

Day 28



os Module in Python:

What Is the os Module?

- os is a built-in Python module for interacting with the operating system
- Lets you perform tasks like:
 - File and directory management
 - Environment variables
 - Process management

Importing the module:

```
main.py
1 import os
```

A screenshot of a code editor window. On the left, there's a file tree showing a folder named 'data' and several documents: '~\$Day28.docx', 'Day28.docx', and 'main.py'. The 'main.py' file is open on the right. It contains the single line of code 'import os'. The code editor has a dark theme with syntax highlighting for Python keywords.

Example: Creating a folder using the os module.

```
main.py
1 import os
2 os.mkdir("data")
```

A screenshot of a code editor window. The file tree on the left shows a 'data' folder. The 'main.py' file on the right contains the code 'import os' on line 1 and 'os.mkdir("data")' on line 2. Below the code editor, the terminal window shows the command 'python .\main.py' being run, followed by the output 'PS E:\Python\Day21-30\Day28> python .\main.py' and 'PS E:\Python\Day21-30\Day28>'. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS, with 'TERMINAL' currently selected.

Example: code to check if a folder exists in our folder or not.

```
main.py
1 import os
2 # os.mkdir("data")
3 print(os.path.exists("data"))
```

A screenshot of a code editor window. The file tree on the left shows a 'data' folder. The 'main.py' file on the right contains the code 'import os' on line 1, a comment '# os.mkdir("data")' on line 2, and 'print(os.path.exists("data"))' on line 3. Below the code editor, the terminal window shows the command 'python .\main.py' being run, followed by the output 'PS E:\Python\Day21-30\Day28> python .\main.py' and 'True'. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS, with 'TERMINAL' currently selected.

Example: to rename

The screenshot shows a dark-themed interface of VS Code. On the left, there's a sidebar with a folder icon labeled 'renamed_data' containing files: '~\$Day28.docx', 'Day28.docx', and 'main.py'. The 'main.py' file is selected. The main editor area contains the following Python code:

```
4
5 import os
6 os.rename("data", "renamed_data")
```

Below the editor, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORT'. The 'TERMINAL' tab is active, showing the command and its output:

```
● PS E:\Python\Day21-30\Day28> python .\main.py
```

Example: to print the list of the folders in a specific folder.

The screenshot shows a dark-themed interface of VS Code. The sidebar shows a folder 'renamed_data' with files: 'Aditya.txt' and 'Utsav'. The 'main.py' file is selected. The editor contains the following Python code:

```
8 import os
9 folders = os.listdir("renamed_data")
10 print(folders)
```

Below the editor, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORT'. The 'TERMINAL' tab is active, showing the command and its output:

```
● PS E:\Python\Day21-30\Day28> python .\main.py
['Aditya.txt', 'Utsav']
```

Example: Current Working Directory

The screenshot shows a dark-themed interface of VS Code. The sidebar is empty. The 'main.py' file is selected. The editor contains the following Python code:

```
8 import os
9 print(os.getcwd())
```

Below the editor, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORT'. The 'TERMINAL' tab is active, showing the command and its output:

```
● PS E:\Python\Day21-30\Day28> python .\main.py
E:\Python\Day21-30\Day28
```

Example: Creating and Removing Directories

The screenshot shows a dark-themed interface of VS Code. The sidebar is empty. The 'main.py' file is selected. The editor contains the following Python code:

```
11 import os
12 os.mkdir("test_folder")      # Create one folder
13 os.makedirs("folder1/folder2") # Create nested folders
14
15 os.rmdir("test_folder")      # Remove one folder
16 os.removedirs("folder1/folder2") # Remove nested folders
```

--The End--