Python Programming

Module 10 - Files Input / Output

High performance. Delivered.

accenture

Strategy | Digital | Technology | Operations

Agenda

File handling

Operations on files and Directories

Module Objectives

At the end of this module, you will be able to understand

- File handling in Python - Operations
- File opening modes
- Fine Encoding
- File handling Functions
- Files and Directories Management in Python
- OS module

File Input /Output in Python

- 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).

Operations on Files

The following are the operations on files.

- Open a file
- Write on to the file
- Read from a file.
- Close the file

Opening a File

To Open a file

- ➢ use the built-in function open().
- ➢ This function returns a file object, also called a handle.
- ➢ This handle is used to read or modify the file.

Example:

```
fh = open("data.txt")    # open the file “data.txt” in current directory
# Opening the file “data.txt which is in C:/PythonPrograms Directory.
# Complete path must be specified
fh = open("C:/PythonPrograms/data.txt")
```

Opening a File (Contd..)

While opening the file, We can also specify the mode in which the file is to be opened

‘r’ for read ‘w ’for write or ‘a’ for append. Etc..

We can also specify whether we want to open the file in

<u>text mode or binary mode</u>.

The default is reading in text mode.

In text mode, we get strings when reading from the file.

In the binary mode , it returns bytes and this is the mode to be used when dealing with non-text files like image or exe files.

Modes of Opening a file in Python

Following table shows the various modes available in Python.

Mode	Description
'r'	Open a file for reading. (default)
'w'	Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
'x'	Open a file for exclusive creation. If the file already exists, the operation fails.
'a'	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
't'	Open in text mode. (default)
'b'	Open in binary mode.
'+'	Open a file for updating (reading and writing)

Modes of Opening a file in Python - Contd

Example:

```
f = open("data.txt")      # equivalent to 'r' or 'rt'
f = open("data.txt",'w')  # write in text mode
f = open("myImage.bmp",'r+b') # read and write in binary mode
```

Since the version 3.x, Python has made a clear distinction between str (text) and bytes (8-bits). Unlike other languages, the character 'a' does not imply the number 97 until it is encoded using ASCII (or other equivalent encodings).

Modes of Opening a file in Python - Contd

When working with files in text mode,

Specify the encoding type

Files are stored in bytes on the disk, we need to decode them into str when we read into Python.

Encoding is performed while writing texts to the file.

The default encoding is platform dependent.

In windows, it is 'cp1252'

In Linux it is 'utf-8'.

Note: Do not rely on the default encoding.

Example:

```
fh = open("data.txt",mode = 'r',encoding = 'utf-8');
```

Closing a File

After completing the desired operation on files, the file should be closed.

Even though, Python has a garbage collector to clean up unreferenced objects,we must not rely on it to close the file.

Closing a file will free up the resources that were tied with the file and is done using the close() method.

Example:

```
f = open("data.txt",encoding = 'utf-8')
# perform file operations
f.close()     # Closing the file
```

Closing a File (Contd)

This method is not entirely safe.

If an exception occurs when we are performing some operation with the file, the code exits without closing the file.

A safer way is to use a try...finally block.

Example:

```
try:
   f = open("data.txt",encoding = 'utf-8')
   # perform file operations
finally:
   f.close()
```

Closing a File (Contd)

This way, we are guaranteed that the file is properly closed even if an exception is raised, causing program flow to stop.

The best way to do this is using the with statement. This ensures that the file is closed when the block inside with is exited. We don't need to explicitly call the close() method. It is done internally.

Example

```
with open("test.txt",encoding = 'utf-8') as f:
   # perform file operations
```

Writing to a File

In order to write into a file we need to open it in write 'w', append 'a' or exclusive creation 'x' mode. We need to be careful with the 'w' mode as it will overwrite into the file if it already exists. All previous data are erased.

Writing a string or sequence of bytes (for binary files) is done using write() method.

This method returns the number of characters written to the file.

Example:

```
with open("data.txt",'w',encoding = 'utf-8') as fh:
   fh.write("This is my First File\n")
   fh.write("Accenture Services Pvt Ltd\n\n")
   fh.write("Bangalore , India\n")
```

This program will create a new file named 'data.txt' if it does not exist.
If it does exist, it is overwritten.
We must include the newline characters ourselves to distinguish different lines.

Reading From a File

To read the contents of a file, the file must be opened in reading mode.

There are various methods available for this purpose.

Input file Contents – data.txt

```
Hello,Welcome to Python Programing!!!. Accenture Services Pvt Ltd
Hign Performance Delivered... BDC , CDC , HDC , PDC , MDC , DDC , KDC.
Python is an Object Oriented Programming Language.
Python can be used for Web Development.
```

Reading from the file – read()

read() method

We can use the read(size) method to read in *size* number of data.

If *size* parameter is not specified, it reads and returns up to the end of the file.

- Example

```
import os
os.system('cls');
fh = open("data.txt",'r',encoding = 'utf-8');   #Opening the file for reading
str1 = fh.read(10);    # read the first 10 Characters
print("First 10 characters of the file 'data.txt' are : ",str1);
str1 = fh.read(10);    # read characters from 11th to 20th Positions
print("Characters from 11th-20th position from the file 'data.txt' are : ",str1);
str1 = fh.read(10);    # read characters from 21st to 30th Positions
print("Characters from 21th-30th position from the file 'data.txt' are : ",str1);
str1 = fh.read();    # read characters from 11st position till EOF
print("Characters from 31th till EOF from the file 'data.txt' are : ",str1);
```

Reading from a file – (Contd..)

```
OUTPUT
First 10 characters of the file 'data.txt' are :  Hello,Welc
Characters from 11th-20th position from the file 'data.txt' are :  ome to Pyt
Characters from 21th-30th position from the file 'data.txt' are :  hon
Progra
Characters from 31th till EOF from the file 'data.txt' are :  ming!!!
Accenture Services Pvt Ltd
Hign Performance Delivered...
BDC , CDC , HDC , PDC , MDC , DDC , KDC.
```

Note:
read() method returns newline as '\n'.
Once the end of file is reached, we get empty string on further reading.

Reading from a file – (Contd..)
Seek() & tell() Methods

- Using seek() method, we can change our current file cursor (position).
- tell() method returns our current position (in number of bytes).

Example: <u>seek() Method</u>

```
fh = open("data.txt",'r',encoding = 'utf-8');   #Opening the file for reading
str1 = fh.read(10);    # read the first 10 Characters
print("First 10 characters of the file 'data.txt' are : ",str1);
fh.seek(15);     # Change the cursor position from 11th Character to 15th Character
str1 = fh.read(10);    # read the 10 Characters from 15th Position
print("Reading 10 characters of the file 'data.txt' from 15th Position: ",str1);
```

<u>OUTPUT</u>

```
First 10 characters of the file 'data.txt' are :  Hello,Welc
Reading 10 characters of the file 'data.txt' from 15th Position:  o Python P
```

Reading from a file – (Contd..) - tell()

```
fh = open("data.txt",'r',encoding = 'utf-8');   #Opening the file for reading
str1 = fh.read(10);    # read the first 10 Characters
print("First 10 characters of the file 'data.txt' are : ",str1);
currentPosition = fh.tell();
print("Current Position of the 'file handle' after read(10) method is : ",currentPosition);
fh.seek(30);
currentPosition = fh.tell();
print("Current Position of the 'file handle' after seek(30) method is : ",currentPosition);
```

OUTPUT

```
First 10 characters of the file 'data.txt' are :  Hello,Welc
Current Position of the 'file handle' after read(10) method is :  10
Current Position of the 'file handle' after seek(30) method is :  30
```

Reading from a file – (Contd..)

- We can read a file line-by-line using a **for** loop.
- This is both efficient and fast.

Example 1:

```
fh = open("data.txt",'r',encoding = 'utf-8');   #Opening the file for
reading
for line in fh:
    print(line);
```

OUTPUT

```
Hello,Welcome to Python Programing!!!. Accenture Services Pvt Ltd

Hign Performance Delivered... BDC , CDC , HDC , PDC , MDC , DDC , KDC.

Python is an Object Oriented Programming Language.

Python can be used for Web Development.
```

Reading from a file – (Contd..)

Example 2:

```
fh = open("data.txt",'r',encoding = 'utf-8');   #Opening the file for
reading
for line in fh:
    print(line , end = '');
```

```
OUTPUT
Hello,Welcome to Python Programing!!!. Accenture Services Pvt Ltd
Hign Performance Delivered... BDC , CDC , HDC , PDC , MDC , DDC , KDC.
Python is an Object Oriented Programming Language.
Python can be used for Web Development.
```

Note
The lines in file itself has a newline character '\n'. Moreover,the `print()` function also appends a newline by default. Hence, we specify the *end* parameter to avoid two newlines when printing.

readline() & readlines() Methods

readline() method to read individual lines of a file.

This method reads a file till the newline, including the newline character.

readlines() method returns a list of remaining lines of the entire file.

All these reading method return empty values when end of file (EOF) is reached.

readline() - Example:

```
fh = open("data.txt",'r',encoding = 'utf-8');   #Opening the file for reading
print(fh.readline());
```

OUTPUT

```
Hello,Welcome to Python Programing!!!. Accenture Services Pvt Ltd
```

readlines() - Example:

```
fh = open("data.txt",'r',encoding = 'utf-8');   #Opening the file for reading
print(fh.readlines());
```

OUTPUT

```
['Hello,Welcome to Python Programing!!!. Accenture Services Pvt Ltd\n', 'Hign Pe
rformance Delivered... BDC , CDC , HDC , PDC , MDC , DDC , KDC.\n', 'Python is a
n Object Oriented Programming Language.\n', 'Python can be used for Web Development.']
```

Python File Methods

Following are the methods available with the file object

Method	**Description**
close()	Close an open file. It has no effect if the file is already closed.
detach()	Separate the underlying binary buffer from the TextIOBase and return it.
fileno()	Return an integer number (file descriptor) of the file.
flush()	Flush the write buffer of the file stream.
isatty()	Return True if the file stream is interactive.
read(n)	Read atmost n characters form the file. Reads till end of file if it is negative or None.
readable()	Returns True if the file stream can be read from.
readline(n=-1)	Read and return one line from the file. Reads in at most n bytes if specified.
readlines(n=-1)	Read and return a list of lines from the file. Reads in at most n bytes/characters if specified.
seek(offset,from=SEEK_SET)	Change the file position to offset bytes, in reference to from (start, current, end).

Python File Methods – (Contd)

Method	Description
seekable()	Returns True if the file stream supports random access.
tell()	Returns the current file location.
truncate(size=None)	Resize the file stream to size bytes. If size is not specified, resize to current location.
writable()	Returns True if the file stream can be written to.
write(s)	Write string s to the file and return the number of characters written.
writelines(lines)	Write a list of lines to the file.

Python Directory and Files Management

If there are large number of files in Python, we can place related files in 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).

Methods in *OS* module

Following are the methods in os module

Method	Description
getcwd()	This method returns the current working directory in the form of a string.
getcwdb()	Returns a bytes object containing the present working directory
chdir()	change the current working directory using the chdir() method. The new path that we want to change to must be supplied as a string to this method.
listdir()	This method takes in a path and returns a list of sub directories and files in that path
mkdir()	This method takes in the path of the new directory. If the full path is not specified, the new directory is created in the current working directory.
rename()	The rename() method can rename a directory or a file. The first argument is the old name and the new name must be supplies as the second argument.
remove()	Removes a file
rmdir()	Removes a Directory, Removes only empty Directory
rmtree()	Removes a non empty Directory also

OS Module Methods – Example – getcwd() & getcwdb()

getcwd() and getcwdb()

```
import os;
print("------- Present Working Directory getcwd() -----------");
pwd = os.getcwd()
print("Return value of getcwd() method = ",pwd);
print("------- Present Working Directory getcwdb() ----------");
pwd = os.getcwdb()
print("Return value of getcwdb() method = ",pwd);
```

OUTPUT

```
------- Present Working Directory getcwd() -----------
Return value of getcwd() method =  C:\PythonPrograms
------- Present Working Directory getcwdb() ----------
Return value of getcwdb() method =  b'C:\\PythonPrograms'
```

OS Module Methods – Example – chdir()

chdir() and listdir()

```
import os;
pwd = os.getcwd()
print("-------Present Working Directory = ",pwd);
os.chdir("c:\\users\\s.thangaraj");
print("-- Changing Directory from present to c:\\users\\s.thangaraj --");
pwd = os.getcwd()
print("-------Present Working Directory AFTER Changing Directory = ",pwd);
```

OUTPUT

```
-------Present Working Directory =  C:\PythonPrograms
-- Changing Directory from present to c:\users\s.thangaraj --
-------Present Working Directory AFTER Changing Directory =  c:\users\s.thangaraj
```

Note:
The extra backslash implies escape sequence.
The print() function will render this properly.

OS Module Methods – Example – listdir()

- Example – listdir()

```
#import os;
print("---- Listing the contents of the Directory listdir() --");
ContentsList = os.listdir();
for i in ContentsList:
                print(i);
```

OUTPUT

```
---- Listing the contents of the Directory listdir() --
data.txt
Date&Time.py
Dictionary.py
Files.py
FirstProgram.py
FunctionPrograms.py
If-Loops.py
Lists.py
Modules.py
mymodule.py
t1.py
```

OS Module Methods – Example – mkdir()

```
import os;
print("---- Listing the contents of the Directory --");
ContentsList = os.listdir();
for i in ContentsList:
            print(i);
os.mkdir('MyDir');     # Creating a new Directory MyDir
print("---- Listing the contents of the Directory AFTER os.mkdir('MyDir')");
ContentsList = os.listdir();
for i in ContentsList:
            print(i);
```

mkdir(<directoryname>)

OUTPUT

```
---- Listing the contents of the Directory --
data.txt
Date&Time.py
Dictionary.py
Files.py
FirstProgram.py
FunctionPrograms.py
If-Loops.py
Lists.py
Sets.py
t1.py
---- Listing the contents of the Directory AFTER
os.mkdir('MyDir')
data.txt
Date&Time.py
Dictionary.py
Files.py
FirstProgram.py
FunctionPrograms.py
If-Loops.py
Lists.py
Modules.py
MyDir
t1.py
```

OS Module Methods – Example – rename()

```
import os;
print("---- Listing the contents of the Directory BEFORE Renaming MyDir--");
ContentsList = os.listdir();
for i in ContentsList:
		print(i);
print("------------------Renaming a Directory ---------------");
os.rename("MyDir","MyNewDir");    # Renaming Directory MyDir to MyNewDir
print("---- Listing the contents of the Directory AFTER
os.rename('MyDir','MyNewDir')");
ContentsList = os.listdir();
for i in ContentsList:
		print(i);
```

OS Module Methods – Example – rename()

OUTPUT

```
---- Listing the contents of the Directory BEFORE Renaming MyDir--
data.txt
Date&Time.py
Dictionary.py
Files.py
FirstProgram.py
FunctionPrograms.py
If-Loops.py
Lists.py
Modules.py
MyDir
t1.py
-------------------Renaming a Directory ---------------
---- Listing the contents of the Directory AFTER os.rename('MyDir','MyNewDir')
data.txt
Date&Time.py
Dictionary.py
Files.py
FirstProgram.py
FunctionPrograms.py
If-Loops.py
Lists.py
Modules.py
mymodule.py
MyNewDir
t1.py
```

OS Module Methods – Example – remove()

```
import os;
print("---- Listing the contents of the Directory BEFORE Removing the file 't1.py' ---");
ContentsList = os.listdir();
for i in ContentsList:
		print(i);
print("------------------Removing the file t1.py ---------------");
os.remove('t1.py');    # Removing the file t1.py
print("---- Listing the contents of the Directory AFTER os.remove('t1.py')");
ContentsList = os.listdir();
for i in ContentsList:
		print(i);
```

OS Module Methods – Example – remove()

OUTPUT

```
---- Listing the contents of the Directory BEFORE Removing the file 't1.py' ---
data.txt
Date&Time.py
Dictionary.py
Files.py
FirstProgram.py
FunctionPrograms.py
If-Loops.py
Lists.py
Modules.py
mymodule.py
MyNewDir
t1.py
-------------------Removing the file t1.py ---------------
---- Listing the contents of the Directory AFTER os.remove('t1.py')
data.txt
Date&Time.py
Dictionary.py
Files.py
FirstProgram.py
FunctionPrograms.py
If-Loops.py
Lists.py
Modules.py
mymodule.py
MyNewDir
```

OS Module Methods – Example – rmdir()

```
import os;
print("---- Listing the contents of the Directory BEFORE Removing the Directory 'MyNewDir' ---");
ContentsList = os.listdir();
for i in ContentsList:
                print(i);
print("------------------Removing the Directory MyNewDir ---------------");
os.rmdir('MyNewDir');    # Removing the Directory  MyNewDir
print("---- Listing the contents of the Directory AFTER os.rmdir('MyNewDir')");
ContentsList = os.listdir();
for i in ContentsList:
                print(i);
```

OS Module Methods – Example – rmdir()

```
OUTPUT
---- Listing the contents of the Directory BEFORE Removing the Directory 'MyNewD
ir' ---
data.txt
Date&Time.py
Dictionary.py
Files.py
FirstProgram.py
FunctionPrograms.py
If-Loops.py
Lists.py
Modules.py
mymodule.py
MyNewDir

-------------------Removing the Directory MyNewDir ---------------
---- Listing the contents of the Directory AFTER os.rmdir('MyNewDir')
data.txt
Date&Time.py
Dictionary.py
Files.py
FirstProgram.py
FunctionPrograms.py
If-Loops.py
Lists.py
Modules.py
mymodule.py
```

OS Module Methods – Example – rmtree()

rmdir() method can only remove empty directories. Use the rmtree() method inside the shutil module to remove non-empty directory

Example:

```
import shutil
shutil.rmtree('MyNewDir')
```

Questions