Python OS module:

--->> 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).

--->> The Python os module enables interaction with the operating system.

--->> The os module provides the functions that are involved in file processing operations like renaming, deleting, etc.

getcwd() method

This method returns the current working directory.

The syntax to use the getcwd() method is given below.

Syntax: os.getcwd()

Example:

```
import os

print(os.getcwd())
```

Output:

D:\Python_Batch

listdir():

- This method returns a list of all the files and folders present inside the specified directory. If no directory is specified then the list of files and folders inside the CWD is returned.

  Example:

  ```
  import os

  print(os.listdir())
  ```

  Output:

  ['demo', 'demo.txt', 'demo2.txt', 'demo3.txt', 'demo4.txt', 'Django', 'files_program.py', 'sample.txt']

Renaming the file:

- It provides us the **rename()** method to rename the specified file to a new name.

- The syntax to use the rename() method is given below.

Syntax: rename(current-name, new-name)

- The first argument is the current file name and the second argument is the modified name. We can change the file name by passing these two arguments.

Example1:

```
import os

#rename file2.txt to file3.txt

os.rename("file2.txt","file3.txt")
```

Output: The above code renamed current file2.txt to file3.txt

Removing the file

- This **remove()** method deletes a file path. It cannot delete a directory. In case the specified path is that of a directory then the OSError is raised.
- **remove()** method is used to remove the specified file only but not directory.

Syntax: remove('file_name')

Example1:

```
import os

#deleting the file named file3.txt

os.remove("file3.txt")
```

Creating the new directory

- This **mkdir()** method creates a new directory according to the specified path. In case the specified directory already exists a **FileExistsError** is raised.

Syntax: os.mkdir("/path/directory name")

Example1:

```
import os

#creating a new directory with the name new

os.mkdir("new")
```

Changing the current working directory:

- The chdir() method is used to changing the current working directory to the specified path.
- The syntax to use the chdir() method is given below.

Syntax: **os.chdir("new-directory")**

Example:

```
import os

print('The current directory is :',os.getcwd())

os.chdir('new')

print('The current directory after changing is :',os.getcwd())
```

Output:

```
The current directory is : D:\Python_Batch

The current directory after changing is : D:\Python_Batch\new
```

Deleting directory:

- This method is used for deleting an empty directory. If the path does not correspond to an empty directory then OSError is raised.
- The syntax to use the rmdir() method is given below.

Syntax : os.rmdir("directory name")

Example1:

```
Import  os

#removing the new directory

os.rmdir("directory_name")
```

Note: It will remove the specified directory.

Q. Remove the directory if no files are available in dorectory ?

```
import os

print(os.getcwd())

os.chdir('new')

print(os.getcwd())

#os.mkdir('create')
```

os.rmdir('create')

Output:

OSError: [WinError 145] The directory is not empty: 'create'

Note : If the directory contains any files then it is not possible to remove directory with out deleting existing files.

os.makedirs() :

This method creates a directory recursively. It means that while creating a leaf directory if any of the intermediate level directories specified in the path is missing then the method creates them all.

Example:

```
import os
os.makedirs("/Python@7.30AM/NewFiles/Files")
```

Note: Here actual path is **/Python@7.30AM/Files** but "**NewFiles**" folder not available already. So this above method will create missing folders also automatically.

os.walk() :

- This method generates the filenames in a directory tree by walking the tree in either a top-down or bottom-up manner.
- os.walk returns a generator that creates a tuple of values (dirpath, dirnames, filenames)

Example:

```
walk_path = os.walk("/Python@7.30AM/Files")
print(tuple(walk_path))
```

Output:

```
(('/Python@7.30AM/Files', ['demo', 'Django'], ['demo.txt', 'demo2.txt', 'demo3.txt',
'demo4.txt', 'files_program.py', 'sample.txt']), ('/Python@7.30AM/Files\\demo', [], []),
('/Python@7.30AM/Files\\Django', ['sample'], ['abc.txt']),
('/Python@7.30AM/Files\\Django\\sample', [], []))
```

Example2:

```
for  dirpath, dirnames, filenames in os.walk("/Python@7.30AM/Files"):
    print("Directory Path:")
```

```
    print(dirpath)
    print("Directories Names:")
    print(dirnames)
    print("Files in Directories:")
    print(filenames)
```

Output:

```
Directory Path:
/Python@7.30AM/Files
Directories Names:
['demo', 'Django']
Files in Directories:
['demo.txt', 'demo2.txt', 'demo3.txt', 'demo4.txt', 'files_program.py', 'sample.txt']
Directory Path:
/Python@7.30AM/Files\demo
Directories Names:
[]
Files in Directories:
[]
Directory Path:
/Python@7.30AM/Files\Django
Directories Names:
['sample']
Files in Directories:
['abc.txt']
Directory Path:
/Python@7.30AM/Files\Django\sample
Directories Names:
[]
Files in Directories:
[]
```

os.path.join() :

- This method joins various path components with exactly one directory separator (“/”) following each non-empty part except for the last path component.
- If the last path component is empty then a directory separator (“/”) is put at the end. This method returns a string with the concatenated path.

Example:

```
import os
base_path = "/Python@7.30AM/Files/"
new_path = os.path.join(base_path, "APIS")
print(new_path)
```

Output:

/Python@7.30AM/Files/APIS

os.path.basename() :

- This method is used to get the base name in a specified path. The method returns a string value that represents the base name of the specified path.

Example:

```
import os
base_name = os.path.basename("/Python@7.30AM/Files/APIS")
print(base_name)
```

Output:

APIS

os.path.split() :

- This method splits the pathname into a pair of head and tail. Here, the tail is the last pathname component and the head is everything that comes before it.
- The method returns a tuple of the head and tail of the specified path.

Example:

```
import os
base_name = os.path.split("/Python@7.30AM/Files/APIS")
print(base_name)
```

Output:

('/Python@7.30AM/Files', 'APIS')

os.path.dirname() :

This method returns the directory name from the path given.

Example:

```
import os
directory_name = os.path.dirname("/Python@7.30AM/Files/sample.txt")
print(directory_name)
```

Output:

/Python@7.30AM/Files

os.path.commonprefix() :

- This method returns the longest path prefix which is a prefix for all the paths in the specified list.

Example:

```
import os
common_prefix = os.path.commonprefix(
    ["/Python@7.30AM/Files/", "/Python@7.30AM/Files/Django/sample",
     "/Python@7.30AM/Files/demo"])
print(common_prefix)
```

Output:

/Python@7.30AM/Files/

os.path.isfile() :

- This method checks whether the specified path corresponds to an existing file or not. This method returns a boolean value.

Example:

```
import os
file_value = os.path.isfile("/Python@7.30AM/Files/Django/abc.txt")
```

```
directory_value = os.path.isfile("/Python@7.30AM/Files/Django/sample")
print(file_value)
print(directory_value)
```

Output:

```
True
False
```

os.path.isdir() :

- This methods checks and reports whether the specified pathname corresponds to an existing directory or not. The method returns a boolean value.

Example:

```
import os
file_value = os.path.isdir("/Python@7.30AM/Files/Django/abc.txt")
directory_value = os.path.isdir("/Python@7.30AM/Files/Django/sample")
print(file_value)
print(directory_value)
```

Output:

```
False
True
```

os.path.exists() :

- This method returns True for existing paths. It returns False for broken symbolic links.

Example:

```
import os
is_path_exists = os.path.exists("/Python@7.30AM/Files/Django/abc.txt")
print(is_path_exists)
```

Output:

```
True
```

File attributes:

1. file.closed

Returns true if file is closed, false otherwise.

2. file.mode

Returns access mode with which file was opened.

3. file.name

Returns name of the file.

Example:

```
file_object = open('demo.txt')
print(file_object.name)
print(file_object.mode)
print(file_object.closed)
file_object.close()
print(file_object.closed)
```

Output:

```
demo.txt
r
False
True
```

Example:

```
Import  os
print('Current directory is :',os.getcwd())
#Changing directory into "new"
print('Changing directory into "new"',os.chdir('new'))
print('Now Current directory is :',os.getcwd())
'''
#making the new directory as "create"
```

```
print('making the "create" as new directory',os.mkdir('create'))
print('"create" directory created successfully.')
'''
#Changing directory into "create"
print('Changing directory into "create"',os.chdir('create'))
print('Now Current directory is :',os.getcwd())
#Remove the required file from "create" directory
print('Remove the required file from "create" directory',os.remove('xyz.txt'))
print('required file deleted successfully from "create" directory.')
#Go back parent directory of "create" directory.
print('Go back parent directory of "create" directory.' ,os.chdir(os.pardir))
print('Now Current directory is :',os.getcwd())
#Now remove the "create" child directory.
print('Now remove the "create" child directory.',os.rmdir('create'))
print('"create" directory deleted successfully.')
print('Now Current directory is :',os.getcwd())
```

Output:

```
Current directory is : D:\Python_Batch
Changing directory into "new" None
Now Current directory is : D:\Python_Batch\new
Changing directory into "create"
Now Current directory is : D:\Python_Batch\new\create
Remove the required file from "create" directory
required file deleted successfully from "create" directory.
Go back parent directory of "create" directory. None
Now Current directory is : D:\Python_Batch\new
Now remove the "create" child directory. None
```

"create" directory deleted successfully.

Now Current directory is : D:\Python_Batch\new