OS Module

It provides a way to interact with the Operating System.

Regular Use-Cases:

- Creating/removing directories
- Changing working directories
- Listing directory contents
- Getting file information
- Path Manipulation

Basic Directory Operation

```
import os
```

```
# Create a New Directory (folder)
os.makedirs("folder_name")

# Create nested directories
os.makedirs("folder_name/sub_folder_name", exist_ok=True)

# Deleting Directory
os.rmdir('file-folder_name')

# Reaname
os.rename('oldfile_name', 'newfile_name')

# Information about file
os.stat('file_name')
print(os.stat('file_name'))
# Getting the size
print(os.stat('file_name').st_size)
```

```
# Change the current Working Directory
os.chdir("path/folder_name_you_want_to_use")
```

```
# Get current working directory
current_dir = os.getcwd()
```

```
# List contents of directory
contents = os.listdir()

return contents # use this when using function.
or
print(contents)
```

File Management for Data Science

- `os.path.exists()` returns True if the file exists, False if it doesn't

  ```
  # Checking File Existence:
  if os.path.exists("dataset.csv"):
      print("File found, proceeding with analysis")
  else:
      print("File not found, please check the path")
  ```

- `os.path.getsize()`

  ```
  # Getting File Size

  file_size = os.path.getsize("dataset.csv")
  ```

  - Returns the size of the file in bytes
  - Useful for:
    - Checking if a file is empty
    - Managing storage space
    - Monitoring file growth

  ```
  file_size = os.path.getsize("dataset.csv")
  size_in_mb = file_size / (1024 * 1024)  # Convert to MB
  print(f"Dataset size: {size_in_mb:.2f} MB")
  ```

- **Path Manipulation**

  ```
  directory, filename = os.path.split("data/raw/dataset.csv")
  name, extension = os.path.splitext(filename)
  ```

  - `os.path.split()`: Separates path into directory and filename
    - For "data/raw/dataset.csv":
      - directory = "data/raw"
      - filename = "dataset.csv"
  - `os.path.splitext()`: Separates filename into name and extension
    - For "dataset.csv":
      - name = "dataset"

        - extension = ".csv"

- **Path Joining**

  ```
  data_path = os.path.join("project", "data", "raw", "dataset.csv")
  ```

  - Creates platform-independent file paths
  - Windows will use backslashes (), Unix will use forward slashes (/)
  - More reliable than string concatenation

  ```
  # Instead of this (error-prone):
  bad_path = "project" + "/" + "data" + "/" +
                      "raw" + "/" + "dataset.csv"

  # Use this (platform-independent):
  good_path = os.path.join("project", "data", "raw", "dataset.csv")
  ```

Example

```
import os
import datetime

def check_dataset_status(file_path):
    if os.path.exists(file_path):
        # Get file info
        size_mb = os.path.getsize(file_path) / (1024 * 1024)
        last_modified = datetime.datetime.fromtimestamp(os.path.getmtime(file_path))

        # Get path components
        directory, filename = os.path.split(file_path)
        name, ext = os.path.splitext(filename)

        print(f"Dataset Status:")
        print(f"Location: {directory}")
        print(f"Filename: {name}")
        print(f"Format: {ext}")
        print(f"Size: {size_mb:.2f} MB")
        print(f"Last modified: {last_modified}")
    else:
        print(f"Dataset not found at {file_path}")

# Usage
dataset_path = os.path.join("project", "data", "dataset.csv")
check_dataset_status(dataset_path)
```

```
# Creating 100 nested directories in folder_name = data

import os
```

```
# this check if u have already created the folder data, if not
# then it will create it.
if (not os.path.exists('data'):
        os.mkdir('data')

# Looping to create multiple sub folder in data
for i in range(0,100):
        os.mkdir(f"data/day{i+1}") # Create 100 sub folder in data.
```

```
# Renaming those 100 folders

for i in range(0,100):
        os.rename(f"data/day{i+1}", f"data/week{i+1}")

# Creating space between week and no. like week1 --> week 1
for i in range(0,100):
        os.rename(f"data/week{i+1}", f"data/week {i+1}")
```

`os.walk()`

- It is a generator function that yields a 3-tuple: (root, dirs, files)
  - root: Current directory path being processed
  - dirs: List of subdirectories in current directory
  - files: List of files in current directory

```
for root, dirs, files in os.walk("some_directory"):
    # root: string - current directory path
    # dirs: list - names of subdirectories in current directory
    # files: list - names of files in current directory
```

Common uses

Finding specific files:

```
# Find all Python files
for root, _, files in os.walk("project"):
    for file in files:
        if file.endswith(".py"):
            print(os.path.join(root, file))
```

Calculating Directory sizes

```
total_size = 0
for root, _, files in os.walk("project"):
    for file in files:
        total_size += os.path.getsize(os.path.join(root, file))
```

Modifying directories list:

```
# Skip certain directories during traversal
for root, dirs, files in os.walk("project"):
    dirs[:] = [d for d in dirs if not d.startswith('.')]
    # Skip hidden directories
```

`os.path`

`os.path.basename('path')` - this give the basename of the path

```
os.path.basename('/desktop/python/main.py')
# main.py
```

`os.path.dirname('path')`

```
os.path.dirname('/desktop/python/main.py')
# /desktop
```

`os.path.exists('path')`

```
print(os.path.exists('/desktop/python/main.py'))
# give TRUE if exist.
```

`os.path.splitext('path')`

```
print(os.path.splitext('/python/main.py'))
# '/python/main', '.py'
```