TASK 1: Variable Modeling & Data Representation

Concepts: Variables, Data Types, Dynamic Typing, `type()`

Model a **student entity** using variables:

- Full name
- Age
- CGPA
- Current semester
- Enrollment status

Print each value along with its data type.
Reassign one variable to a different type and observe the change.

Reasoning Questions

1. How does Python's dynamic typing influence memory allocation and runtime behavior?
2. Why is strict type enforcement preferred in system-level software but not in scripting languages?

TASK 2: User Input, Type Casting & Runtime Errors

Concepts: `input()`, Type Casting, Runtime Exceptions

Create a **basic financial calculator**:

- Input: monthly income, monthly expenses
- Output: savings or deficit

Show what happens if numeric input is used **without type casting**.

Reasoning Questions

1. Why does Python delay type errors until runtime, unlike compiled languages?
2. How could unvalidated user input compromise real-world systems (e.g., billing, voting)?

TASK 3: Arithmetic Logic & Expressions

Concepts: Operators, Expressions, Evaluation Order

Create a **student performance score**:

- Input: assignment score, lab score, exam score
- Use weighted formula to compute final score

Display result with proper formatting.

Reasoning Questions

1. Why is operator precedence critical in scientific or financial software?
2. How could floating-point precision errors affect real-world applications?

TASK 4: Branching & Decision Making

Concepts: `if`, `elif`, `else`, Boolean Logic

Build an **academic decision system**:

- Input: attendance %, total marks
- Decide eligibility and grade

Create a **scholarship eligibility checker**:

- Conditions based on CGPA, income, and attendance

TASK 5: Iteration Using `for` Loops

Concepts: Iteration, Accumulators, Loop Control

Create a **marks analyzer**:

- Input marks for `n` subjects
- Output total, average, highest, lowest

TASK 6: Input Validation with `while` Loops

Concepts: `while`, Validation, Loop Termination

Accept subject marks **only between 0 and 100**.
Re-prompt until valid input is entered.

Conceptual Questions

1. Why is validation better handled with `while` loops than `for` loops?

TASK 7: String Processing & Traversal

Concepts: Strings, Indexing, Iteration

Analyze a user-entered sentence:

- Count vowels, consonants, digits, spaces
- Convert case and remove extra spaces

Reasoning Questions

1. Why is string processing critical in cybersecurity and NLP?
2. How can improper string handling introduce vulnerabilities?

TASK 8: Functions

Concepts: Functions, Parameters, Return Values, Reusability

Create reusable functions for:

- Calculating average marks
- Determining grade
- Formatting output

The main program should **call these functions**, not duplicate logic.

Conceptual / Brainstorming Questions

1. Why is returning values better than printing inside functions?

TASK 9: Mini System Integration Task

Concepts: Integration of Basics, Program Flow

Build a **menu-driven student utility system**:

1. Enter student details
2. Enter marks
3. View result summary
4. Exit

Use:

- Variables
- Input & casting
- Branching
- Loops
- Functions
- Strings

Reasoning Questions

1. What changes would be required to convert this into a web application?

📤 Submission Criteria & Git Workflow

This assignment must be submitted via GitHub using a new branch in your earlier assignment existing repository or by creating a new repository.

Code Content Requirements

For **each task file**:

1. Code must:
   - Run without syntax errors
   - Follow proper indentation and readability
   - Use meaningful variable and function names
2. **Reasoning / Conceptual Questions**
   - Must be answered as **Python comments at the bottom of the same file**
   - Clearly labeled, for example:

File Naming Rules (Strict)

Each task must be implemented in a **separate Python file** using the exact naming convention:

Task	File Name
TASK 1	`task1_variables.py`
TASK 2	`task2_input_casting.py`
TASK 3	`task3_expressions.py`
TASK 4	`task4_branching.py`
TASK 5	`task5_for_loops.py`
TASK 6	`task6_while_validation.py`
TASK 7	`task7_strings.py`
TASK 8	`task8_functions.py`
TASK 9	`task9_menu_system.py`

Submit a GitHub Repository Link (Mandatory)

In the **“Your work”** section of Google Classroom, submit:

- 🔗 **Your GitHub repository URL**
- 🧵 **Branch name used for this assignment**

📌 Example Submission Text:

```
GitHub Repository: https://github.com/username/assignment-1
Branch Name: assignment-2-python-basics
```

⚠️ Do **NOT** upload `.py` files directly to Google Classroom.