Module-4

PPT-1

File Handling

- File handling is an important part of any web application.
- Python has several functions for creating, reading, updating, and deleting files.
- The key function for working with files in Python is the open() function.
- The open() function takes two parameters; filename, and mode.
- There are four different methods (modes) for opening a file:
    1. "r" - Read - Default value. Opens a file for reading, error if the file does not exist
    2. "a" - Append - Opens a file for appending, creates the file if it does not exist
    3. "w" - Write - Opens a file for writing, creates the file if it does not exist
    4. "x" - Create - Creates the specified file, returns an error if the file exists

- In addition you can specify if the file should be handled as binary or text mode
  - "t" - Text - Default value. Text mode
  - "b" - Binary - Binary mode (e.g. images)
- To open a file for reading it is enough to specify the name of the file:

  ```
  f = open("demofile.txt")
  ```

  The code above is the same as:

  ```
  f = open("demofile.txt", "rt")
  ```

  Because "r" for read, and "t" for text are the default values, you do not need to specify them.

  Note: Make sure the file exists, or else you will get an error.

Python File Open

- To open the file, use the built-in open() function.
- The open() function returns a file object, which has a read() method for reading the content of the file:

```
f = open("demofile.txt", "r")
print(f.read())
```

- If the file is located in a different location, you will have to specify the file path, like this:
- Open a file on a different location:

```
f = open("D:\\myfiles\welcome.txt", "r")
print(f.read())
```

Sr.No.	Modes & Description
1	**r** Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
2	**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.
3	**r+** Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
4	**rb+** Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.
5	**w** Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
6	**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.
7	**w+** Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
8	**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.
9	**a** Opens a file for appending. 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.
10	**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.
11	**a+** Opens a file for both appending and reading. 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.
12	**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.

- **Read Only Parts of the File**
- By default the read() method returns the whole text, but you can also specify how many characters you want to return:
- Return the 5 first characters of the file:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

- **Read Lines**

You can return one line by using the readline() method:

- Read one line of the file:

```
f = open("demofile.txt", "r")
print(f.readline())
```

- **Read two lines of the file:**

```
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

- By looping through the lines of the file, you can read the whole file, line by line:
- Loop through the file line by line:

```
f = open("demofile.txt", "r")
for x in f:
  print(x)
```

- **Close the file when you are finish with it:**

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

Note: You should always close your files, in some cases, due to buffering, changes made to a file may not show until you close the file.

Writing Data to a 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
- 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())
```

- 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())
```

Create a New 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

  - Create a file called "myfile.txt":

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

  Result: a new empty file is created!

  - Create a new file if it does not exist:

  ```
  f = open("myfile.txt", "w")
  ```

Delete a File

- To delete a file, you must import the OS module, and run its os.remove() function:
- Remove the file "demofile.txt":

```
import os
os.remove("demofile.txt")
```

Check if File exist:

- To avoid getting an error, you might want to check if the file exists before you try to delete it:

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

Delete Folder

- To delete an entire folder, use the os.rmdir() method:

- Remove the folder "myfolder":
- import os
- os.rmdir("myfolder")

Note: You can only remove empty folders.

Additional File Methods

Method	Description
close()	Closes an opened file. It has no effect if the file is already closed.
detach()	Separates the underlying binary buffer from the TextIOBase and returns it.
fileno()	Returns an integer number (file descriptor) of the file.
flush()	Flushes the write buffer of the file stream.
isatty()	Returns True if the file stream is interactive.
read(n)	Reads at most n characters from 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)	Reads and returns one line from the file. Reads in at most n bytes if specified.
readlines(n=-1)	Reads and returns a list of lines from the file. Reads in at most n bytes/characters if specified.
seek(offset,from=SEEK_SET)	Changes the file position to offset bytes, in reference to from (start, current, end).
seekable()	Returns True if the file stream supports random access.
tell()	Returns the current file location.
truncate(size=None)	Resizes the file stream to size bytes. If size is not specified, resizes to current location.
writable()	Returns True if the file stream can be written to.
write(s)	Writes the string s to the file and returns the number of characters written.
writelines(lines)	Writes a list of lines to the file.