

*Data File Handling
in Python 3.7.4
Updated March
2020*

All rights reserved by Somnath PaulChoudhury

Update Version 3.2

Data can be stored in secondary storage so that it can be used time and again.

If we want to store something we need to open a file first then perform write operation finally we save the file and close the file. Sometimes we may just open the file to read it or add something more to it.

In Python we deal with Text and Binary files.

Text files stores in ASCII format or UNICODE format has End Of Line delimiters. Binary has no such property and everything is directly stored in machine readable format and size is also less.

Using **open()**

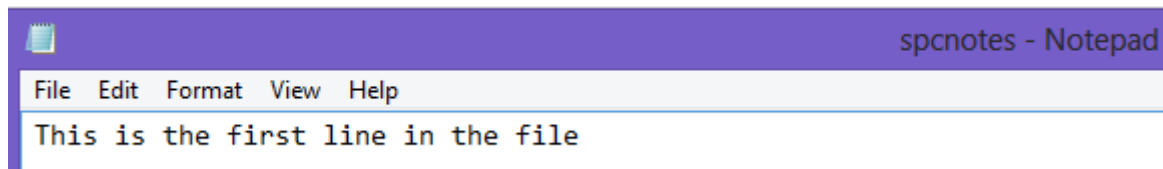
This is a function that can open a file. The name of the file it will open is the first argument. The second argument is the mode in which it will open the file. When we use the second argument as x the file if not exists already will be created. [although we may omit this in current version]

```
spcfile=open("spcnotes.txt","x")
spcfile.write("This is the first line in the file")
spcfile.close()
```

If we open the folder where the .py files are located (working directory) we can see the txt file



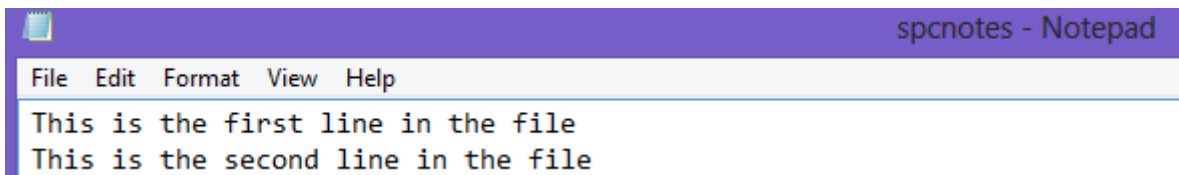
Click open the same and the line is written in it



If we want to add another line we must use the second argument as a (append)

```
spcfile=open("spcnotes.txt","a")
spcfile.write("This is the second line in the file")
spcfile.close()
```

If we click open the txt file spcnotes we see both the lines written



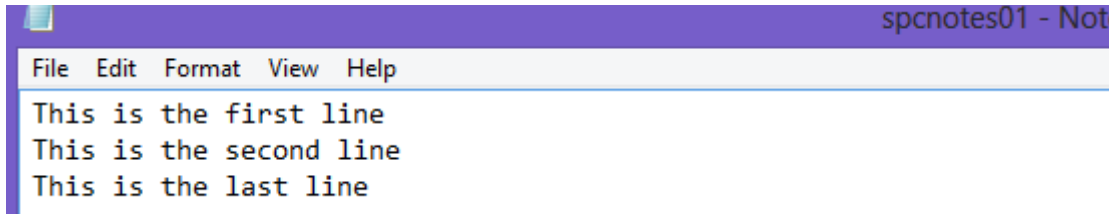
In the above we have used another function write(). It uses a string as a parameter and writes the string in the file that is open. Here w mode stands for write. It will overwrite all previous entry. We have another function writelines() that can write multiple lines. Consider the code below

```

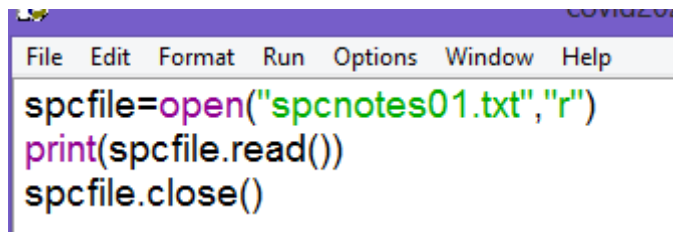
spcfile=open("spcnotes01.txt","w")
mylist=["This is the first line","\nThis is the second line","\nThis is the last line"]
spcfile.writelines(mylist)
spcfile.close()

```

If we run the script we create a file spcnotes01 and the entire content is written in the file



We can read the txt file created without clicking on the same by writing this code



When we run the script we display the content of the txt file spcnotes01

```

>>>
=====
This is the first line
This is the second line
This is the last line
>>>

```

Let us now try to read back any file from the disk. If the file exists it will display the lines, words and characters in it. If it doesn't exist it should be created and something should be written in it and finally same details will be displayed.

Here is the output of the code which is given in the next page

```

Enter a file name spcnotes01.txt
File already exists
This is the first line
This is the second line
This is the last line
Number of lines : 1 Number of words : 5 Number of characters : 23
Number of lines : 2 Number of words : 10 Number of characters : 47
Number of lines : 3 Number of words : 15 Number of characters : 68
>>>

```

The code

```
import os
spcfile=input("Enter a file name ")
lc=0
wc=0
cc=0
if(os.path.exists(spcfile)):
    print("File already exists")
    spc=open(spcfile)
    print(spc.read())
    with open(spcfile, 'r') as f:
        for line in f:
            lc += 1
            words=line.split()
            wc+=len(words)
            cc=cc+len(line)
            print("Number of lines : ",lc," Number of words : ",wc," Number of characters : ",cc)
else:
    spc=open(spcfile,"x")
    spc.write("This is the first line in the file")
    spc.close()
    spc=open(spcfile,"r")
    print(spc.read())
    with open(spcfile, 'r') as f:
        for line in f:
            lc+=1
            words=line.split()
            wc+=len(words)
            cc=cc+len(line)
            print("Line number is : ",lc," Cumulative number of words : ",wc," Cumulative number of characters : ",cc)
```

We need to import the os which stands for operating system. We can check the current working directory by typing as follows

```
>>> import os
>>> print(os.getcwd())
C:\Program Files\Hidden\Python37
```

The relative path starts from cwd the absolute path starts from the top most in the hierarchy. Sometimes due to internal settings you will not be able to save the .py files in cwd

```
You don't have permission to save in this location.
Contact the administrator to obtain permission.
Would you like to save in the App folder instead?
```

So you will save in the other locations...

Dealing with binary files

First lets start with the file modes,

We have used r as reading mode in txt files it will be rb for binary it stands for read binary.

Use w as writing mode in txt files it will be wb for binary files.

We can also use w+ and wb+ file modes in text and binary files. File mode w+ is opening a text file in both writing and reading mode. For binary we use wb+.

Lets go back to our previous program where we have written multiple lines in a text file using the function writelines, we will modify the program code and try to write and read back in a single program.

Lets try the code and see the output,

```

spcfile=open("spcnotes02.txt","w+")
mylist=["This is the first line","\nThis is the second line","\nThis is the last line"]
spcfile.writelines(mylist)
print(spcfile.read())
spcfile.close()

```

The output is shown below and nothing is displayed as we can see

```

===== REST,

```

```

>>>

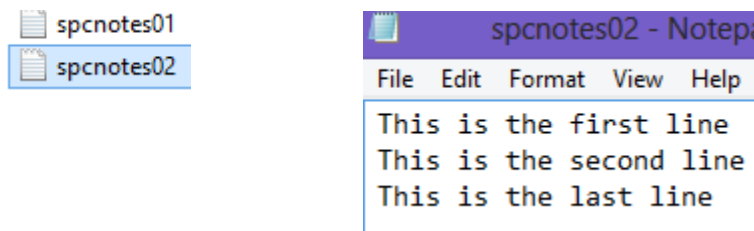
```

```

>>>

```

But the file spcnotes02.txt is created as shown here, so what went wrong?



Actually when we have written the last line, the current location of the file pointer is at the end of the file. There are two file pointers tell() and seek().

File pointer tell() tells us the current location and seek() can move the file pointer. The parameter 0 in seek() i.e. seek(0) can move the file pointer to the beginning. If we add an extra line now as shown below

```

spcfile=open("spcnotes02.txt","w+")
mylist=["This is the first line","\nThis is the second line","\nThis is the last line"]
spcfile.writelines(mylist)
spcfile.seek(0)
print(spcfile.read())
spcfile.close()

```

We get the desired output when we run the script

```

===== RES`
This is the first line
This is the second line
This is the last line

```

If we further add a line as shown

```

spcfile=open("spcnotes02.txt","w+")
mylist=["This is the first line","\nThis is the second line","\nThis is the last line"]
spcfile.writelines(mylist)
whereareu=spcfile.tell()
print(whereareu)
spcfile.seek(0)
print(spcfile.read())
spcfile.close()

```

The output is

```
70
This is the first line
This is the second line
This is the last line
>>>
```

Remember the total no of characters was 68 add the EOF delimiter character and consider next line due to print() function.

Can we just print the last line? Just try this

```
spcfile=open("spcnotes02.txt","w+")
mylist=["This is the first line","\nThis is the second line","\nThis is the last line"]
spcfile.writelines(mylist)
whereareu=spcfile.tell()
print(whereareu)
spcfile.seek(48,0)
print(spcfile.read())
spcfile.close()
```

seek(48,0) indicates to move 48 bytes from beginning, output is

```
70

This is the last line
```

How to print only the first line? Try this

```
spcfile=open("spcnotes03.txt","w+")
mylist=["This is the first line","\nThis is the second line","\nThis is the last line"]
spcfile.writelines(mylist)
spcfile.seek(0)
print(spcfile.read(23))
spcfile.close()
```

It reads 23 bytes from the beginning so we get the desired output

```
===== RESTART:
This is the first line
```

Want to print the second line?

```
spcfile=open("spcnotes03.txt","w+")
mylist=["This is the first line","\nThis is the second line","\nThis is the last line"]
spcfile.writelines(mylist)
spcfile.seek(24,0)
print(spcfile.read(23))
spcfile.close()
```

Output is



```
===== R
This is the second line
```

Lets us try to create a copy of the file spcnotes03.txt in spcnotes033.txt by writing a code. Then we will use the file mode a+ for appending and reading in the newly created file in another program.

Code to copy

```
spcfile_in=open("spcnotes03.txt","r") # opens the existing file in read mode
spcfile_out=open("spcnotes033.txt","w") # opens the new file in write mode
spc=spcfile_in.readline() # reads the first line
while(spc): # in the loop
    spcfile_out.write(spc) # writes a line in the new file
    spc=spcfile_in.readline() #reads again from the existing file
spcfile_in.close() # close both files ..
spcfile_out.close() # close
```

Its created

 spcnotes03	27-03
 spcnotes033	27-03

If we want to copy and display in the same program we may modify a little

```
spcfile_in=open("spcnotes03.txt","r") # opens the existing file in read mode
spcfile_out=open("spcnotes033.txt","w+") # opens the new file in write mode
spc=spcfile_in.readline() # reads the first line
while(spc): # in the loop
    spcfile_out.write(spc) # writes a line in the new file
    spc=spcfile_in.readline() #reads again from the existing file
spcfile_out.seek(0)
print(spcfile_out.read())
spcfile_in.close() # close both files ..
spcfile_out.close() # close
```

Output is

```
-----
This is the first line
This is the second line
This is the last line
```

Lets add a line at the end of the file spcnotes033.txt, the following code will do it

```
spcfile_out=open("spcnotes033.txt","a+") # opens the file in append and read mode
spcfile_out.write("\nThis is the fourth line which will be added at the end")
spcfile_out.close() # close
```

The output is

```
This is the first line
This is the second line
This is the last line
This is the fourth line which will be added at the end
```

Lets us try to read the file spcnotes033.txt again using with. If we use **with** statement all resources will be deallocated automatically once file operation is complete

```
with open("spcnotes033.txt","r") as spcfile:  
    print(spcfile.read())
```

We get all the lines displayed as,

```
This is the first line  
This is the second line  
This is the last line  
This is the fourth line which will be added at the end  
>>>
```

Finally lets start dealing with the binary files.

The write() and read() methods we have used with txt files uses string parameter. While working with the binary file conversion of data while reading and writing is required. Python has a module pickle for the purpose. Pickling is converting a data structure into byte stream for writing. Unpickling is the reverse, conversion of byte stream into original structure before reading.

This program writes some integers into a .dat file and reads back the same

```
import pickle  
spcfile = open("spcnotes04.dat","wb")  
while True:  
    x=int(input("Enter a number "))  
    pickle.dump(x,spcfile)  
    response=input("Continue? ")  
    if(response=="n" or response=="N"):  
        break  
spcfile = open("spcnotes04.dat","rb")  
try:  
    while True:  
        i=pickle.load(spcfile)  
        print(i)  
except EOFError:  
    pass  
spcfile.close()
```

Sample run is

```
Enter a number 12  
Continue? y  
Enter a number 54  
Continue? y  
Enter a number 78  
Continue? n  
12  
54  
78  
>>>
```


No part of this material should be used by any one. Only for private circulation amongst limited individuals. Please inform the author about unauthorized use. Tweet @somnathpc

Prevent COVID-19

Last 24 hours globally 61110 new cases of COVID-19 has been registered not to mention the undetected ones and has resulted in 3114 deaths globally (only last 24 hours)

Remember till 17th Feb 2020 both Italy and India had 3 registered cases each.

Today Italy is suffering other Nations are suffering. Probably it will affect us also on a large scale but we can only slow down by taking some measures that is widely circulated.

So stay safe, stay indoors, help yourself.