📘 MODULE 0: Introduction to Programming

(Foundations before learning Java or any programming language)

---

1. What is Programming?

Definition (Interview-ready)

Programming is the process of designing and writing a set of instructions that a computer can execute to solve a specific problem.

Explanation

A computer cannot think, reason, or make decisions on its own.

It strictly follows instructions provided to it. Programming provides a structured way to express **logic**, **decisions**, and **calculations** so that a computer can process input data and produce meaningful output.

Simple Example

Problem: Find the sum of two numbers.

Programming involves:

- Taking input values
- Applying logic (addition)
- Producing output

🔑 Trick to Remember

> **Programming = Logic + Instructions**

---

2. Why Do We Need Programming?

Programming is required to:

- Automate repetitive and manual tasks
- Solve problems efficiently and accurately
- Build software systems such as:
    - Web applications
    - Mobile applications
    - Operating systems
    - Databases
    - Artificial Intelligence systems

Interview POV

> Programming enables humans to communicate problem-solving logic to machines in a form they can execute.

🔑 Trick to Remember

> **No programming → no software**

---

3. Where is Programming Used?

Programming is used in almost every domain:

- **Web Development** – websites, APIs, cloud services
- **Mobile Applications** – Android, iOS
- **System Software** – operating systems, drivers
- **Data Science & AI** – data analysis, machine learning

- **Embedded Systems** – IoT, robotics
- **Enterprise Software** – banking, ERP systems

🔑 Trick to Remember

> **If it runs on electricity, programming is behind it**

---

4. How a Program Works (High-Level View)

Every program follows the same basic model:

`Input → Processing → Output`

Example

- Input: user enters two numbers
- Processing: program adds them
- Output: result is displayed

This model applies to **all programming languages**.

🔑 Trick to Remember

> **IPO = Input, Process, Output**

---

5. Types of Programming Paradigms

A **programming paradigm** defines the style and structure used to write programs.

5.1 Procedural Programming

Definition

Procedural programming organizes code into **procedures or functions** that operate on data.

Characteristics

- Step-by-step execution
- Data and functions are separate
- Top-down approach

Example Languages

- C
- Pascal

Interview POV

> Procedural programming focuses on executing instructions in a defined sequence using functions.

🔑 Trick to Remember

> **Procedure = Steps**

5.2 Functional Programming

Definition

Functional programming treats computation as the evaluation of **mathematical functions** without modifying shared data or state.

Characteristics

- Immutability
- No side effects
- Functions are first-class citizens

Example Languages

- Haskell
- Scala

- JavaScript (supports functional concepts)

Interview POV

> Functional programming emphasizes pure functions and avoids mutable state.

🔑 Trick to Remember

> **Function in → Result out (no change inside)**

5.3 Object-Oriented Programming (OOP)

Definition

Object-Oriented Programming organizes software around **objects**, which combine data and behavior.

Characteristics

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

Example Languages

- Java
- C++
- Python

Interview POV

> OOP models real-world entities using objects to improve modularity and code reuse.

🔑 Trick to Remember

> **Object = Data + Methods**

---

6. Static vs Dynamic Programming Languages

6.1 Static Typing

Definition

In statically typed languages, variable types are checked **at compile time**.

Characteristics

- Type must be declared
- Errors caught early
- Better performance and safety

Example

```
int x = 10;
```

Languages

- Java
- C++
- C

Interview POV

> Static typing improves reliability by detecting type errors during compilation.

🔑 Trick to Remember

> **Static = Safe before run**

6.2 Dynamic Typing

Definition

In dynamically typed languages, variable types are checked **at runtime**.

Characteristics

- No explicit type declaration
- More flexibility
- Runtime errors possible

Example

```
x = 10
x = "hello"
```

Languages

- Python
- JavaScript

Interview POV

> Dynamic typing offers flexibility but shifts error detection to runtime.

🔑 Trick to Remember

> **Dynamic = Decide while running**

---

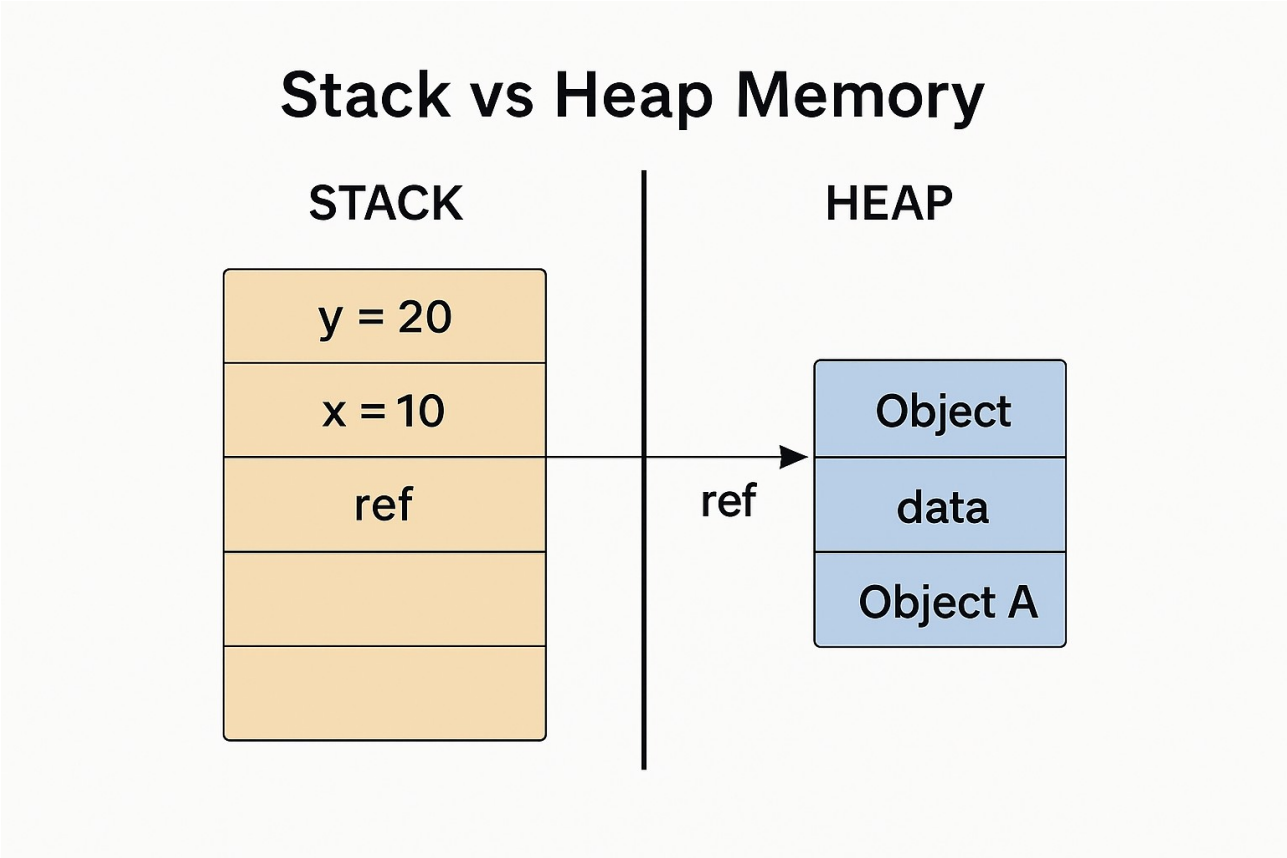
7. Memory Basics in Programming

7.1 Stack Memory

- Stores function calls and local variables
- Memory allocation is automatic
- Follows LIFO (Last In, First Out)
- Faster access

7.2 Heap Memory

- Stores objects and dynamically allocated data
- Shared across functions
- Managed by the runtime environment



Interview POV

> Stack memory manages execution flow, while heap memory stores objects.

🔑 Trick to Remember

> **Stack = Execution**
> **Heap = Objects**

---

8. Pass by Value vs Pass by Reference

8.1 Pass by Value

Definition

A copy of the variable’s value is passed to a function.

- Changes inside the function do not affect the original variable

Used in

- Primitive types in Java

8.2 Pass by Reference (Conceptual)

Definition

A reference to a memory location is passed.

- Changes affect the original object

Important Note (Java)

Java is **pass-by-value**, but when objects are passed, the **value of the reference** is passed.

Interview POV

> Java always uses pass-by-value, including object references.

🔑 Trick to Remember

> **Java never passes the object, only the reference value**

---

9. Garbage Collection (Memory Management)

Definition

Garbage Collection is the automatic process of identifying and freeing memory occupied by objects that are no longer in use.

Why It Is Needed

- Prevents memory leaks
- Reduces manual memory management

- Improves application stability

Example

If an object has no active references, it becomes eligible for garbage collection.

Interview POV

> Garbage collection automatically reclaims unused heap memory.

🔑 Trick to Remember

> **No reference → Garbage collected**

---

10. Key Takeaways (Quick Revision)

- Programming is structured problem-solving for computers
- Programs follow **Input → Processing → Output**
- Procedural, Functional, and OOP are major paradigms
- Static typing checks types at compile time
- Dynamic typing checks types at runtime
- Stack manages execution, heap stores objects
- Java provides automatic garbage collection
