Marwadi University

01CE1705-Programming with Python

Unit-5 File Handling

Prof. Chetan Chudasama
Computer Engineering Department

Outline

- Why File Handling
- What is Files?
- Read and Write functions
- Types of files
- Seek and tell function
- Other File Operation

File Handling

- Till now, we were taking the input from the console and writing it back to the console to interact with the user.
- 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 since the memory is volatile, it is impossible to recover the programmatically generated data again and again.
- The file handling plays an important role when the data needs to be stored permanently into the file.

Files:

- Files is the collection of data that is available to program. We can retrieve and use data stored in a file after the program termination.

Advantages:

- Stored data is permanent unless someone removes it.
- Stored data can be shared.
- It is possible to remove or update data.

- Each line of a file is terminated with special character like , or \n.It ends the current line and tells the interpreter a new one has begun.
- 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 the resources that are tied with the file are freed.
- Hence, in Python, a file operation takes place in the following order:

  Open a file

  Read or write (perform operation)

  Close the file

Note:-For small application we use file and for large application we use database.

Opening a File

- Before performing any operation on the file like reading or writing, first, we have to open that file.
- Python has a built-in open() function to open a file.
- It takes two arguments, file name and access mode.

Example:-

f=open(“marwadiuni.txt”,”r”)

f=open(“D://college/marwadiuni.txt”,”r”)#full path

- The function returns a file object or file pointer which can be used to perform various operations like reading, writing etc.
- The files can be accessed using various modes like read, write, or append. The following are the details about the access mode to open a file.

Mode	Description
r	Opens a file for reading. The file is by default open in this mode if no access mode is passed.
w	Opens a file for writing. It overwrites the file if previously exists or create a new one if no file exists with same name.
a	Opens a file for appending at the end of the file. Creates a new file if it does not exist. It wont override existing data.
x	It creates a new file with the specified name. It generate an error a file exists with the same name.(only write operation, not support read)
b	Opens in binary mode.
r+	Opens for reading and writing
w+	Opens for writing and reading. It will overwrite existing data.
a+	Opens for appending then reading. It won't overwrite existing data.

Example:

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

- When working with files in text mode, it is highly recommended to specify the encoding type.

Example:

```
f = open("test.txt", mode='r', encoding='utf-8’)
```

- Default encoding is platform dependent. In windows it is cp1252 but utf-8 in Linux. So we should not depend on default encoding or else our code will behave differently in different platform.

Writing to Files in Python

- In order to write into a file in Python, we need to open it in write w, append a or exclusive creation x mode.
- We need to be careful with 'w' mode, as it will overwrite into the file if it already exists. Due to this, all the previous data are erased.
- Writing a string or sequence of bytes (for binary files) is done using the write() method.
- This method returns the number of characters written to the file.

Example:

```
with open("subjectName.txt",'w',encoding='utf-8') as f:
    f.write("Python Programming\n")
    f.write("Advanced Web Technology\n\n")
    f.write("Artificial Intelligence")
```

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

Reading Files in Python

- To read a file using the Python script, the Python provides the read() method.
- It reads a string from the file.
- It can read the data in the text as well as a binary format.

Syntax:fileObj.read(count)

- Here, count is the number of bytes to be read from the file.
- If the count is not specified, then it may read the content of the file until the end.

Example:

```
f=open("subjectName.txt","r")
content=f.read(6)#If We use f.read() then it will print all content of the file
print(content)
print(type(content))
```

Output:

Python

<class 'str’>

- In the above code, we have read the content of subjectName.txt by using the read() function.
- We have passed count value as six which means it will read first six character from the file.
- We can read a file line-by-line using a for loop. This is efficient and fast.

Example:

```
f=open("subjectName.txt","r")
for line in f:
        print(line,end='')
```

Output:

Python Programming

Advanced Web Technology

Artificial Intelligence

- In this program, the lines in the file itself include a newline character \n. So, we use the end parameter of the print() function to avoid two newlines when printing.
- Python gives facility to read the file line by line using readline() function. The readline() method reads the lines of the file from the beginning, i.e., if we use the readline() method three times, then we can get the first three lines of the file.

Example:

```
f=open("subjectName.txt","r")
f.readline()#print Python Programming
f.readline()#print Advanced Web Technology
f.readline()#print Artificial Intelligence
f.readline()#print empty string
```

- Python provides also the readlines() method which is used for the reading lines. It returns the list of the lines till the end of file(EOF) is reached.

Example:

```
f=open("subjectName.txt","r")
print(f.readlines())
f.close()
```

Output:

['Python Programming\n','Advanced Web Technology\n','Artificial Intelligence']

Closing a file

- When we are done with performing operations on the file, we need to properly 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.
- Python has a garbage collector to clean up unreferenced objects but we must not rely on it to close the file.

Example:

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

- After closing the file, we cannot perform any operation in the file. The file needs to be properly closed. If any exception occurs while performing some operations in the file then the program terminates without closing the file.
- We should use the following method to overcome such type of problem.

```
try:
        f=open("marwadiuni.txt")
finally:
        f.close()
```

- The best way to close a file is by using the with statement. This ensures that the file is closed when the block inside the with statement is exited.
- It is always suggestible to use the with statement in the case of files because, if the break, return, or exception occurs in the nested block of code then it automatically closes the file, we don't need to write the close() function. It doesn't let the file to corrupt.

Example:

```
with open("marwadiuni.txt","r") as f:
        content=f.read()
        print(content)
```

- We don't need to explicitly call the close() method. It is done internally.

Types of Files

- There are two types of files in Python:
  1. Text File
  2. Binary File

Text File:-

- Text file are used to store characters or strings.
- It contains ASCII characters and Unicode characters.
- It consume extra memory. Example:-.txt,.doc
- File Mode:-r, w, a, r+, w+, a+

Binary File:-

- Binary file store entire data in the form of bytes.
- Binary file can be used to store image, audio and video.
- It is also known as Non-ASCII file or Non-Unicode file.

- It consume less memory.
  Example:-.exe,.mp4,.gif
  File Mode:-rb, wb, ab, rb+, wb+, ab+

Binary file operation

- Python uses pickle module for binary operation.
- Pickle module has dump() function to write into binary file.
- Pickle module has load() method to read from binary file.
- Conversion of python objects into byte stream is called pickling or serialization
- Conversion of Byte stream into python object is called unpickling or de-serialization.

Example: How to perform write operation in Binary file

```
import pickle
file=open("xyz.dat","wb")
pickle.dump({"name":"vaibhav","age":30},file)
file.close()
```

Example: How to perform read operation in Binary file

```
import pickle
file=open("xyz.dat","wb")
print(pickle.load(file))
file.close()
```

seek()

- It is used to change current location of file handle.
- File pointer is like a cursor, which defines from where the data to be read or written in the file.

Syntax:-f.seek(offset,from_where),where f is file pointer

offset:-Number of positions to move forward

from_where:-It is used for selecting the reference point. It can accept three values.

0:-sets the reference point at the beginning of the file

1:-sets the reference point at the current file position

2:-sets the reference point at the end of the file

- The from_where argument is set to 0 by default.
- Reference point at current position / end of file cannot be set in text mode except when offset is equal to 0.

Example:-

```
f=open("marwadiuni.txt","r")
f.seek(7)
print(f.tell())
print(f.read())
f.close()
```

- seek() function with negative offset only works when file is opened in binary mode.

Example:

```
f=open("marwadiuni.txt","rb")
f.seek(-10,2)
print(f.read())#It return b'bumrah'
f.close()
```

Note:-write f.read().decode('utf-8') to remove 'b' from output.

tell()

- It can be used to get the position of file handle.
- tell() method returns current position of file object.
- This method takes no parameters and returns an integer value.
- Initially file pointer points to the beginning of the file(if not opened in append mode). So, the initial value of tell() is zero.

Syntax:-file_object.tell()

Example:-Position of file pointer before reading or writing to file.

```
f=open("marwadiuni.txt","r")
print(f.tell())
f.close()
```

Example:-Position of File Handle after reading data from file.

```
f=open("marwadiuni.txt","r")
```

f.read(8)
print(f.tell())
f.close()

Example:-Position of File Handle before writing and after writing(Binary File)

```
f=open("marwadiuni.txt","wb")
print(f.tell())
f.write(b'1010101')
print(f.tell())
f.close()
```

Other File Operation

flush():-

- It is used to flush the internal buffer. The word flush means clear.
- After the use of close() function, python automatically clear the buffer. But sometime programmer need to flush(clear) the buffer before closing it.
- Buffer can be said as path to the memory from Input/Output devices.Therefore,while reading or writing the data to memory, the buffer is flushed(clear).
- Buffer can also be called as a portion of main memory(RAM) used to temporarily hold the data that is being transferred into main storage.

Syntax:file_object.flush()

Example:

```
f=open("marwadiuni.txt","w")
f.write("Python Programming\nAdvanced web technology")
f.flush()
print(f.read())
```

Example:

```
f=open("marwadiuni.txt","w")
f.write("Python Programming\nAdvanced web technology")
print(f.read())
f.flush()
print(f.read())
#read content of the file after flush, reads nothing as the internal buffer is cleared
f.close()
```

readable():-

- It is used to check whether a file is readable or not.
- It returns True if the file is readable otherwise returns False.

Syntax:-file_object.readable()

Example:

```
fileName=input("Enter name of file\n")
f=open(fileName,"r")
if f.readable():
        print("The given file can be read")
else:
        print("The given file is not readable")
```

writelines():-

- It is used to write multiple lines to a file at once.
- The lines are in the form of elements of a list.

Syntax:-file_object.writelines(list)

Example:-

```
f=open("cricket.txt","w")
f.writelines(["rohit sharma","KL rahul","virat kohli"])
f.close()
```

- Three lines(Cricketer Name) are written in single line. This is because, I've not inserted a newline before writing the second sentence. Therefore we need to place \n when we want to insert a newline. Therefore, using writelines(), multiple strings (paragraphs or whatever you say) can be written in the file.

truncate():-

- It is used to reduce the size of a file.
- It is used to resize the file to given number of bytes.

Example:

```
f=open("sairam.txt","a")
f.truncate(50)#Write 5,1 and check what will happen
f.close()
```

- After executing the above program, the size of file gets truncated. Now the size is 50 bytes.

Syntax:-file_object.truncate(size)

- The size parameter is optional. Therefore, if the size parameter is not specified, then the current position will be used. The default value of size is None.

Example:

```
fileName=input("Enter the name of file\n")
fileObject=open(fileName,"a")
sizeOfFile=int(input("Enter the size of file\n"))
fileObject.truncate(sizeOfFile)
fileObject.close()
print("The file {} is created of size {} bytes".format(fileName,sizeOfFile))
```

Marwadi University

Thank You