Python Directory and Files Management

programiz.com/python-programming/directory

Join our newsletter for the latest updates.

In this tutorial, you'll learn about file and directory management in Python, i.e. creating a directory, renaming it, listing all directories, and working with them.

Python Directory

If there are a large number of files to handle in our Python program, we can arrange our code within different directories to make things more manageable.

A directory or folder is a collection of files and subdirectories. Python has the `os` module that provides us with many useful methods to work with directories (and files as well).

Get Current Directory

We can get the present working directory using the `getcwd()` method of the `os` module.

This method returns the current working directory in the form of a string. We can also use the `getcwdb()` method to get it as bytes object.

```
>>> import os

>>> os.getcwd()
'C:\\Program Files\\PyScripter'

>>> os.getcwdb()
b'C:\\Program Files\\PyScripter'
```

The extra backslash implies an escape sequence. The `print()` function will render this properly.

```
>>> print(os.getcwd())
C:\Program Files\PyScripter
```

Changing Directory

We can change the current working directory by using the `chdir()` method.

The new path that we want to change into must be supplied as a string to this method. We can use both the forward-slash `/` or the backward-slash `\` to separate the path elements.

It is safer to use an escape sequence when using the backward slash.

```
>>> os.chdir('C:\\Python33')

>>> print(os.getcwd())
C:\Python33
```

List Directories and Files

All files and sub-directories inside a directory can be retrieved using the `listdir()` method.

This method takes in a path and returns a list of subdirectories and files in that path. If no path is specified, it returns the list of subdirectories and files from the current working directory.

```
>>> print(os.getcwd())
C:\Python33

>>> os.listdir()
['DLLs',
'Doc',
'include',
'Lib',
'libs',
'LICENSE.txt',
'NEWS.txt',
'python.exe',
'pythonw.exe',
'README.txt',
'Scripts',
'tcl',
'Tools']

>>> os.listdir('G:\\')
['$RECYCLE.BIN',
'Movies',
'Music',
'Photos',
'Series',
'System Volume Information']
```

Making a New Directory

We can make a new directory using the `mkdir()` method.

This method takes in the path of the new directory. If the full path is not specified, the new directory is created in the current working directory.

```
>>> os.mkdir('test')

>>> os.listdir()
['test']
```

Renaming a Directory or a File

The `rename()` method can rename a directory or a file.

For renaming any directory or file, the `rename()` method takes in two basic arguments: the old name as the first argument and the new name as the second argument.

```
>>> os.listdir()
['test']

>>> os.rename('test','new_one')

>>> os.listdir()
['new_one']
```

Removing Directory or File

A file can be removed (deleted) using the `remove()` method.

Similarly, the `rmdir()` method removes an empty directory.

```
>>> os.listdir()
['new_one', 'old.txt']

>>> os.remove('old.txt')
>>> os.listdir()
['new_one']

>>> os.rmdir('new_one')
>>> os.listdir()
[]
```

Note: The `rmdir()` method can only remove empty directories.

In order to remove a non-empty directory, we can use the `rmtree()` method inside the `shutil` module.

```
>>> os.listdir()
['test']

>>> os.rmdir('test')
Traceback (most recent call last):
...
OSError: [WinError 145] The directory is not empty: 'test'

>>> import shutil

>>> shutil.rmtree('test')
>>> os.listdir()
[]
```