OS Module in Python 🖥️

Python has a built-in os module with methods for interacting with the operating system, like creating files and directories, management of files and directories, input, output, environment variables, process management, etc.

---

1. Importing the OS Module

import os

---

2. Working with the File System

2.1. Get the Current Working Directory

print(os.getcwd()) # Output: Current directory path

2.2. Change the Current Directory

os.chdir("/path/to/directory") # Change working directory
print(os.getcwd()) # Verify the change

2.3. List Files and Directories

print(os.listdir()) # Lists files & folders in current directory
print(os.listdir("/path/to/directory")) # List files in a specific directory

2.4. Create a Directory

os.mkdir("new_folder") # Creates a folder
Create multiple directories (Nested Folders)
os.makedirs("parent_folder/child_folder")

2.5. Remove a Directory

os.rmdir("new_folder") # Removes a folder (only if empty)
Remove multiple directories
os.removedirs("parent_folder/child_folder")

2.6. Rename a File or Directory

os.rename("old_name.txt", "new_name.txt") # Renames a file
os.rename("old_folder", "new_folder") # Renames a directory

---

3. Working with Files

3.1. Check if a File Exists

print(os.path.exists("example.txt")) # Output: True/False

3.2. Check if a Path is a File or Directory

print(os.path.isfile("example.txt")) # True if it's a file
print(os.path.isdir("my_folder")) # True if it's a directory

3.3. Get File Size

```
print(os.path.getsize("example.txt"))  # Output: Size in bytes
```

3.4. Get Absolute Path

```
print(os.path.abspath("example.txt"))
```

---

4. Environment Variables

4.1. Get an Environment Variable

```
print(os.environ.get("HOME"))  # Output: Home directory (Linux/Mac)
print(os.environ.get("USERNAME"))  # Output: Username (Windows)
```

4.2. Set an Environment Variable

```
os.environ["MY_VARIABLE"] = "Hello, World!"
print(os.environ["MY_VARIABLE"])  # Output: Hello, World!
```

---

5. Executing System Commands

5.1. Run a Shell Command

```
os.system("ls")  # Linux/Mac - List files
os.system("dir")  # Windows - List files
```

5.2. Open a File Using Default Application

```
os.startfile("example.txt")  # Windows
os.system("open example.txt")  # Mac
os.system("xdg-open example.txt")  # Linux
```

---

6. Process Management

6.1. Get Process ID

```
print(os.getpid())  # Output: Current process ID
```

6.2. Get Parent Process ID

```
print(os.getppid())  # Output: Parent process ID
```

6.3. Create a Child Process

```
pid = os.fork()  # Only on Linux/macOS
if pid == 0:
    print("Child process")
else:
    print("Parent process")
```

---

7. Path Operations with os.path

The os.path module helps in handling **file paths**.

7.1. Join Paths

```
path = os.path.join("folder", "file.txt")
print(path)  # Output: folder/file.txt (Linux/Mac) or folder\file.txt (Windows)
```

7.2. Get File Extension

```
print(os.path.splitext("example.txt"))  # Output: ('example', '.txt')
```

7.3. Get Directory Name

```
print(os.path.dirname("/path/to/file.txt"))  # Output: /path/to
```

7.4. Get Base File Name

```
print(os.path.basename("/path/to/file.txt"))  # Output: file.txt
```

---

8. Summary of OS Module Functions

Function	Description
os.getcwd()	Get current directory
os.chdir(path)	Change directory
os.listdir(path)	List files & directories
os.mkdir(name)	Create a directory
os.rmdir(name)	Remove an empty directory
os.rename(old, new)	Rename file/directory
os.path.exists(path)	Check if path exists
os.path.isfile(path)	Check if it’s a file
os.path.isdir(path)	Check if it’s a directory
os.environ.get(var)	Get environment variable
os.system(command)	Run shell command
os.getpid()	Get process ID

Conclusion

The os module is powerful for file handling, directory management, environment variables, and system commands. Would you like practice exercises or a mini-project using the os module? 🚀

Python os Module: Complete Reference

The Python `os` module provides a way of using operating system-dependent functionality such as interacting with files and directories, accessing environment variables, and managing processes.

Working with Directories and Files

- os.mkdir(path, mode=0o777): Creates a new directory.
- os.remove(path): Deletes a file.
- os.listdir(path='.') : Lists all files and directories in the specified path.

File and Directory Manipulation

- os.path: Provides utilities for manipulating file paths.
- os.rename(src, dst): Renames a file or directory.
- os.remove(path): Deletes a file.

Path Methods (os.path)

- os.path.join(*paths): Combines path components into a single path.
- os.path.abspath(path): Returns the absolute version of the path.
- os.path.normpath(path): Normalizes a path (e.g., removing redundant separators).
- os.path.exists(path): Checks if a path exists.
- os.path.isfile(path): Checks if a path is a file.
- os.path.isdir(path): Checks if a path is a directory.
- os.path.basename(path): Returns the last component of a path.
- os.path.dirname(path): Returns the directory component of a path.
- os.path.split(path): Splits the path into (head, tail) components.
- os.path.splitext(path): Splits the file path into (root, extension).
- os.path.isabs(path): Checks if a path is absolute.
- os.path.relpath(path, start): Returns the relative path from 'start' to 'path'.
- os.path.getsize(path): Returns the size of the file in bytes.
- os.path.getmtime(path): Returns the last modification time of the file.

Environment Variables and System Information

- os.getenv(key, default=None): Gets the value of an environment variable.
- os.environ: A mapping object representing environment variables.

Other Commonly Used Methods in os Module

- os.getcwd(): Returns the current working directory.
- os.chdir(path): Changes the current working directory.
- os.rmdir(path): Removes an empty directory.
- os.makedirs(path, exist_ok=False): Creates directories recursively.
- os.walk(top, topdown=True, onerror=None, followlinks=False): Generates file names in a directory tree.
- os.system(command): Executes a system command.
- os.getpid(): Returns the current process ID.
- os.urandom(n): Returns a string of `n` random bytes.
- os.access(path, mode): Checks the accessibility of a path.

date.py

```python
import os


# combined=os.path.join("c:/mydir","./sample.txt")
# print(combined)

# print(os.path.abspath("./sample.txt"))
# print(os.path.abspath("./dates.py"))
# print(os.path.normpath("d:\Batch\\Python_classes\\Day_30\dates.py"))
# print(os.path.exists("./sample.txt"))

# print(os.path.isfile("./sample.txt"))
# print(os.path.isdir("../DAY_30"))
# print(os.path.basename("d:\Batch\Python_classes\Day_30\sample.txt"))

# print(os.path.dirname("d:\Batch\Python_classes\Day_30\date.py")+"\sample.txt")

# print(os.path.split("d:\Batch\Python_classes\Day_30\date\date.py\sample.txt"))
```

```python

# print(os.path.split("d:\Batch\Python_classes\Day_30\date\date.py\sample.txt"))

# print(os.path.splitext("d:\Batch\Python_classes\Day_30\sample.txt")[1]==".png")
# print(os.path.isabs("/date.py"))
# print(os.path.relpath("d:\Batch\Python_classes\Day_30\date.py"))
# print(os.path.getsize("d:\Batch\Python_classes\Day_30\date.py"))
# print(os.path.getmtime("d:\Batch\Python_classes\Day_30\date.py"))
```