Getting started with Python-

Working with Directories in Python

Contents

Working with Directories

Python provides libraries containing methods that allow your programs to interact with files in your computer. Files and Directories form a Hierarchical structure in your system drive.

Files contain information. For example: csv files, word files, or python files.

Directories contain files and sub directories inside them. Directories can be helpful in managing large number of files in your program.

Working with Directories

- Python has the 'os' module which provides us with several useful functions to work with files and directories.
- To get started, `import os` in your namespace

```
import os
```

- We can check current working directory or change working directory.
- **`getcwd()`** returns your current working directory in the form of a string.
- **`chdir()`** lets you change the working directory. The new path must be given as a string to this function.

```
# Check the current working directory
import os
os.getcwd()
```

```
'C:\\Users\\HP\\.spyder'
```

```
# Change the working directory
os.chdir("C:/Users/HP/Documents/")
```

Both forward (/) or backward slash(\) can be used.

Working with Directories

- **`listdir()`** takes exactly 1 argument as a path and returns list of subdirectories and files in that path.

```
# List directories and files
os.listdir("c:/")
```

```
['$Recycle.Bin', 'Documents and Settings', 'hiberfil.sys',
'MSOCache', 'pagefile.sys', 'PerfLogs', 'Program Files', 'Program
Files (x86)', 'ProgramData', 'Python27', 'Recovery', 'SWSetup',
'SWTOOLS', 'System Volume Information', 'Users', 'Windows']
```

Working with Directories

- Create new directory

```
# Creating a new directory (using mkdir() function)
os.mkdir('mypython')
```

If the full path is not supplied, the new directory is created in the current working directory

- Rename a directory or file

```
# Renaming a directory or file (using rename() function)
os.rename('mypython','mynewpython')
```

First argument: old name
Second argument: new name

- Remove a directory or file

```
# Removing directory or file (using rmdir() & remove() function)
os.rmdir('mynewpython')
os.remove('oldfile.txt')
```

rmdir() removes an empty directory
remove() removes a file.

Quick Recap

Working with Directories	• **`os.getcwd()`** and **`os.chdir()`**: To check and set working directory respectively
Create new directory	• **`os.mkdir():`** To create a new directory
Rename directory	• **`os.rename():`** To rename a created a directory
Remove directory & file	• **`os.rmdir()`** and **`os.remove():`** To remove directory and file respectively