Python Programming

By Neelesh Biswas, Moderator for Python Programming in The Uplift Project by GirlScript Foundation (Team: py4-team4)

File Handling : File Position & Directory Methods

File Position

- With every file, the file management system associates a pointer often known as file pointer that facilitates the movement across the file for reading/writing data.
- Once the read/write operation is completed, the pointer is automatically updated.
- **`tell()`** tells the current position within the file at which the next read or write operation will occur.
- **`seek(offset[,from])`** method is used to set the position of the file pointer or move the file pointer to a new location.
- **`offset`** argument indicates the number of bytes to be moved and the **`from`** argument specifies the reference position from where the bytes to be moved.

`from` and its Position & `os` module

From	Reference Position
0	From the Beginning of the file
1	From the current position of the file
2	From the end of the file

The `os` module: This module has various methods that can be used to perform file-performing operations like renaming **`[rename()]`** & deleting **`[remove()]`** files. To use the methods, we must import the module in the program first.

Syntax:

```
import os
```

Renaming & Deleting a File

- **To rename a file:**

```
os.rename(“[source file name]”,”[destination file name]”)
```

- **To remove/delete a file:**

```
os.remove(“[source file name]”)
```

Programs related to file position

1) Program that tells & sets the position of the file pointer.
2) Program that creates a file & writes a data.
3) Program that copies first 10 bytes of a binary file into another.
4) Renaming a file using rename().
5) Deleting a file using remove()

Directory & Directory Methods

A directory is a collection of files where each file may be of same or different formats. Python has various methods in the **`os`** module that help to work with directories.

Methods	Description
`mkdir()`	Used to create directories in the current directory
`getcwd()`	Used to display the current directory
`chdir()`	Used to change the current directory
`rmdir()`	Used to remove a directory
`makedirs()`	Used to create multiple directories

Directory Methods (contd.)

Method	Description
`os.path.join()`	Returns a string with a file path.
`os.path.abspath()`	Converts a string or a relative path to a absolute path
`os.path.isabs(path)`	Returns **True** if the path is absolute and **False** otherwise
`os.path.relpath(path,start)`	Accepts a string & returns a relative that begins from the start. **If start is not given, the current directory is taken as start.**
`os,path.dirname(path)`	Returns a string that includes everything specified in the path that comes before the last slash.
`os.path.basename(path)`	Returns a string that includes everything specified in the path that comes after the last slash.

Directory Methods (contd.)

Method	Description
`os.path.split(path)`	Accepts a file path & returns its directory name as well as basename.
`os.path.getsize(path)`	Returns the size of the file specified in the path argument.
`os.listdir(path)`	Returns a list of filenames in the specified path.
`os.path.exists(path)`	Returns **True** if the file or folder specified in the path exists and **False** otherwise
`os,path.isfile(path)`	Returns **True** if the existing file is specified by the path and **False** otherwise.
`os.path.isdir(path)`	Returns **True** if the existing directory is specified by the path and **False** otherwise.

Program related to directory methods

1) Create a new directory
2) Use of **`rmdir()`**
3) Use of **`os.path.join()`**
4) Print the absolute path of a file.
5) Use of **`os.path.split()`** & **`os.listdir(path).`**
6) Count the number of tabs, spaces & newline characters in a file.
7) Check if a flash drive is connected to a computer.
8) Program that accepts filename as an input from the user & counts the no of times a character appears in the file