>>>> Day 2:

Understanding `if __name__ == "__main__":` vs Direct Code Execution

🔹 1. Without `if __name__ == "__main__":`

💻 **Example:**

```
def celsius_to_farenhite(c):
    return (c * 9/5) + 32

temp_celsius = float(input("Enter temperature in Celsius: "))
print("Temperature in Fahrenheit:",
celsius_to_farenhite(temp_celsius))
```

📖 **Meaning:**

- Code runs **immediately** when the file is executed **or imported**.
- No control — everything executes automatically.

⚠️ **Drawbacks:**

- If this file is imported into another Python file, it will **still take input** or **run prints**, which can cause errors or confusion.

✅ **Use When:**

- Writing **small one-time scripts** that will **never be imported**.
- Quick testing or learning purpose only.

🔹 2. With `if __name__ == "__main__":`

💻 **Example:**

```
def celsius_to_farenhite(c):
    return (c * 9/5) + 32

def farenhite_to_celsius(f):
    return (f - 32) * 5/9

if __name__ == "__main__":
    temp_celsius = float(input("Enter temperature in Celsius: "))
    print("Temperature in Fahrenheit:",
celsius_to_farenhite(temp_celsius))
```

📖 **Meaning:**

- The code inside this block runs **only when this file is executed directly**, not when imported.
- Functions or classes defined above are **always available for import**.

🧩 **How it works:**

- Every Python file has a built-in variable `__name__`.
- When you run the file directly → `__name__` = `"__main__"`.
- When you import the file → `__name__` = `"filename"` (not `"__main__"`).

✅ **Advantages:**

- Prevents unwanted code execution when importing.
- Makes code **modular**, **clean**, and **reusable**.
- Commonly used in professional and large Python projects.

✅ **Use When:**

- You want to make your file **reusable as a module**.
- Your file contains **functions/classes + some executable/test code**.

📘 Summary Table

Feature	Without `if __name__ == "__main__":`	With `if __name__ == "__main__":`
Code runs on import	✅ Yes	❌ No
Good for reusable code	❌ No	✅ Yes
Used in large projects	❌ Rare	✅ Always
Suitable for small test scripts	✅ Yes	✅ Yes
Imports only functions/classes	❌ No	✅ Yes

🔹 3. Using `{}` in `print()` (f-strings)

💻 **Example:**

```
temp = 25

print(f"Temperature in Celsius: {temp}")
```

📖 **Meaning:**

- Place variables inside `{}` in a string prefixed with `f`.
- No need for explicit `str()` conversion.
- Can include expressions directly (`{temp+5}`).

✅ **Advantages:**

- Modern, clean, and readable.
- Works well for **expressions inside print**.
- Recommended for Python 3.6+.

🔑 Final Takeaway:

🟢 **Use `if __name__ == "__main__":` for reusable code modules.**
🟢 **Use f-strings `{}` in print for clean variable output.**
🔴 Avoid auto-running scripts outside `__main__` when importing.