Python Directory and Files Management

A directory is a collection of files and subdirectories. A directory inside a directory is known as a subdirectory.

Python has the `os` module that provides us with many useful methods to work with directories (and files as well).

Get Current Directory in Python

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. For example,

```
import os

print(os.getcwd())

# Output: C:\Program Files\PyScripter
```

Here, `getcwd()` returns the current directory in the form of a string.

Changing Directory in Python

In Python, 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. And we can use both the forward-slash `/` or the backward-slash `\` to separate the path elements.

Let's see an example,

```
import os

# change directory
os.chdir('C:\\Python33')

print(os.getcwd())

Output: C:\Python33
```

Here, we have used the `chdir()` method to change the current working directory and passed a new path as a string to `chdir()`.

List Directories and Files in Python

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.

```
import os

print(os.getcwd())
```

```
C:\Python33

# list all sub-directories
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 in Python

In Python, 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, `rename()` takes in two basic arguments:

- the old name as the first argument
- the new name as the second argument.

Let's see an example,

```
import os

os.listdir()
['test']

# rename a directory
os.rename('test','new_one')

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

Here, `'test'` directory is renamed to `'new_one'` using the `rename()` method.

Removing Directory or File in Python

In Python, we can use the `remove()` method or the `rmdir()` method to remove a file or directory.

First let's use `remove()` to delete a file,

```
import os

# delete "myfile.txt" file
os.remove("myfile.txt")
```

Here, we have used the `remove()` method to remove the `"myfile.txt"` file.

Now let's use `rmdir()` to delete an empty directory,

```
import os

# delete the empty directory "mydir"
os.rmdir("mydir")
```

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

```
import shutil

# delete "mydir" directory and all of its contents
shutil.rmtree("mydir")
```

It's important to note that these functions permanently delete the files or directories, so we need to careful when using them.

Also Read:

- Python Program to Get the Full Path of the Current Working Directory

Table of Contents

- Introduction
- Get Current Directory in Python
- Changing Directory in Python
- List Directories and Files in Python
- Making a New Directory in Python
- Renaming a Directory or a File
- Removing Directory or File in Python