Tab 1

Introduction to Computational Thinking

Before writing code, one must understand how computers process instructions. This is called **Algorithmic Thinking**.

1.1 The Input-Process-Output (IPO) Model

Every program follows this flow:

- **Input:** Data from a user, a file, or a sensor.
- **Process:** The logic, calculations, or data manipulation.
- **Output:** The result displayed on a screen, saved to a database, or sent over a network.

1.2 Compilers vs. Interpreters

- **Compilers (e.g., C++, Java):** Translates the entire code into machine language *before* execution. It is generally faster
- **Interpreters (e.g., Python, JavaScript):** Translates code line-by-line *during* execution. It is easier to debug and more flexible

Variables, Constants, and Memory

Memory management is the foundation of efficient coding.

2.1 Static vs. Dynamic Typing

- **Static Typing (Java/C#):** You must declare the type (e.g., int x = 5). This prevents errors early.
- **Dynamic Typing (Python/JS):** The type is determined at runtime (e.g., x = 5). This allows for faster prototyping.

2.2 Naming Conventions

Professional code follows strict naming rules:

- **camelCase:** userProfileImage (Standard for JavaScript/Java)
- **snake_case:** user_profile_image (Standard for Python)
- **PascalCase:** UserProfileImage (Standard for Classes)

Operators and Boolean Algebra

Logic is the "decision-making" engine of your code.

3.1 Truth Tables

Understanding how Booleans interact is vital for complex if statements.

A	B	A AND B	A OR B
True	True	True.	True
True	False	False	True
False	True	False	True
False	False	False	False

3.2 Operator Precedence

Just like PEMDAS in math, code has an order of operations. Comparison operators (like ==) are usually evaluated after arithmetic operators (like +).

Control Flow - Selection Structures

Selection allows your program to "branch" into different paths.

4.1 The If-Else Ladder

When you have multiple conditions that are mutually exclusive, use an else-if chain. This prevents the computer from checking unnecessary conditions once a "True" one is found.

4.2 The Ternary Operator

A shorthand for simple if-else statements:condition ? value_if_true : value_if_false

Control Flow - Iteration (Loops)

Loops allow for automation of repetitive tasks.

5.1 Definite vs. Indefinite Loops

Definite (For): Use when you have a collection (like a list of 100 emails) to process.
Indefinite (While): Use when the end depends on an external factor (like waiting for a user to type "STOP").

5.2 Infinite Loops and Break/Continue

Break: Exits the loop immediately.
Continue: Skips the rest of the current "lap" and moves to the next iteration.

Functions and Scope

Functions turn a "script" into a "program."

6.1 Function Anatomy

Identifier: The name.
Parameters: Placeholder variables for input.
Return Type: What the function "gives back" to the caller.

6.2 Variable Scope

Global Scope: Variables accessible from anywhere in the file.
Local Scope: Variables created inside a function that "die" when the function finishes.

Basic Data Structures

Storing single values isn't enough; we need to store collections.

7.1 Arrays/Lists

An ordered sequence of elements accessed by an index.
Important: Most languages are Zero-Indexed, meaning the first item is at position 0, not 1.

7.2 Key-Value Pairs (Maps/Dictionaries)

Instead of an index number, you use a unique key (like a username) to find a value (like user data). This is the basis for most database interactions.

The Software Development Life Cycle (SDLC)

Coding is only one part of being a developer.
Requirements: Defining what the software should do.
Design: Planning the architecture and UI.
Implementation: The actual coding.
Testing: Finding bugs (Unit Testing, Integration Testing).
Deployment: Putting the code on a server for users.
Maintenance: Fixing issues and adding new features.