Chapter 4: File Handling in Python

**1. Explain file handling along with file opening modes**

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but alike other concepts of Python, this concept here is also easy and short. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let’s start with Reading and Writing files.

**Opening a file:**

The open() function is used to open a file which takes two parameters:

f=open(‘FileName’,’ModeType’)

|  |  |
| --- | --- |
| r | Read mode (Default). Open file for reading and gives error if file does not exist |
| w | Write. Open file for writing and creates a file if it does not exist |
| a | Append. Open file for appending and creates the file if it does not exist |
| x | Create. Creates a specific file and gives error if the file already exist |
| t | Text (Default). Text mode |
| b | Binary. Binary mode |

**2. Explain read operation on file**

There is more than one way to read a file in Python. If you need to extract a string that contains all characters in the file then we can use file.read(). The full code would work like this:

# Python code to illustrate read() mode

file = open("file.text", "r")

print (file.read())

Another way to read a file is to call a certain number of characters like in the following code the interpreter will read the first five characters of stored data and return it as a string:

# Python code to illustrate read() mode character wise

file = open("file.txt", "r")

print (file.read(5))

**3. Explain write operation on file**

To manipulate the file, write the following in your Python environment:

# Python code to create a file

file = open('demo.txt','w')

file.write("This is the write command")

file.write("It allows us to write in a particular file")

file.close()

The close() command terminates all the resources in use and frees the system of this particular program.

**4. Explain append operation on file**

To write to an existing file, you must add a parameter to the open() function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

Example

Open the file "demofile2.txt" and append content to the file:

f = open("demofile2.txt", "a")

f.write("Now the file has more content!")

f.close()

#open and read the file after the appending:

f = open("demofile2.txt", "r")

print(f.read())

Example

Open the file "demofile3.txt" and overwrite the content:

f = open("demofile3.txt", "w")

f.write("Woops! I have deleted the content!")

f.close()

#open and read the file after the appending:

f = open("demofile3.txt", "r")

print(f.read())

Note: the "w" method will overwrite the entire file.

**5. Explain create operation on file**

To create a new file in Python, use the open() method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

Example

Create a file called "myfile.txt":

f = open("myfile.txt", "x")

Result: a new empty file is created!