Python File Handling

What is a file?
How a file operation takes place in python

- Open a file

The open() function takes two parameters; *filename*, and *mode*.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"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)

Open a file

```
f = open("demofile.txt")
f = open("demofile.txt", "rt")
f = open("C:/Python33/README.txt")
f = open("test.txt", mode='r', encoding='utf-8’)
```

- Modes:
- '+' Open a file for updating (reading and writing)
- r, rb, r+, rb+, w, wb, w+, wb+, a, ab, a+, ab+

Read a file

- demofile.txt

  Hello! Welcome to demofile.txt
  This file is for testing purposes.
  Good Luck!

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

Output

Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!

Read only parts of the file:

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

Hello

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

Hello! Welcome to demofile.txt

This file is for testing purposes.

Loop through the file line by line:

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

Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!

Python File readline() Method

- Read the first line of the file "demofile.txt":
- f = open("demofile.txt", "r")
- print(f.readline())
- The readline() method returns one line from the file.
- You can also specify how many bytes from the line to return, by using the size parameter.
- Syntax
- file.readline(size)
- Parameter Values
- Parameter Description
- size Optional. The number of bytes from the line to return. Default -1, which means the whole line.
- Return only the five first bytes from the first line:
- f = open("demofile.txt", "r")
- print(f.readline(5))
- O/p: Hello

Python File readlines() Method

- Return all lines in the file, as a list where each line is an item in the list object:

- f = open("demofile.txt", "r")
- print(f.readlines())

- Do not return the next line if the total number of returned bytes ar more than 33:

- f = open("demofile.txt", "r")
- print(f.readline(33))

- The readlines() method returns a list containing each line in the file as a list item.

- Use the hint parameter to limit the number of lines returned. If the total number of bytes returned exceeds the specified number, no more lines are returned.

- Syntax
- file.readlines(hint)

- hint Optional. If the number of bytes returned exceed the hint number, no more lines will be returned. Default value is -1, which means all lines will be returned.

Write to a file

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()
```

```
f = open("demofile2.txt", "r")
print(f.read())
```

```
Hello! Welcome to demofile2.txt
This file is for testing purposes.
Good Luck!Now the file has more content!
```

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the
content!")
f.close()
```

```
f = open("demofile3.txt", "r")
print(f.read())
```

```
Woops! I have deleted the content!
```

Python File writelines() Method

- Open the file with "a" for appending, then add a list of texts to append to the file:

- f = open("demofile3.txt", "a")
- f.writelines(["See you soon!", "Over and out."])
- f.close()

- #open and read the file after the appending:
- f = open("demofile3.txt", "r")
- print(f.read())

- The writelines() method writes the items of a list to the file.

- Where the texts will be inserted depends on the file mode and stream position.

- "a": The texts will be inserted at the current file stream position, default at the end of the file.

- "w": The file will be emptied before the texts will be inserted at the current file stream position, default 0.

Python File writelines() Method

- Syntax
- file.writelines(list)

- Parameter Values
- Parameter Description
- list The list of texts or byte objects that will be inserted.

- Inserting line breaks for each list item:

- f = open("demofile3.txt", "a")
- f.writelines(["\nSee you soon!", "\nOver and out."])
- f.close()

- #open and read the file after the appending:
- f = open("demofile3.txt", "r")
- print(f.read())

Create a New File

Use the open() method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist
f = open("myfile.txt", "x")

"a" - Append - will create a file if the specified file does not exist
f = open("myfile.txt", “a")

"w" - Write - will create a file if the specified file does not exist
f = open("myfile.txt", “w")

Delete a File/Folder

- File

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

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

- Folder

- import os
  os.rmdir("myfolder")

Python File Methods

close()	Close an open file. It has no effect if the file is already closed.
fileno()	Return an integer number (file descriptor) of the file.
read(n)	Read atmost n characters form the file. Reads till end of file if it is negative or None.
seek(offset,from=SEEK_SET)	Change the file position to offset bytes, in reference to from (start, current, end)
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.

File Positions

- The tell() method tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.

- The seek(offset[, from]) method changes the current file position. The offset argument indicates the number of bytes to be moved. The from argument specifies the reference position from where the bytes are to be moved.

- If from is set to 0, it means use the beginning of the file as the reference position and 1 means use the current position as the reference position and if it is set to 2 then the end of the file would be taken as the reference position.

File Positions

- >>> f.tell() # get the current file position
  56
- >>> f.seek(0) # bring file cursor to initial position
  0
- >>> f.seek(5) # Go to the 6th byte in the file
- >>> f.seek(-3, 2) # Go to the 3rd byte before the end

Writing in file

Writing the contents of *string* to the file

```
>>> f.write('This is a test\n')
```

Writing *other objects* to the file

```
>>> value = ('the answer', 42)
>>> s = str(value)  # convert the tuple to string
>>> f.write(s)
```

What will ‘demo.txt’ contain after execution of this code?

```
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()

file = open(‘demo.txt','a')
file.write("This will add this line")
file.close()
```

demo.txt

This is the write command

It allows us to write in a particular file

This will add this line