# Section 11

Files

Mohammed Asir Shahid

2021-08-04

## Contents

# 1 Filenames and Absolute/Relative File Paths

We'll learn about files, folders, and how Python can work with them.

If we want our data to persist after our program is finished, we need to save it to a file.

## 1.1 Fileames and File Paths

Each of our files has 2 key properties. The file name and the file path.

## 1.2 os module

The os module contains numerous file path related functions that we can use.

### 1.2.1 os.getcwd()

Every program has a setting called the current working directory, this tells us the directory that we are currently in. We can use the getcwd() function in order to find our current working directory.

```
import os
```

```
print(os.getcwd())
```

```
/home/mohammeds/Documents/Automate the Boring Stuff/Section 11: Files
```

### 1.2.2 os.chdir()

We can also change the current working directory using this chdir() function.

```
import os

print(os.getcwd())

os.chdir("/")

print(os.getcwd())
```

```
/home/mohammeds/Documents/Automate the Boring Stuff/Section 11: Files
/
```

## 1.3 Absolute and Relative Paths

There are two kinds of file paths, relative and absolute. The absolute file path begins with the root directory and gives you the complete location of the file. A relative file path is relative to the current working directory.

## 1.4 . and .. Folders

These are not real directories, but they can be used with relative paths. The single dot stands for this directory while two dots stands for the paret directory, the directory that your current working directory is in.

## 1.5 os.path module

### 1.5.1 os.path.join()

This is a join function inside of a path module inside of an os module. It takes several string arguments and returns a string value of a path for the os that you are using.

```
import os
print(os.path.join("folder1","folder2","folder3","file.png"))
```

```
folder1/folder2/folder3/file.png
```

### 1.5.2 os.path.abspath()

The abspath() function will return the absolute path of the path that you pass it.

```
import os

print(os.path.abspath("Section 11.org"))
```

```
/home/mohammeds/Documents/Automate the Boring Stuff/Section 11: Files/Section 11.org
```

### 1.5.3 os.path.isabs()

The isabs() function is a way to determine if a given path is absolute.

```
import os

print(os.path.abspath("Section 11.org"))

print(os.path.isabs("Section 11.org"))

print(os.path.abspath("/home/mohammeds/Documents/Automate the Boring Stuff/Section 11:

print(os.path.isabs("/home/mohammeds/Documents/Automate the Boring Stuff/Section 11: Fi
```

```
/home/mohammeds/Documents/Automate the Boring Stuff/Section 11: Files/Section 11.org
False
/home/mohammeds/Documents/Automate the Boring Stuff/Section 11: Files/Section 11.org
True
```

### 1.5.4 os.path.relpath()

This function will give you the relative path between two paths. The first argument is the file/directory we want to get to and the second is the directory we are already in.

```
import os

print(os.path.relpath("Section 11.org", "/usr/bin"))

print(os.path.relpath("Section 11.org", "/home/mohammeds/"))
```

```
../../home/mohammeds/Documents/Automate the Boring Stuff/Section 11: Files/Section 11.o
Documents/Automate the Boring Stuff/Section 11: Files/Section 11.org
```

### 1.5.5 os.path.dirname()

This function pulls out just the directory part of the path.

```
import os

print(os.path.dirname("/home/mohammeds/Documents/Automate the Boring Stuff/Section 11:
```

```
/home/mohammeds/Documents/Automate the Boring Stuff/Section 11: Files
```

### 1.5.6 os.path.basename()

This function pulls out just the last part of the path. Either the filename or just the directory itself if there is no file in the path.

```
import os

print(os.path.basename("/home/mohammeds/Documents/Automate the Boring Stuff/Section 11
```

```
Section 11.org
```

### 1.5.7 os.path.exists()

This function can check and see if the path you are passing actually exists.

```
import os

print(os.path.exists("/home/mohammeds/Documents/Automate the Boring Stuff/Section 11: F

print(os.path.exists("/home/mohammeds/Fake File Path"))
```

```
True
False
```

### 1.5.8 os.path.isfile()

We can use this function to see if what we pass to it is a file or not.

```
import os

print(os.path.isfile("/home/mohammeds/Documents/Automate the Boring Stuff/Section 11: F

print(os.path.isfile("/home/mohammeds/Fake File Path"))
```

```
True
False
```

### 1.5.9 os.path.isdir()

We can use this function to see if what we pass to it is a directory or not.

```
import os

print(os.path.isdir("/home/mohammeds/Documents/Automate the Boring Stuff/Section 11: Fi

print(os.path.isdir("/home/mohammeds/"))
```

```
False
True
```

### 1.5.10 os.path.getsize()

This function gives us the size of the file in bytes.

```
import os

print(os.path.getsize("/home/mohammeds/Documents/Automate the Boring Stuff/Section 11:
```

```
13214
```

## 1.6 os.listdir()

This function isn't in the path module. It can be passed a file path of a folder and then it'll return a list of strings of the filenames and folder names that it contains.

```
import os

print(os.listdir("/home/mohammeds/Documents"))
```

```
['[Chapman and Hall_CRC the R Ser] Albert, Jim_ Baumer, Benjamin S._ Marchi, Max - Anal
```

## 1.7 Example Code: Finding the total size of all files in a folder.

```
import os

totalSize=0

for file in os.listdir("/home/mohammeds/Documents"):
    if not os.path.isfile(os.path.join("/home/mohammeds/Documents", file)):
        continue
    totalSize+=os.path.getsize(os.path.join("/home/mohammeds/Documents", file))

print(totalSize)
```

```
382371482
```

### 1.8 os.makedirs()

This function can create directories for you.

```
import os

print(os.getcwd())

print(os.listdir())

os.makedirs("TestDir")

print(os.listdir())
```

## 2 Reading and Writing Plaintext Files

We can start writing strings to files that we create in order to save information. We can then read this data with Python.

### 2.1 Plaintext and Binary Files

Plaintext files only contain basic text characters and don't include information about color or font. Often times they have .txt extensions. Python scripts are also plain text files, except they use the .py file extension.

They can be opened with text editors such as Emacs.

Binary files are all other types of files. PDF, JPG, PNG, etc. When you open these with a text editor, it will be impossible to understand.

### 2.2 Reading or Writing Files in Python

There are three main steps for reading and writing files in Python.

#### 2.2.1 The open() Function

This function opens files.

```
open("hello.txt")
```

### 2.2.2 Read Mode

The open() function opens the file in read mode. This only lets you read the data, it does not let you modify it.

### 2.2.3 The read() method

This method lets you read the opened file.

### 2.2.4 The close() method

This method closes the opened file. If you want to continue accessing it, you should save it to a variable.

```
helloFile=open("hello.txt")


content=helloFile.read()

print(content)

helloFile.close()
```

```
Hello!
How are you?How are you?How are you?
```

### 2.2.5 The readlines() Method

This method returns all of the lines as strings within a list.

```
helloFile=open("hello.txt")
content=helloFile.readlines()

print(content)
```

```
['Hello!\n', 'How are you?How are you?How are you?']
```

### 2.2.6 Write mode

In order to write to a file, it needs to be opened in write or append mode. This can overwrite existing files. For write mode, you can pass in a "w" argument to the open() function.

```
helloFile=open("hello2.txt","w")
helloFile.write("Hello!!!!")
```

### 2.2.7 Append mode

In order to write to a file, it needs to be opened in write or append mode. This can overwrite existing files. For append mode, you can pass in a "a" argument to the open() function.

```
helloFile=open("hello.txt", "a")
helloFile.write("How are you?")
```

### 2.2.8 Example

```
baconFile=open("bacon.txt", "w")

baconFile.write("Bacon is not a vegetable")
baconFile.close()

baconFile=open("bacon.txt")

print(baconFile.read())
```

```
Bacon is not a vegetable
```

## 2.3 The shelve Module

Writing and reading text files is a good way to store single long strings, but if we want to save more complex data structures like lists and dictionaries, then we can save Python programs to binary shelve files.

### 2.3.1 The shelve.open() Method

This method can open shelve files.

```
import shelve

shelfFile=shelve.open("mydata")
shelfFile["cats"]=["Zophie","Pooka","Simon","Fat-tail","Cleo"]
shelfFile.close()

shelfFile=shelve.open("mydata")
print(shelfFile["cats"])
```

```
['Zophie', 'Pooka', 'Simon', 'Fat-tail', 'Cleo']
```

The benefit of using the shelve module is that you can store lists, dictionaries, and non text data and then reopen them in the future.

### 2.3.2 The keys() and values() Shelf Methods

Shelf file objects are very similar to dictionaries in the sense that they have keys and values.

```
import shelve

shelfFile=shelve.open("mydata")

print(shelfFile.keys())
print(shelfFile.values())


print(list(shelfFile.keys()))
print(list(shelfFile.values()))

shelfFile.close()
```

```
KeysView(<shelve.DbfilenameShelf object at 0x7fb5cf2e1130>)
ValuesView(<shelve.DbfilenameShelf object at 0x7fb5cf2e1130>)
['cats']
[['Zophie', 'Pooka', 'Simon', 'Fat-tail', 'Cleo']]
```

# 3 Copying and Moving Files and Folders

We can also organize files on our drive using Python.

## 3.1 Shell Utilities Module

This module has functions that let us copy, rename, and delete files in Python programs.

### 3.1.1 shutil.copy() Function

This function lets us copy files in Python. We can either copy and leave the filename or copy and give a new file name.

```
import shutil,os

shutil.copy("hello2.txt", "TestDir")

shutil.copy("hello2.txt", "TestDir/newhello2.txt")

print(os.listdir("TestDir"))
```

```
['hello2.txt', 'newhello2.txt']
```

### 3.1.2 shutil.copytree() Function

The copy function works for single files, but what if we want to copy an entire folder? Then we can use the copytree function.

```
import shutil,os

shutil.copytree("TestDir","TestDirBackup")
```

```
import shutil,os

print(os.listdir("TestDirBackup"))
```

```
['hello2.txt']
```

### 3.1.3 shutil.move() function

This can be used for moving and renaming files. If you want to rename a file, you can move it to the same directory using the move() function and then change the filename.

```
import shutil,os

shutil.move("TestDirBackup/newhello2.txt", ".")

print(os.listdir("TestDirBackup"))
```