
- However, if we need to do so, we may store it onto the local file system which is non volatile and can be accessed every time.
- Here, comes the need of file handling.
- Python provides the facility of working on Files.
- File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory(e. g. hard disk)
- Since, random access memory(RAM) is volatile which loses its data when computer is turned off, we use files for future use of the data.

- A File is an external storage on hard disk from where data can be stored and retrieved.
- A file is a chunk of logically related data or information which can be used by computer programs.

Types of Files:

There are 2 types of files

Text File: text files stores the data in the form of characters

eg: abc.txt

Binary File: binary files stores data in the form of bytes

eg: images, video files, audio files etc...

- 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 resources that are tied with the file are freed
- A file operation takes place in the following order:
    1. Open a file
    2. Read or write (perform operation)
    3. Close the file

Opening a File:

Python provides the open() function which accepts two arguments, file name and access mode in which the file is accessed.

The function returns a file object which can be used to perform various operations like reading, writing, etc.

Syntax:

fileObject=open(file_name,access_mode,buffering)

here,
file_name: It is the name of the file which you want to access.
access_mode: It specifies the mode in which File is to be opened.

- There are many types of mode like read, write, append.
- Mode depends the operation to be performed on File.
- Default access mode is read.

buffering:

- If the buffering value is set to 0, no buffering takes place**.**
- If the buffering value is 1, line buffering is performed while accessing a file

Example

```
f = open("python.txt",'w')                  # open file in current directory
f = open("/home/rgukt/Desktop/python/hello.txt")
                                            # specifying full path
```

Modes of file:

The allowed modes in Python are

1. r → open an existing file for read operation. The file pointer is positioned at the beginning of the file.If the specified file does not exist then we will get FileNotFoundError.This is default mode.
2. w → open an existing file for write operation. If the file already contains some data then it will be overridden. If the specified file is not already avaialble then this mode will create that file.

3. a → open an existing file for append operation. It won't override existing data.If the specified file is not already avaialble then this mode will create a new file.

4. r+ → To read and write data into the file. The previous data in the file will not be deleted.The file pointer is placed at the beginning of the file.

5. w+ → To write and read data. It will override existing data.

6. a+ → To append and read data from the file.It wont override existing data.

7. x → To open a file in exclusive creation mode for write operation. If the file already exists then we will get FileExistsError.

Note: All the above modes are applicable for text files. If the above modes suffixed with 'b' then these represents for binary files.

Eg: rb,wb,ab,r+b,w+b,a+b,xb

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.

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.

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.

rb+

Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.

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.

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.

Various properties of File Object:

Once we opend a file and we got file object,we can get various details related to that file by using its properties.

name → Name of opened file

mode → Mode in which the file is opened

closed → Returns boolean value indicates that file is closed or not

readable()→ Retruns boolean value indicates that whether file is readable or not

writable()→ Returns boolean value indicates that whether file is writable or not.

```
f=open("abc.txt",'w')
print("File Name:",f.name)
print("File Mode:",f.mode)
print("Is File Readable:",f.readable())
print("Is File Writable:",f.writable())
print("Is File Closed:",f.closed)
f.close()
print("Is File Closed:",f.closed)
```

```
>>>
File Name: abc.txt
File Mode: w
Is File Readable: False
Is File Writable: True
Is File Closed: False
Is File Closed: True
```

Closing Files in Python

When we are done with performing operations on the file, we need to properly close the file.

Python has a garbage collector to clean up unreferenced objects but we must not rely on it to 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.

Syntax: file object. close( )

```
f = open("test.txt")
f.close()
```

Here, the file represented by the file object f is closed. It means f is deleted from the memory. Once the file object is lost, the file date will become inaccessible.If we want to do any work with file again,we should once again open the the file using the open() function

Writing data to text files:

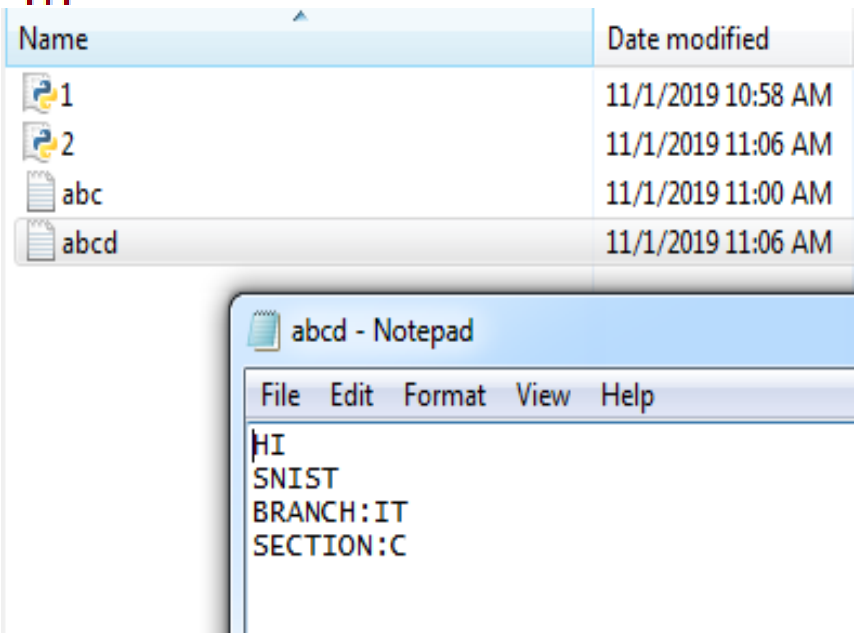
We can write character data to the text files by using thefollowing 2 methods.

write(str)

writelines(list of lines)

```
f=open("abcd.txt",'w')
f.write("HI\n")
f.write("SNIST\n")
f.write("BRANCH:IT\n")
f.write("SECTION:C\n")
print("Data written to the file successfully")
f.close()
```

```
Data written to the file successfully
>>>
```

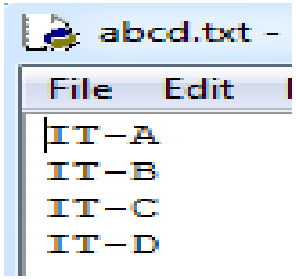


Note: In the above program, data present in the file will be overridden everytime if we run the program. Instead of overriding if we want append operation then we should open the file as follows.

```
f = open("abcd.txt","a")
```

```
f=open("abcd.txt",'w')
list=["IT-A\n","IT-B\n","IT-C\n","IT-D"]
f.writelines(list)
print("List of lines written to the file successfully")
f.close()
```

```
List of lines written to the file successfully
>>>
```



Note: while writing data by using write() methods, compulsory we have to provide line seperator(\n),otherwise total data should be written to a single line.

Reading Character Data from text files:

We can read character data from text file by using the following read methods.

read()→ To read total data from the file

read(n) → To read 'n' characters from the file

readline()→ To read only one line

readlines()→ To read all lines into a list

```
#To read total data from the file
f=open("abcd.txt",'r')
data=f.read()
print(data)
f.close()
```

```
>>>
IT-A
IT-B
IT-C
IT-D
>>>
```

abcd.txt - C:\Python34\filesprog\abcd.txt (3.4.3)

File Edit Format Run Options Window Help

```
SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY
BRANCH:IT
SECTION:C
```

```
#To read only first 10 characters:
f=open("abcd.txt",'r')
data=f.read(10)
print(data)
f.close()
```

```
>>>
SREENIDHI
>>>
```

```
#To read data line by line:
f=open("abcd.txt",'r')
line1=f.readline()
print(line1,end='')
line2=f.readline()
print(line2,end='')
line3=f.readline()
print(line3,end='')
f.close()
```

```
SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY
BRANCH:IT
SECTION:C
>>>
```

```
#To read all lines into list:
f=open("abcd.txt",'r')
lines=f.readlines()
for line in lines:
    print(line,end='')
f.close()
```

```
SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY
BRANCH:IT
SECTION:C
>>>
```

```
f=open("abcd.txt","r")
print(f.read(3))
print(f.readline())
print(f.read(6))
print("Remaining data")
print(f.read())
```

```
SRE
ENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY

BRANCH
Remaining data
:IT
SECTION:C
```

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 a 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

The with statement:

- The with statement can be used while opening a file.
- We can use this to group file operation statements within a block.
- The advantage of with statement is it will take care closing of file, after completing all operations automatically even in the case of exceptions also, and we are not required to close explicitly.

```
with open("abc.txt","w") as f:
    f.write("Durga\n")
    f.write("Software\n")
    f.write("Solutions\n")
    print("Is File Closed: ",f.closed)
print("Is File Closed: ",f.closed)
```

```
Is File Closed:  False
Is File Closed:  True
>>>
```

program to append text and display the contents of file on the screen

```
f1= open("myfile.txt","a+)
f1.append("God is great")
f1.seek(0,0)
print(f1.read())
f1.close()
```

Output: Hi how r u
I am fine
God is great

program to replace a word with other word in a file

```
f1=open("myfile.txt","r+")
str=f1.read()
str.replace("fine","good")
f1.write(str)
f1.close()
```

Output:Hi how r u
I am good

#**Program to read text from a file using for loop**

```
f1=open("myfile.txt","r")
for line in f1:
        print(line)
f1.close()
```

Output:

```
Hi how r u
I am fine
```

Program to copy the contents of one file to other file

```
f1=open("myfile.txt","r")
f2=open("mydata.txt","w")
s=f1.read()
f2.write(s)
print("File copied,open the file and see")
f1.close()
f2.close()
```

Program to count number of characters,words and lines in a text file.

```
cl=cw=cc=0
f1=open("myfile.txt","r")
for line in f1:
        cl+=1
        words=line.split()
        cw+=len(words)
        cc+=len(line)
print("Number of lines=",cl)
print("Number of words=",cw)
print("Number of characters=",cc)
f1.close()
```

Output:

Number of lines=2

Number of words=7

Number of characters=19

Working with binary files

```
f1=open('Tulips.jpg','rb')
f2=open('flower.jpg','wb')
byte=f1.read()
f2.write(byte)
f1.close()
f2.close()
```

File Pointer positions

tell():

- We can use tell() method to return current position of the cursor(file pointer) from
- beginning of the file. [can you please tell current cursor position]
- The position(index) of first character in files is zero just like string index.

```
f=open("abcd.txt","r")
print(f.tell())
print(f.read(2))
print(f.tell())
print(f.read(3))
print(f.tell())
```

seek():

- We can use seek() method to move cursor(file pointer) to specified location. [Can you please seek the cursor to a particular location]

f.seek(offset, fromwhere)

Here, offset represents the number of positions to move,fromwhere represents from which position to move and fromwhere can be 0,1 or 2

0---->From beginning of file(default value),1---->From current position,2--->From end of the file

Note: Python 2 supports all 3 values but Python 3 supports only zero.

```
f = open("foo.txt", "w")
f.write("abcdeghijklmnop")
f.close()
```

```
f=open("foo.txt","r")
f.seek(1)
print(f.read(5))
f.seek(2,0)
print(f.read(1))
f.close()
```

Output:bcdef

d

```
f = open("abc.txt", "wb")
f.write("Welcome to python language".encode())
f.close()
```

```
f=open("abc.txt","rb")
print(f.tell())                          0
print(f.read(5).decode())                Welco
f.seek(-2,1)
print(f.tell())                          3
print(f.read(3).decode())                com
f.seek(3,1)
print(f.tell())                          9
print(f.read(3).decode())                o p
f.seek(2,0)
print(f.tell())                          2
print(f.read(3).decode())                lco
f.seek(-8,2)
print(f.tell())                          18
print(f.read(5).decode())                langu
f.close()
```

Python os module

- The os module provides us the functions that are involved in file processing operations like renaming, deleting, etc.

How to check a particular file exists or not?

- We can use os library to get information about files in our computer.
- os module has path sub module, which contains isFile() function to check whether a particular file exists or not?
- **os.path.isfile(fname)**

Note:

- sys.exit(0) →To exit system without executing rest of the program.
- argument represents status code .
- 0 means normal termination and it is the default value.

```
#Write a program to check whether the given
 file exists or not. If it is
#available then print its content?
import os,sys
fname=input("Enter File Name: ")
if os.path.isfile(fname):
     print("File exists:",fname)
     f=open(fname,"r")
else:
     print("File does not exist:",fname)
     sys.exit(0)
print("The content of file is:")
data=f.read()
print(data)
```

```
Enter File Name: abcd.txt
File exists: abcd.txt
The content of file is:
PREMNADH
>>> ==============================
>>>
Enter File Name: prem.txt
File does not exist: prem.txt
>>>
```

Renaming the file

- The os module provides us the rename() method which is used to rename the specified file to a new name.
- The syntax to use the rename() method :

  rename(“current-name”, ”new-name”)

Ex:

```
import os;
os.rename("abc.txt",“snist.txt")
```

Removing the file

- The os module provides us the remove() method which is used to remove the specified file.
- The syntax to use the remove() method:

  remove(“file-name”)

Ex:

```
import os;
os.remove(“snist.txt")
```

What is Directory in Python?

- If there are a large number of files to handle in your Python program, you can arrange your code within different directories to make things more manageable.
- A directory or folder is a collection of files and sub directories.
- Python has the os module, which provides us with many useful methods to work with directories (and files as well).

Working with Directories:

- **It is very common requirement to perform operations for directories like**

1. To know current working directory
2. To create a new directory
3. To remove an existing directory
4. To rename a directory
5. To list contents of the directory etc...

Directories

A directory is a folder which holds other directories and sub directories.To work with directories,we have methods from os module.

Methods:

1.getcwd():It is used to get the path of current working directory.

Ex: import os

```
print(os.getcwd())
```

Output: C;\Users\Student\Desktop

2.mkdir():The mkdir() method is used to create the directories in the current working directory.

syntax to create the new directory :mkdir**(“directory name”)**

Ex:

```
import os;
#creating a new directory with the name new
os.mkdir("new")
#creating sub directory  sub
os.mkdir(‘new\sub’)
```

3.makedirs():It is used to create directory and a subdirectory in curtrent working directory.

Ex:
```
import os
os.makedirs("new/sub1/sub2")
```

4.chdir():The chdir() method is used to change the current working directory to a specified directory.

syntax to use the chdir() method: **chdir("new-directory")**

Ex:
```
import os;

#changing the current working directory to new
os.chdir("new")
```

5.rmdir():The rmdir() method is used to delete the specified directory.

syntax to use the rmdir() method: **os.rmdir("directory name")**

```
import os;
#removing the new directory
os.rmdir("new")
```

6.removedirs():It is used to remove all the directories given as argument.

Ex:
```
import os
os.removedirs(“new/sub1/sub2")
```

8.rename():It is used to rename a directory.

Ex:
```
import os
os.rename(“old”,”new”)
```

8.listdir():It is used to show the directories and files residing in current working directory.

Ex:
```
import os
print(os.listdir())
```