Module 2: Using Python to Interact with the Operating System

📂 Data Storage and Absolute Paths

- **Data** is stored in **files**, which live inside **directories/folders**.
- **Absolute Paths:** Full file address starting from the **root**.
    - Root examples:
        - Windows → `C:`
        - Linux/Mac → `/`
    - Examples:
        - Windows → `C:/Users/Arman/Documents/Projects/Data.txt`
        - Linux/Mac → `/home/arman/documents/projects/data.txt`
- **Relative Paths:** Start from the **current directory**.
    - Example: If you're in `C:/Users/Arman/Documents/`, the file `Projects/Data.txt` is referenced simply as `Projects/Data.txt`.

🛠️ Python Path Preference

- Both `C:\Users\Arman\Documents` and `C:/Users/Arman/Documents` refer to the same location.
- Python prefers **forward slashes `/`** because:
    - Works on **Windows, Mac, and Linux**.
    - Avoids escape character issues with `\\`.

🗂️ OS Module File Operations

- `os.remove('data.txt')` → Deletes a file.
- `os.rename('data.txt', 'newfile.txt')` → Renames a file.
- `os.path.exists('data.txt')` → Checks if the file exists.
- `os.path.getsize('data.txt')` → Returns file size in bytes.
- `os.path.getmtime('data.txt')` → Gets last modified timestamp.
- `os.path.abspath('data.txt')` → Returns absolute path.
- `os.mkdir()` → Creates a directory.
- `os.chdir()` → Changes the current working directory.
- `os.rmdir()` → Removes an **empty** directory.
- `os.listdir(CurrentDir)` → Lists files in a directory.
- `os.path.join(dir, 'data.txt')` → Safely joins directory + filename.
- `os.chmod()` → Changes file/folder permissions.

📄 Reading and Writing Files

- When Python reads a file, the OS gives a **File Descriptor**.
- Python converts it into a **File Object** with methods:
    - `read()`
    - `write()`

    - `close()`

- Always close files using `.close()`.

📘 File Processing Modes

- `r` → Read mode
- `w` → Write mode (overwrites or creates)
- `a` → Append mode (adds new data at the end)
- `encoding="UTF-8"` ensures proper text handling.

🗒️ Reading Large Files

- Small files → read fully.
- Large files → read **line by line** using loops or `.readline()`.

🔐 Permissions

Permissions determine who can:

- Read
- Write
- Execute

🌍 Unicode & Encoding

- **Unicode:** A universal dictionary of characters.
- **UTF-8:** The most common way to store Unicode characters as bytes.

📊 Working with CSV Files

CSV (Comma-Separated Values) is a simple text-based data format.

- **Rows** → Separated by new lines.
- **Columns** → Separated by commas.

🧪 Python Example (Reading CSV)

```
import csv
f = open("a.txt", "r")
csv_f = csv.reader(f)
for row in csv_f:
    name, phone = row
    print(f"Name:{name} Phone:{phone}")
f.close()
```

🐧 Linux Commands & Shebang

💻 Terminal Commands

- `cat` → Prints file content.
- `nano` → Opens a simple terminal-based editor.

🔽 Shebang Line

- `#!/usr/bin/env python3`
    - Tells the system which interpreter to use.

🔧 Making a Script Executable

- `chmod +x generate_report.py`
    - Gives the script **execute** permission.
