



Concept of a Program & Programming Languages

From an idea in your head to instructions a machine can follow

Fundamentals of Programming • Unit 1

What Is a Program?



Definition

A program is a set of instructions written in a language a computer can understand, arranged in a precise sequence to perform a specific task.

- A finite, ordered sequence of steps
- Written using a programming language
- Solves one well-defined problem or task
- Executed by hardware via a CPU

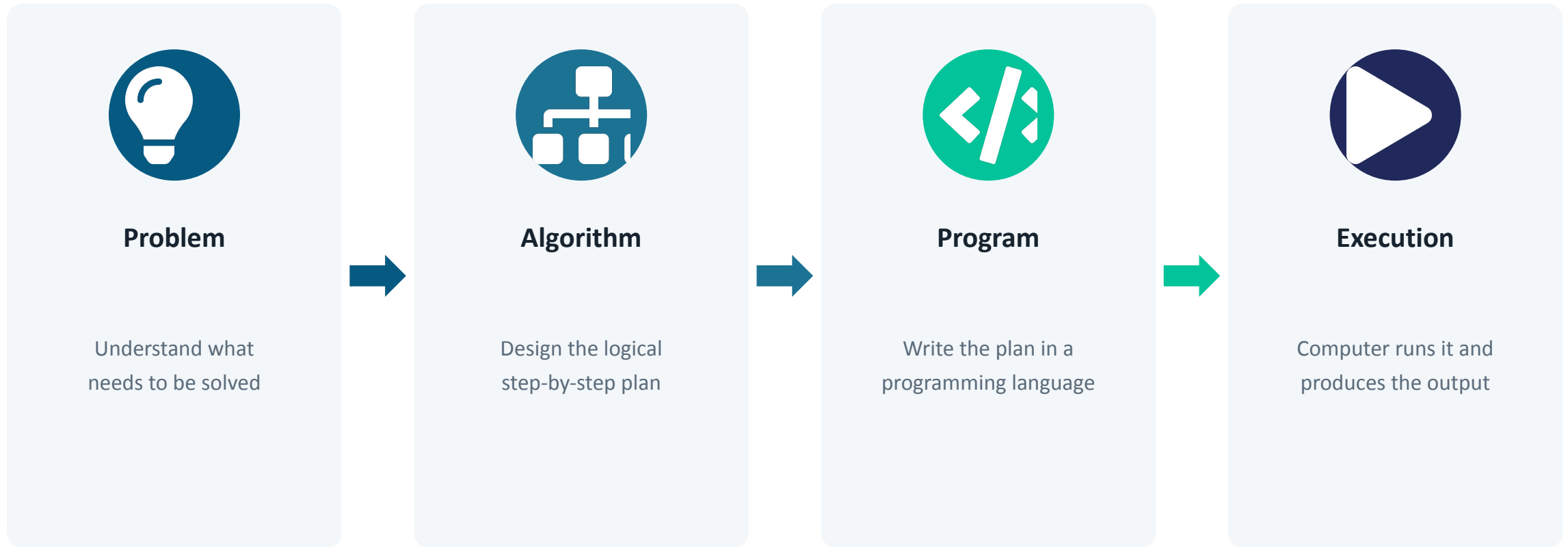
A Tiny Program, in Plain Words

- 1 start
- 2 ask the user for two numbers
- 3 add the numbers together
- 4 display the result
- 5 stop

This ordered list of steps is the essence of every program — from a five-line script to a full operating system.

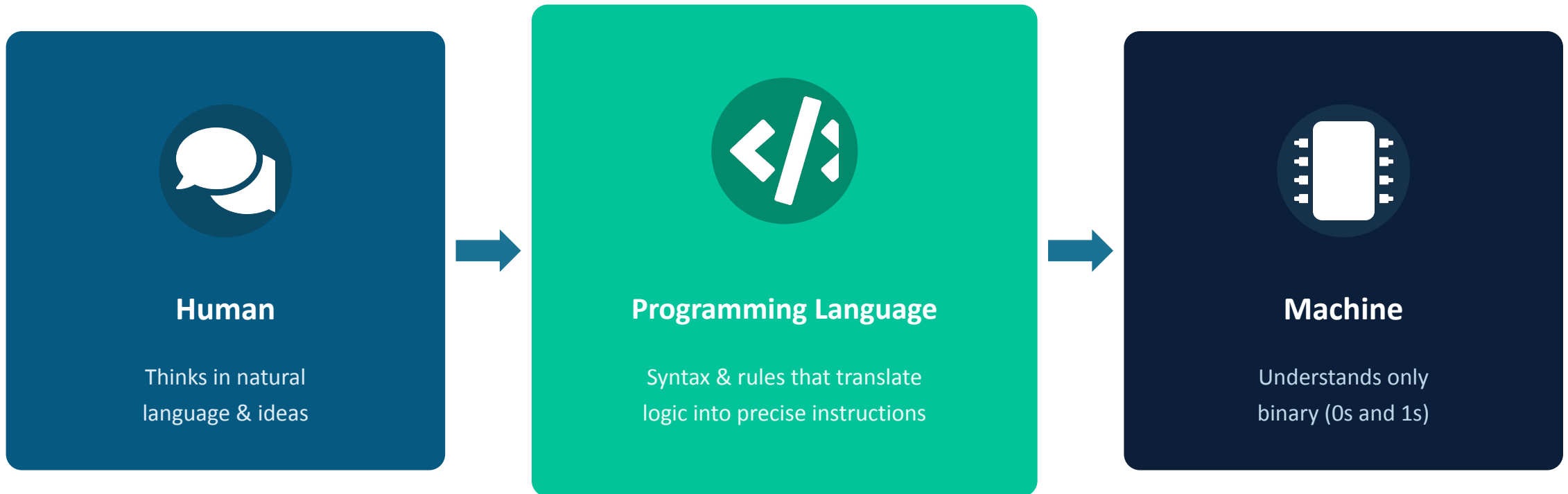
From Problem to Program

A program is built on an algorithm — a step-by-step plan for solving a problem



What Is a Programming Language?

A programming language is a formal, notational system of rules and symbols used to write instructions that a computer can execute — a bridge between human thought and machine action.



Levels of Programming Languages

Languages range from close to hardware to close to human thought

Machine Language

Raw binary (0s & 1s) — directly executed by the CPU, no translation needed.

Assembly Language

Mnemonic codes (MOV, ADD) mapped one-to-one to machine instructions.

High-Level Language

English-like syntax (C, Python, Java) — readable and portable across machines.

Very High-Level / 4GL

Declarative, close to human language (SQL) — states **what**, not **how**.

Higher levels trade a little execution speed for large gains in readability, portability, and developer speed.

Types of Programming Languages

Grouped by the programming paradigm — the style of thinking used to write solutions



Procedural

Step-by-step instructions grouped into procedures / functions.

e.g. C, Pascal, FORTRAN



Object-Oriented

Models data & behavior together as reusable "objects".

e.g. C++, Java, Python



Functional

Builds programs by composing pure, reusable functions.

e.g. Haskell, Lisp, Scala



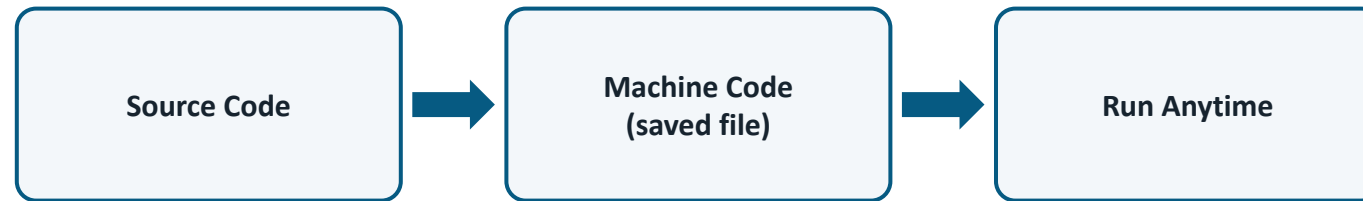
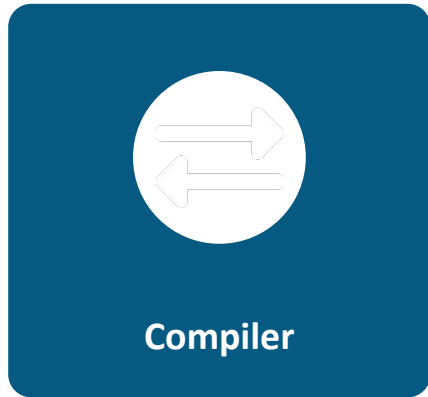
Scripting

Lightweight languages for automation and quick tasks.

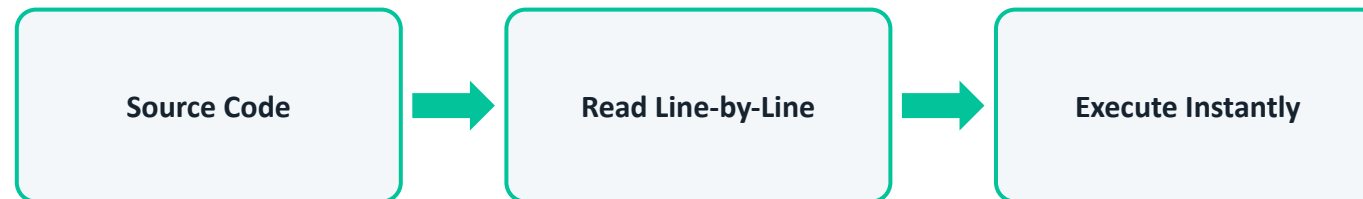
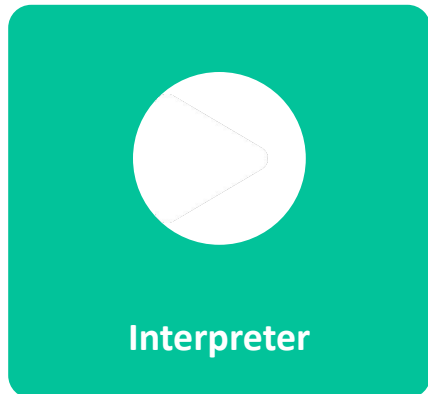
e.g. Python, JavaScript, Bash

Turning Code Into Action: Translators

The CPU only understands machine code — a translator bridges the gap



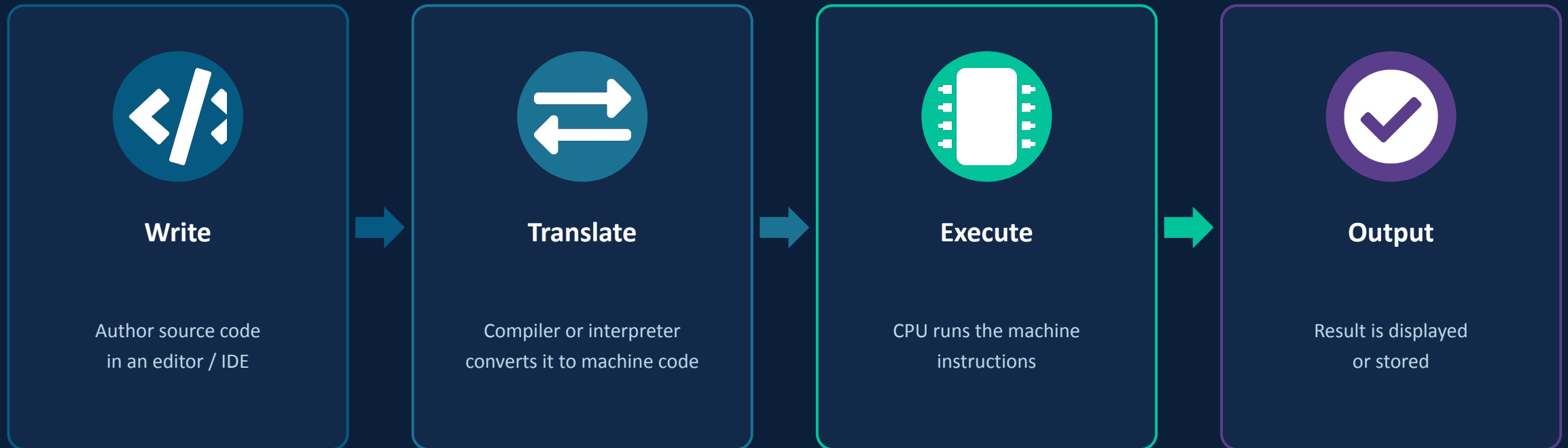
Compiler: translates the whole program once, then runs the fast machine-code file directly (e.g. C, C++).



Interpreter: translates and runs one line at a time, no separate file produced (e.g. Python, JavaScript).

The Program Execution Cycle

What happens between writing code and seeing a result on screen



A Few Languages You'll Meet

Different languages, different strengths — chosen based on the task at hand

C

Fast, low-level control.
Operating systems,
embedded.

C++

C plus OOP features.
Games, systems,
performance apps.

Python

Simple, readable
syntax.
AI/ML, data science,
scripting.

Java

"Write once, run
anywhere".
Enterprise & Android
apps.

JavaScript

Runs in the browser.
Web interactivity,
full-stack apps.



Key Takeaways

- **A program** — is an ordered set of instructions that solves a specific problem.
- **An algorithm** — is the logical plan behind a program, written before any code.
- **A programming language** — provides the formal syntax to express that plan for a machine.
- **Language levels** — range from binary machine code up to human-readable high-level code.
- **Compilers & interpreters** — translate source code into instructions the CPU can execute.