Introductory Programming Concepts

1. Variable

A variable is a named storage location used to hold data that can change during program execution. It allows programmers to store values such as numbers, text, or other data types and reuse them later in the code. For example, a variable called `age` might store the value `21`, which can be updated if needed.

2. Data Type

A data type defines the kind of data a variable can store and determines what operations can be performed on it. Common data types include integers (whole numbers), floats (decimal numbers), strings (text), and Booleans (true/false values). Choosing the correct data type helps ensure efficient memory use and prevents errors.

3. Conditional Statement

A conditional statement allows a program to make decisions based on certain conditions. It checks whether a statement is true or false and executes specific code accordingly. Common examples include `if`, `else if`, and `else` statements, which help programs respond differently to varying inputs or situations.

4. Loop

A loop is a control structure that repeats a block of code multiple times until a specified condition is met. Loops help automate repetitive tasks, making programs shorter and more efficient. Examples include `for` loops (repeat a fixed number of times) and `while` loops (repeat as long as a condition remains true).

5. Function

A function is a reusable block of code designed to perform a specific task. Functions help organize code, reduce repetition, and improve readability. They can accept inputs called parameters and may return an output after processing the data.

6. Array / List

An array or list is a collection of multiple values stored in a single variable. Each element is accessed using an index number, allowing efficient storage and manipulation of related data.

7. Object and Class

A class is a blueprint used to create objects, while an object is an instance of a class. They are fundamental to object-oriented programming and allow programmers to model real-world entities using properties (attributes) and behaviours (methods).

8. Algorithm

An algorithm is a step-by-step set of instructions designed to solve a specific problem. Efficient algorithms help improve program performance and reduce execution time.

9. Error Handling

Error handling refers to techniques used to detect, manage, and respond to errors during program execution. It prevents programs from crashing and improves reliability using mechanisms such as exception handling.

10. Input and Output (I/O)

Input and output operations allow programs to receive data from users or external sources and display results. Input may come from keyboards, files, or sensors, while output may appear on screens, files, or other devices.