OS module

OS module is a predefined module which comes with python software. OS stands for operating system. This module is used to communicate with operation system (OR) python program executes the functions of OS using OS module.

Os module is operating system dependent.

os.name
The name of the operating system dependent module imported. The following names have currently been registered: 'posix', 'nt', 'java'.

```
>>> import os
>>> os.name
'nt'
```

nt → windows
posix → unix or linux
java → solaries

os.getcwd()
Return a string representing the current working directory.

```
>>> import os
>>> os.getcwd()
'C:\\Users\\nit\\AppData\\Local\\Programs\\Python\\Python312
'
>>> f=open("file1.txt","w")
```

If the file is created or opened without defining path, it create or open the file exists in current working directory

Example:
```
import os
```

```
print("Hello")
f=open("file1.txt","w")
print(os.getcwd())
```

Output

```
Hello
E:\python5pmjun
```

os.chdir(*path*)

Change the current working directory to *path*.

Example:

```
import os
print("Hello")
os.chdir("e:\\")
print("Chnaged Current Working Dicrectory ")
print(os.getcwd())
f=open("file1.txt","w")
print(os.getcwd())
```

Output

```
Hello
Chnaged Current Working Dicrectory
e:\
e:\
```

os.mkdir(path)

Create a directory named path.

Example:

```
# Write a program to create folder or directory

import os

fname=input("FolderName :")
os.mkdir(fname)
print("Folder Created...")
```

Output
```
FolderName :e:\\f1
Folder Created...
```

os.listdir(*path*='.')
Return a list containing the names of the entries in the directory given by *path*. The list is in arbitrary order, and does not include the special entries '.' and '..' even if they are present in the directory.

```
import os

print(os.getcwd())
list1=os.listdir()
print(list1)
print("=========================================================")
list2=os.listdir("..")
print(list2)
print("=========================================================")
list3=os.listdir("e:\\MySQL")
print(list3)
```

os.rmdir(path, *, dir_fd=None)
Remove (delete) the directory path. If the directory does not exist or is not empty, a FileNotFoundError or an OSError is raised respectively. In order to remove whole directory trees, shutil.rmtree() can be used.

Example:
```
import os

os.rmdir("e:\\folder1")
print("Folder is Deleted...")
```

Output
```
Folder is Deleted...
```

Example:

```
import os

os.rmdir("e:\\data")
print("Folder is Deleted...")
```

Output

```
Traceback (most recent call last):
  File "E:/python5pmjun/ostest4.py", line 3, in <module>
    os.rmdir("e:\\data")
OSError: [WinError 145] The directory is not empty: 'e:\\data'
```

shutil.rmtree(*path)*

Delete an entire directory tree. It will delete non empty folder or directories.

```
import shutil

shutil.rmtree("e:\\proj1")
print("Folder is Deleted...")
```

Output

```
Folder is Deleted...
```

Examine files

os.path module provides the functionality for testing the files or file properties.

os.path.exists(path)

Return True if path refers to an existing path or an open file descriptor. Returns False

Example:

```
# Write a program to read content of file
```

```
import os.path
fname=input("Enter filename to read")

if os.path.exists(fname):
    fobj=open(fname,"r")
    data=fobj.read()
    print(data)
else:
    print("Invalid filename or filename not exists")
```

Output

```
Enter filename to reade:\\file2.txt
Jython3.1265651.5
dglkdg sdlfgkdfg
topeiter
tyoprer fghl;dfkg;
ert;lerk
s'ldfgkdsg'k

Enter filename to reade:\\xyz
Invalid filename or filename not exists
```

os.path.isfile(*path*)
Return True if *path* is an existing regular file.

os.path.isdir(*path*)
Return True if *path* is an existing directory

Example:

```
import os.path

b1=os.path.isfile("e:\\file2.txt")
print(b1)
b2=os.path.isdir("e:\\folder2")
print(b2)
b3=os.path.isfile("e:\\folder2")
print(b3)
```

```
b4=os.path.isdir("e:\\file2.txt")
print(b4)
```

Output

```
True
True
False
False
```

Example:

```
# Write a program to read content of file


import os.path

fname=input("Enter FileName to Read ")
if os.path.exists(fname):
    if os.path.isfile(fname):
        fobj=open(fname,"r")
        data=fobj.read()
        print(data)
    else:
        print("Given filename is folder")
else:
    print("File not found")
```

Output

```
Enter FileName to Read e:\\file2.txt
Jython3.1265651.5
dglkdg sdlfgkdfg
topeiter
tyoprer fghl;dfkg;
ert;lerk
s'ldfgkdsg'k
```

```
Enter FileName to Read e:\\folder2
Given filename is folder

Enter FileName to Read e:\\file6.txt
File not found
```

Example:

```
import os
import os.path

fname=input("FolderName ")
if os.path.exists(fname):
    if os.path.isdir(fname):
        os.chdir(fname)
        list1=os.listdir(fname)
        fc,dc=0,0
        for name in list1:
            if os.path.isfile(name):
                fc+=1
            else:
                dc+=1
        print(f'File count {fc}')
        print(f'Directory count {dc}')
    else:
        print("it is not director or folder")
else:
    print("given path is not exists")
```

Output

```
FolderName E:\\djangomay7pm
File count 85
Directory count 68

FolderName e:\\
File count 65
Directory count 35
```

os.remove(*path*, *, *dir_fd=None*)¶
Remove (delete) the file *path*. If *path* is a directory, an OSError is raised.

Example:

```
# Write a program to delete or remove file

import os
import os.path

fname=input("Enter filename to delete/remove ")
if os.path.exists(fname):
    if os.path.isfile(fname):
        os.remove(fname)
        print("File deleted.. ")
    else:
        print("given filename is folder")
else:
    print("file not exists")
```

Output

```
Enter filename to delete/remove e:\\file1.txt
File deleted..

Enter filename to delete/remove e:\\filet1.txt
file not exists

Enter filename to delete/remove e:\\folder2
given filename is folder
```

sys.path

path is environment variable
by default python search modules and packages import in program using path.
Path is nothing location.
In order to add path dynamically,

sys.path.append("location")

Example

```
import sys

sys.path.append("e:\\")
import m1

print(sys.path)
m1.fun1()
```

Output

```
['E:/python5pmjun',
'C:\\Users\\nit\\AppData\\Local\\Programs\\Python\\Python312
\\Lib\\idlelib',
'C:\\Users\\nit\\AppData\\Local\\Programs\\Python\\Python312
\\Lib\\site-packages',
'C:\\Users\\nit\\AppData\\Local\\Programs\\Python\\Python312
\\%PYTHONPATH%', 'e:\\folder1',
'C:\\Users\\nit\\AppData\\Local\\Programs\\Python\\Python312
',
'C:\\Users\\nit\\AppData\\Local\\Programs\\Python\\Python312
\\python312.zip',
'C:\\Users\\nit\\AppData\\Local\\Programs\\Python\\Python312
\\DLLs',
'C:\\Users\\nit\\AppData\\Local\\Programs\\Python\\Python312
\\Lib', 'e:\\']
inside fun1
```

shutil.copyfile(*src*, *dst*)

Copy the contents (no metadata) of the file named src to a file named dst and return dst in the most efficient way possible. src and dst are path-like objects or path names given as strings.
dst must be the complete target file name; look at copy() for a copy that accepts a target directory path. If src and dst specify the same file, SameFileError is raised.

Example:

```
import shutil


shutil.copy("e:\\file2.txt","e:\\file3.txt")
print("file copied...")
shutil.copy("e:\\file2.txt","e:\\Test")
print("file copied...")
shutil.copy("e:\\file2.txt","e:\\Test\\file3.txt")
print("file copied...")
```

Output

```
file copied...
file copied...
file copied...
```

os.system()

os.system("shutdown option")

options

/s → shutdown
/r → restart
/l → logout
/t → time

```
os.system("shutdown /s /t 0")
os.system(“shutdown /r /t 4”)
os.system(“shutdown /l t 0”)
```