




PYTHON



OS MODULE



Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.

The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality.



The Python OS module lets us work with the files and directories.

To work with the OS module, we need to **import** the OS module.

import os



OS FUNTIONS

OS.NAME()

This function provides the name of the operating system module that it imports. Currently, it registers 'posix', 'nt', 'os2', 'ce', 'java' and 'riscos'.

```
import os
```

```
print(os.name)
```

OS.MKDIR()

The **os.mkdir()** function is used to create new directory. Consider the following example.

```
import os
```

```
os.mkdir("d:\\newdir")
```

It will create the new directory to the path in the string argument of the function in the D drive named folder newdir.

OS.GETCWD()

It returns the current working directory(CWD) of the file.

```
import os
```

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


OS.CHDIR()

The **os** module provides the **chdir()** function to change the current working directory.

```
import os
```

```
os.chdir("d:\\")
```

OS.RMDIR()

The **rmdir()** function removes the specified directory with an absolute or related path. First, we have to change the current working directory and remove the folder.

```
import os
```

```
# It will throw a Permission error; that's why we have to change the current working directory.
```

```
os.rmdir("d:\\newdir")
```

```
os.chdir("..")
```

```
os.rmdir("newdir")
```

OS.ERROR()

The `os.error()` function defines the OS level errors. It raises `OSError` in case of invalid or inaccessible file names and path etc.

import os

try:

If file does not exist, then it throw an IOError

filename = 'Python.txt'

f = open(filename, 'rU')

text = f.read()

f.close()

The Control jumps directly to here if any lines throws IOError.

except IOError:

print(os.error) will <class 'OSError'>

print('Problem reading: ' + filename)

OS.POPEN()

This function opens a file or from the command specified, and it returns a file object which is connected to a pipe.

```
import os
```

```
fd = "python.txt"
```

```
# popen() is similar to open()
```

```
file = open(fd, 'w')
```

```
file.write("This is awesome")
```

```
file.close()
```

```
file = open(fd, 'r')
```

```
text = file.read()
```

```
print(text)
```

```
# popen() provides gateway and accesses the file directly
```

```
file = os.popen(fd, 'w')
```

```
file.write("This is awesome")
```

```
# File not closed, shown in next function.
```

OS.CLOSE()

This function closes the associated file with descriptor **fr**.

```
import os  
  
fr = "Python1.txt"  
  
file = open(fr, 'r')  
  
text = file.read()  
  
print(text)  
  
os.close(file)
```

OS.RENAME()

A file or directory can be renamed by using the function **os.rename()**. A user can rename the file if it has privilege to change the file.

```
import os
```

```
fd = "python.txt"
```

```
os.rename(fd,'Python1.txt')
```

```
os.rename(fd,'Python1.txt')
```


OS.ACCESS()

This function uses real **uid/gid** to test if the invoking user has access to the path.

```
import os
```

```
import sys
```

```
path1 = os.access("Python.txt", os.F_OK)
```

```
print("Exist path:", path1)
```

```
path2 = os.access("Python.txt", os.R_OK)
```

```
print("It access to read the file:", path2)
```

```
path3 = os.access("Python.txt", os.W_OK)
```

```
print("It access to write the file:", path3)
```

```
path4 = os.access("Python.txt", os.X_OK)
```

```
print("Check if path can be executed:", path4)
```



CONTINUE IN NEXT UNIT