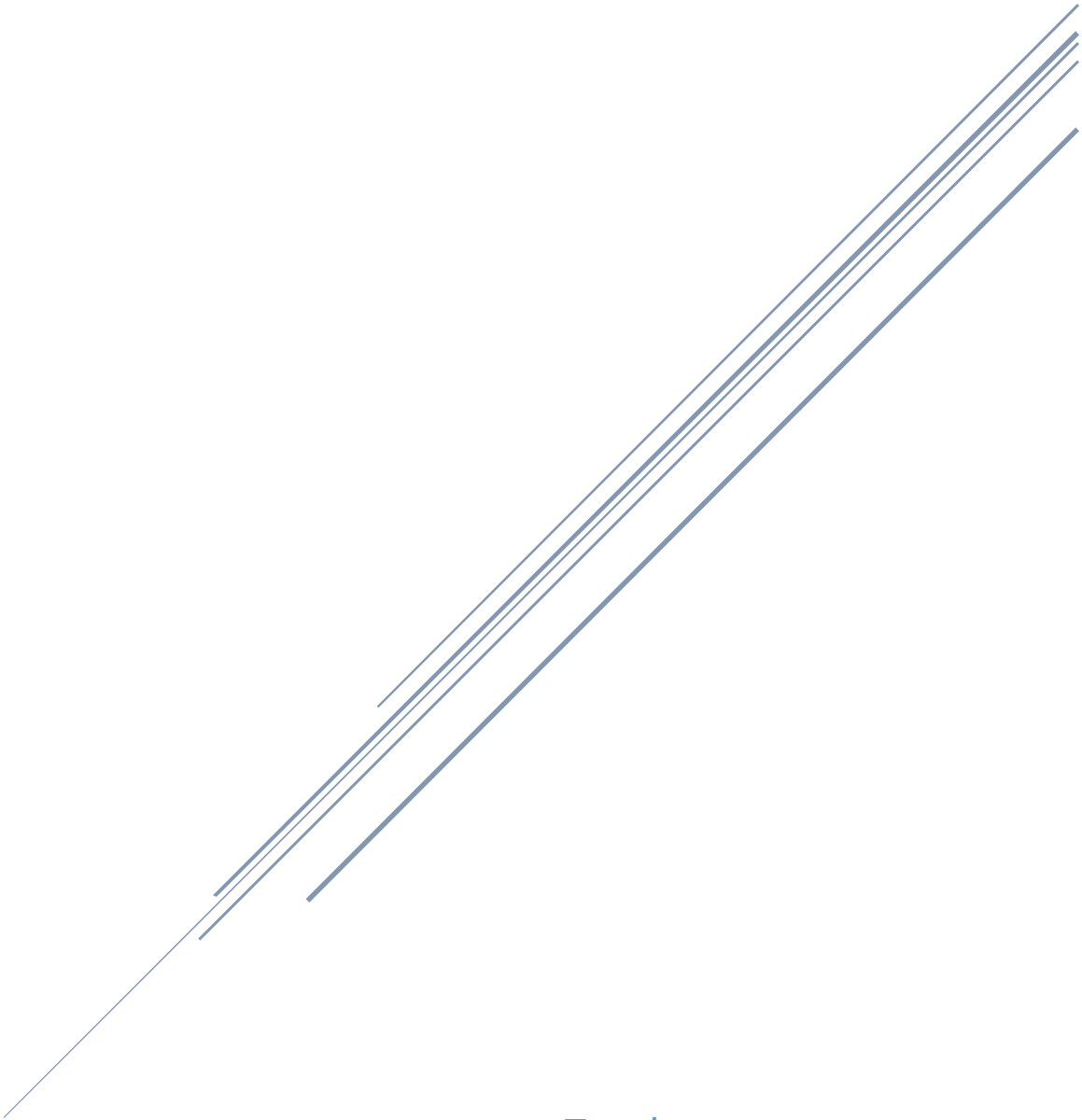
CODE AND SUMMARY OF CLASS 6

Sahir Ahmed Sheikh

Saturday (2 – 5)



Teachers:

Muhammad Bilal And Ali Aftab Sheikh

Code And Summary Of Class 6 – Saturday (2 – 5) | Quarter 3

Assalamualaikum, I hope everyone is doing well. Today's class was conducted by Sir **Ameen** and Sir **Ali Aftab Sheikh**. Sir Ameen Alam taught us about Project Structure and Real-World Coding, explaining how we can create industry-standard projects. He emphasized that we will no longer focus on syntax but will instead engage in concept-based coding at an industry level. Sir Ali Aftab Sheikh guided us through a revision session in preparation for the test scheduled for next week.

🏛️ Class Theme: Practical Learning Through Concepts

This class focused on transitioning from syntax-based coding to **conceptual and industry-standard coding practices**. Students were introduced to how real-world projects are planned, structured, and tested. Every topic was explained using **real-life metaphors**, **step-by-step demonstrations**, and **interactive examples** in Python.

🎁 Project Structure & Real-World Coding

In this section, students learned how to structure a real-world Python project like professional developers do in the industry. The goal was to understand how actual software is developed and maintained using clean, modular practices.

🧰 Setting Up the Project

- The instructor opened **Visual Studio Code** and created a new folder named Project One.
- Emphasis was given to **proper folder naming conventions** and **maintaining a clean directory structure**.
- Example: Creating an inventory project that simulates a real store or warehouse management system.

📁 Folder Structure and File Naming

- Folders and files should be named meaningfully and consistently (e.g., inventory.py, tests/, models/)
- Industry-standard naming makes it easy for teams to understand and maintain code.

🔧 Understanding Coding from a Conceptual View

- A new perspective was introduced: stop seeing programming only as syntax.
- Begin thinking of programming as **solving a problem based on requirements**.
- Example: “If I’m asked to build a bucket that holds products, how would I define it in code?”

🪣 Real-World Metaphor: The Bucket

- A **class** was compared to a **bucket**.
- Buckets have:
  - A **name** (class name)
  - **Space inside** (attributes or properties)
  - **Actions/methods** (like adding or removing items)

💻 *Real-Life Scenario:*

- The teacher gave an example: a **restaurant owner** and a **departmental store owner** both want to store items.
- They ask, “Do I need a bucket?” — the bucket represents a storage container (class).
- The **class definition** becomes the blueprint for this bucket.
- It should include methods to add products, remove products, and maybe show contents.

🚰 Unit Testing (with Real Use Case)

- Students learned the importance of writing **unit tests** along with coding.

- Real-world practice: every developer writes tests to ensure their code works.
- A **test case** was written to simulate adding two products (e.g., “iPhone 8” and “Android 8”) to a bucket.

✅ *Test Workflow:*

1. Create a bucket instance.
2. Add products using the method.
3. Use assert statements to check whether the products exist inside.
4. Make sure nothing gets lost (simulate if the bucket is broken).
5. Confirm the total number of items.

🚰 Why Unit Testing Is Important

- We explained that unit tests ensure your program behaves as expected.
- A **“bucket” metaphor** was used: classes are buckets, and products (data) go inside using specific methods.
- **Testing Example**:
  - Create a test bucket
  - Add 2 items: iPhone 8 and Android 8
  - Verify both items are inside
  - Use pytest to automate and confirm the test

💻 Running Pytest

- First, include pytest in requirements.txt
- Install with command: pip install -r requirements.txt
- Run test with: pytest
- Output must confirm that items were correctly added and retained

🔁 Operator Fundamentals (Recap + Deep Dive)

➕ Operators:

Operators are symbols used to perform operations on variables and values.

Types of Operators:

1. **Arithmetic Operators**
    - + → Adds two values
    - - → Subtracts second from first
    - * → Multiplies values
    - / → Divides (floating-point)
    - // → Floor Division (removes decimals)
    - % → Modulus (remainder)
    - ** → Exponentiation (power)

```
a = 10
b = 3
print(a + b)   # 13
print(a * b)   # 30
print(a / b)   # 3.33
print(a // b)  # 3
print(a % b)   # 1
print(a ** b)  # 1000
```

2. **Assignment Operators**
    - = → Assigns value
    - += → Add and reassign
    - -= → Subtract and reassign
    - *= → Multiply and reassign
    - /=, //=, %=, **= → same logic for respective operations

```python
a = 10
b = 3
print(a + b)   # 13
print(a * b)   # 30
print(a / b)   # 3.33
print(a // b)  # 3
print(a % b)   # 1
print(a ** b)  # 1000
```

3. **Comparison Operators**
    - == → Equal to
    - != → Not equal to
    - > → Greater than
    - < → Less than
    - >= → Greater than or equal to
    - <= → Less than or equal to
4. **Logical Operators**
    - and → True if both conditions are true
    - or → True if one of the conditions is true
    - not → Inverts the boolean value


```python
a = 10
b = 3
print(a + b)   # 13
print(a * b)   # 30
print(a / b)   # 3.33
print(a // b)  # 3
print(a % b)   # 1
print(a ** b)  # 1000
```

💡 Programming Concepts

🔍 What Is a Bucket Metaphor?

- A **class** is like a **bucket**

- **Product (Data)** goes **inside** the bucket
- A **method** (function) is used to put or remove items
- Buckets (classes) have:
    - A **name**
    - **Space** (attributes)
    - **Actions** (methods like add_to_bucket)

💡 **Example:**

```
a = 10
b = 3
print(a + b)   # 13
print(a * b)   # 30
print(a / b)   # 3.33
print(a // b)  # 3
print(a % b)   # 1
print(a ** b)  # 1000
```

🚰 Test-Driven Development (Expanded)

- First, **create a test case**
- Create bucket → Add iPhone 8 → Add Android 8
- Confirm with assert that both items are inside

```
a = 10
b = 3
print(a + b)   # 13
print(a * b)   # 30
print(a / b)   # 3.33
print(a // b)  # 3
print(a % b)   # 1
print(a ** b)  # 1000
```

🕸️ Strings and Their Methods

✍️ String Declaration

- 'Single', "Double", '''Triple''' quotes supported
- Triple quotes allow multi-line strings

⚙️ Useful Methods:

- .split() → Breaks string into list
- .join() → Joins list into string
- .replace() → Replaces substrings

🔤 Escape Sequences

- \n = Newline
- \t = Tab
- \b = Backspace
- \\ = Backslash
- \" = Double quote

🌐 Raw Strings

- Prefix with r to prevent escape sequence interpretation

```
print(r"Line1\nLine2")  # Prints: Line1\nLine2
```

🔄 Control Flow (If, Else, Elif)

✅ If Statement

```
if age >= 18:
    print("Eligible")
```

❌ Else Statement

```
if age < 18:
    print("Underage")
else:
    print("Eligible")
```

🔁 Elif (Else If)

```
if age < 18:
    print("Underage")
elif age == 18:
    print("Just turned 18")
else:
    print("Adult")
```

🚨 Exception Handling

🔎 Try-Except Block

```
try:
    x = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
```

🔚 Finally Block

Always runs

```
finally:
    print("Cleanup code")
```

➕ Else Block

Runs if no error

```
else:
    print("No error occurred")
```

📌 Multiple Excepts

```
try:
    int("abc")
except ValueError:
    print("Value error")
except TypeError:
    print("Type error")
```

❗ Raising Custom Exceptions

```
raise Exception("This is a custom error")
```

📚 Final Test Preparation Guide

✅ Do This Before the Quiz:

- **Revise all 9 Steps** shared in the repository
- Practice all **code cells** and understand their logic
- Focus more on **theory questions** in the end
- Use tools like **ChatGPT**, **Claude**, **DeepSeek**, or **Grok** for support

🚫 Don’t Do:

- Don’t rely only on what’s covered in class
- Don’t skip practice thinking syntax is enough

🌟 **Tip:**

Syntax can be Googled, but logic is what makes you a developer

---

📚 Class Resources & Assignments

📌 Today's Class Code

The code for today's class can be accessed on Google Colab:
[Class Code]

🎯 Assignments

✅ *Assignment 1: Secure Data Encryption*

Work on the Secure Data Encryption project from the repository:
[Secure Data Encryption – GitHub Repo]

✅ *Assignment 2: Complete Previous Assignments*

Make sure all previous assignments are completed and shared on LinkedIn, and also submitted via the class submission form.

⚠️ Important Reminder

Please make sure to review and understand all 9 steps/topics before the exam. Due to limited class time, we couldn’t cover everything in class — so it's essential to go through the remaining topics on your own.

📅 Deadline: Before the next class

📌 Submission:

- Upload all assignments on LinkedIn

- Submit via the Assignment Submission Form

Stay consistent and keep learning!

🏁 **Final Words:**

The rest is up to **you** now. Learn actively. Engage consistently. Be confident in class participation—even wrong answers help you learn. Practice logic daily. Test will come soon.

Allah Hafiz — See you in the next class!

Thank You for Reading!

Hope you understood Class 6 well.

" Success in any profession starts with a solid education; it's the foundation that equips you to innovate, lead, and excel." – *Indra Nooyi*