Working with Directories in Python

The **OS module** in python provides functions for interacting with the operating system. This module contains an interface to many **operating system-specific** functions to manipulate processes, files, file descriptors, directories and other "low level" features of the OS.

*Current Working Directory

The **getcwd()** returns the path to the current working directory. This is the directory which the OS use to transform a relative file name into an absolute file name.

Example:

```
import os
cur_dir = os.getcwd()
print(cur_dir)
```

*Create a new Directory/Folder

The **mkdir()** method creates a new directory. It returns an error if the parent directory does not exist.

example

```
import os

os.mkdir("Temp")
```

The above example create a new directory "Temp" in the current path.

*Creating Subdirectories

import os

os.makedirs("Temp/temp1/temp2/")

*Deleting an empty Directory/Folder

The **rmdir()** method will delete an empty directory or folder.

Example

import os

os.rmdir("Temp")

*Renaming a directory/folder

The **os.rename()** method can rename a folder from an old name to a new one.

example

import os

os.rename("Temp","Temp11")

*Check whether a file exists using Python

When writing **Python scripts** , we might just need to know if a specific file or directory or a path **exists or not** . Python offers several alternative ways of checking whether a file exists or not. To check this, we use functions built into the core language and the **Python standard library** . They are:

- os.path.isfile()
- os.path.exists()
- pathlibPath.exists() (Python 3.4+)
- open() and try...except
- os.path.isdir()

*os.path.isfile()

This is the **simplest way** to check if a file exists or not.

```
import os.path

filename = "my_file.txt"

if(os.path.isfile(/filepath/filename)):

  print("File Exists!!")

else:

  print("File does not exists!!")
```

If the file "my_file.txt" exist in the current path, it will return **true** else **false** .

File Handling

The key function for working with files in Python is the open() function.

The open() function takes two parameters; *filename*, and *mode*.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

Syntax

To open a file for reading it is enough to specify the name of the file:
f = open("demofile.txt")

The code above is the same as:
f = open("demofile.txt", "rt")

Because "r" for read, and "t" for text are the default values, you do not need to specify them.

Note: Make sure the file exists, or else you will get an error.

Open a File on the Server

Assume we have the following file, located in the same folder as Python:

demofile.txt

Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!

To open the file, use the built-in open() function.

The open() function returns a file object, which has a read() method for reading the content of the file:

Example

```
f = open("demofile.txt", "r")
print(f.read())
```

*Read Only Parts of the File

By default the read() method returns the whole text, but you can also specify how many characters you want to return:

Example

Return the 5 first characters of the file:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

Read Lines

You can return one line by using the readline() method:

Example

Read one line of the file:

```
f = open("demofile.txt", "r")
print(f.readline())
```

By calling `readline()` two times, you can read the two first lines:

Example

Read two lines of the file:

```
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

By looping through the lines of the file, you can read the whole file, line by line:

Example

Loop through the file line by line:

```
f = open("demofile.txt", "r")
for x in f:
  print(x)
```

Close Files

It is a good practice to always close the file when you are done with it.

Example

Close the file when you are finish with it:

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

Write to an Existing File

To write to an existing file, you must add a parameter to the `open()` function:

`"a"` - Append - will append to the end of the file

`"w"` - Write - will overwrite any existing content

Example

Open the file "demofile2.txt" and append content to the file:

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

Example

Open the file "demofile3.txt" and overwrite the content:

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()

#open and read the file after the appending:
f = open("demofile3.txt", "r")
print(f.read())
```

Note: the "w" method will overwrite the entire file.

Create a New File

To create a new file in Python, use the `open()` method, with one of the following parameters:

`"x"` - Create - will create a file, returns an error if the file exist

`"a"` - Append - will create a file if the specified file does not exist

`"w"` - Write - will create a file if the specified file does not exist

Example

Create a file called "myfile.txt":

```
f = open("myfile.txt", "x")
```

Result: a new empty file is created!

Example

Create a new file if it does not exist:

```
f = open("myfile.txt", "w")
```

Delete a File

To delete a file, you must import the OS module, and run its `os.remove()` function:

Example

Remove the file "demofile.txt":

```
import os
os.remove("demofile.txt")
```

Check if File exist:

To avoid getting an error, you might want to check if the file exists before you try to delete it:

Example

Check if file exists, *then* delete it:

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

Delete Folder

To delete an entire folder, use the `os.rmdir()` method:

Example

Remove the folder "myfolder":

```
import os
os.rmdir("myfolder")
```

Exercises:

1. Write a Python program to read an entire text file.
2. Write a Python program to read first n lines of a file.
3. Write a Python program to read last n lines of a file.
4. . Write a Python program to copy the contents of a file to another file.
5. Write a Python program to extract characters from various text files and puts them into a list.