Introduction to Programming

Day 6
16/07/2025

C Language

ingeneral C used to Build System software

Machine Dependent

Compiled language

C Language doesnt have object oriented programming

In that case we have to use C++ for object oriented Programming.

Python

open source
↓
which leads to More upgradations
↓
Re usable libraryies

Interpreted language → Portable Language

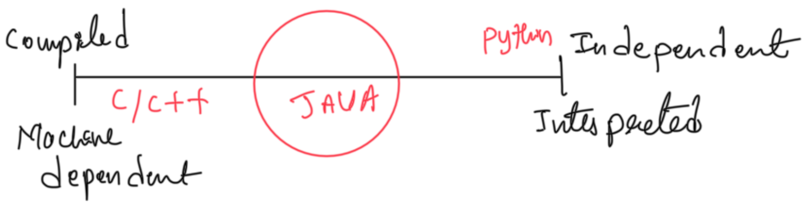
Machine Independent Language → which make python OS Independent

Python used in web applications front End BackEnd and widely Python re usable libraries are used for Machine Learning techniques.

Java

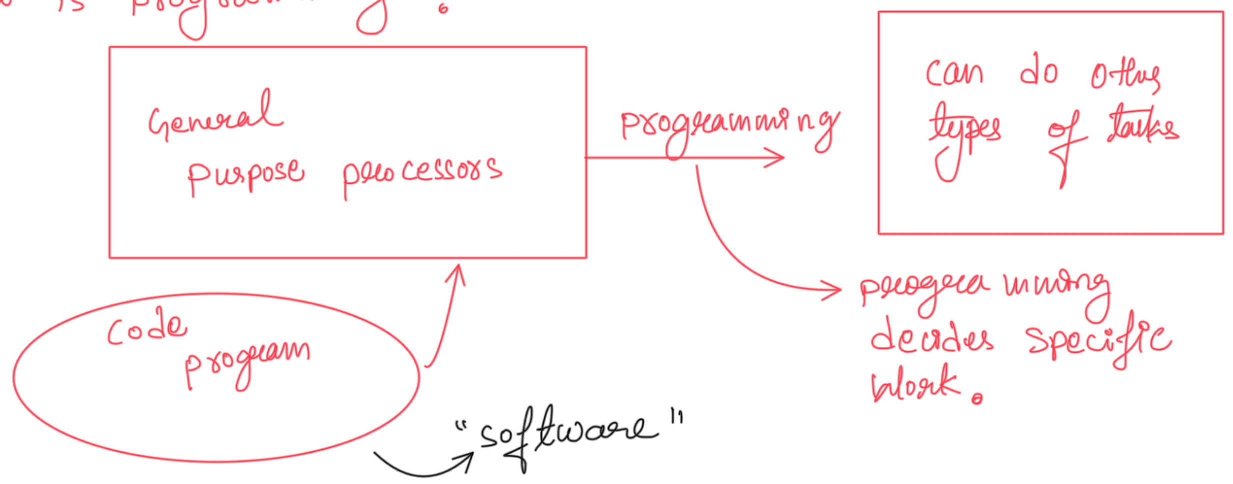
Java is compiled language
But Even though Java is compiled language
But it is a Machine Independent

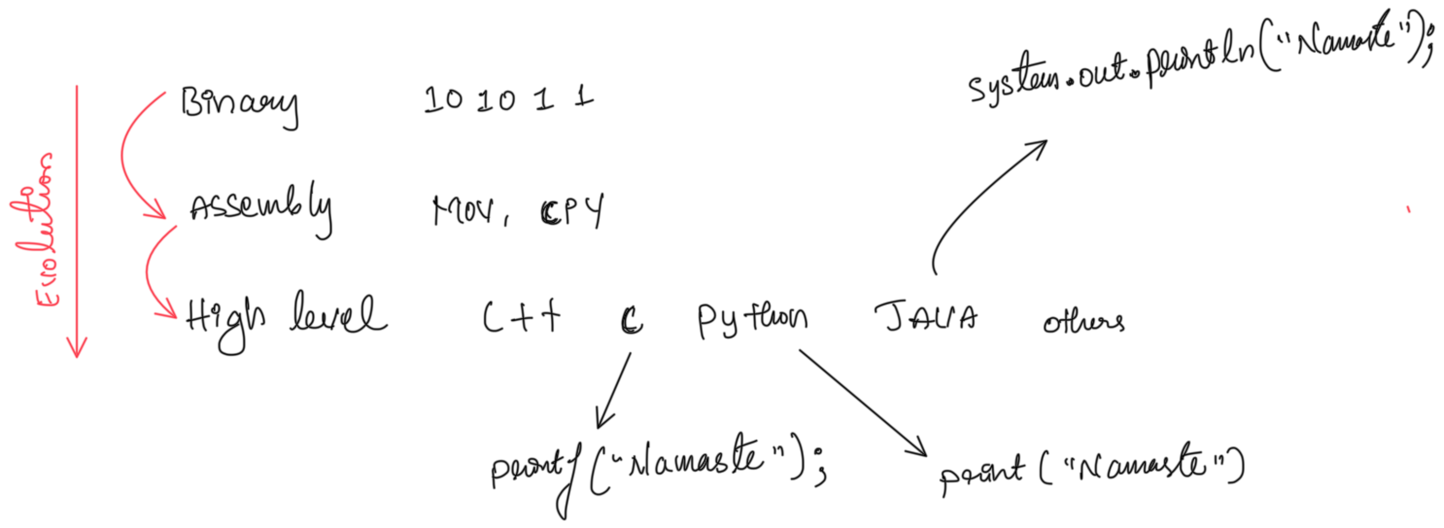
But it is a Machine Independent.



JAVA Has the Advantage of Compiled Languge and Independent language (JAVA is Compiled & Machine Independent)

What is programming ?

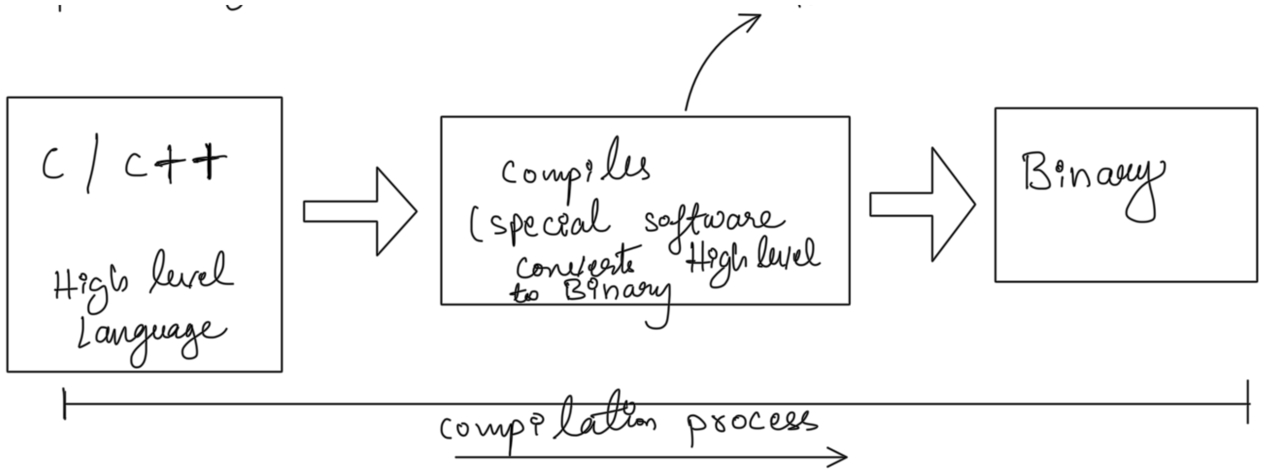

General purpose processors
programming
can do other types of tasks
programming decides specific work.
code program
"software"

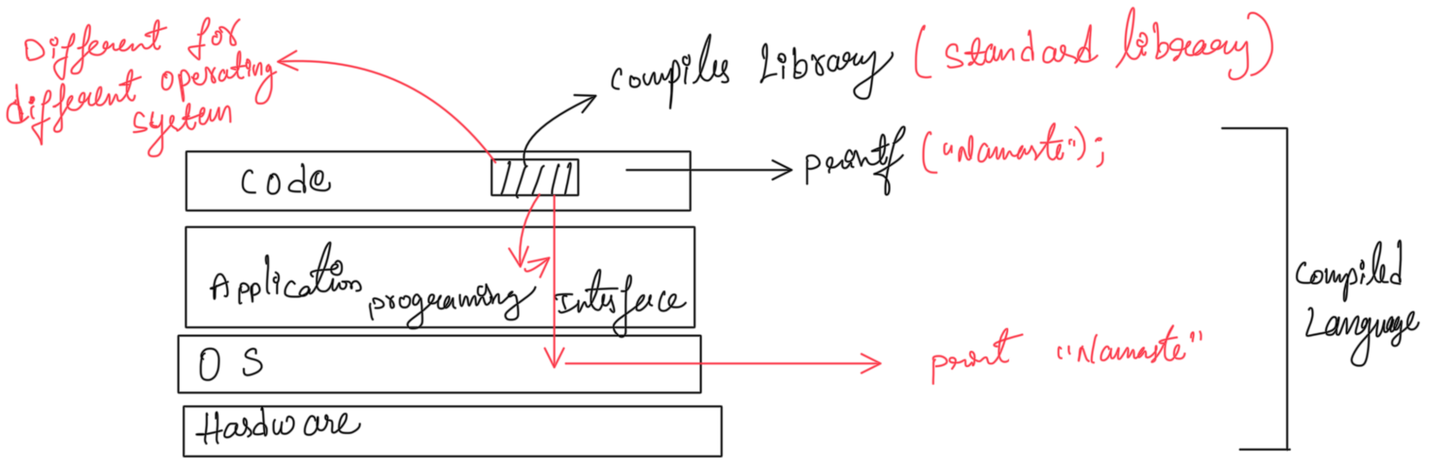


There is a little syntaxical difference in one programming Language to another.

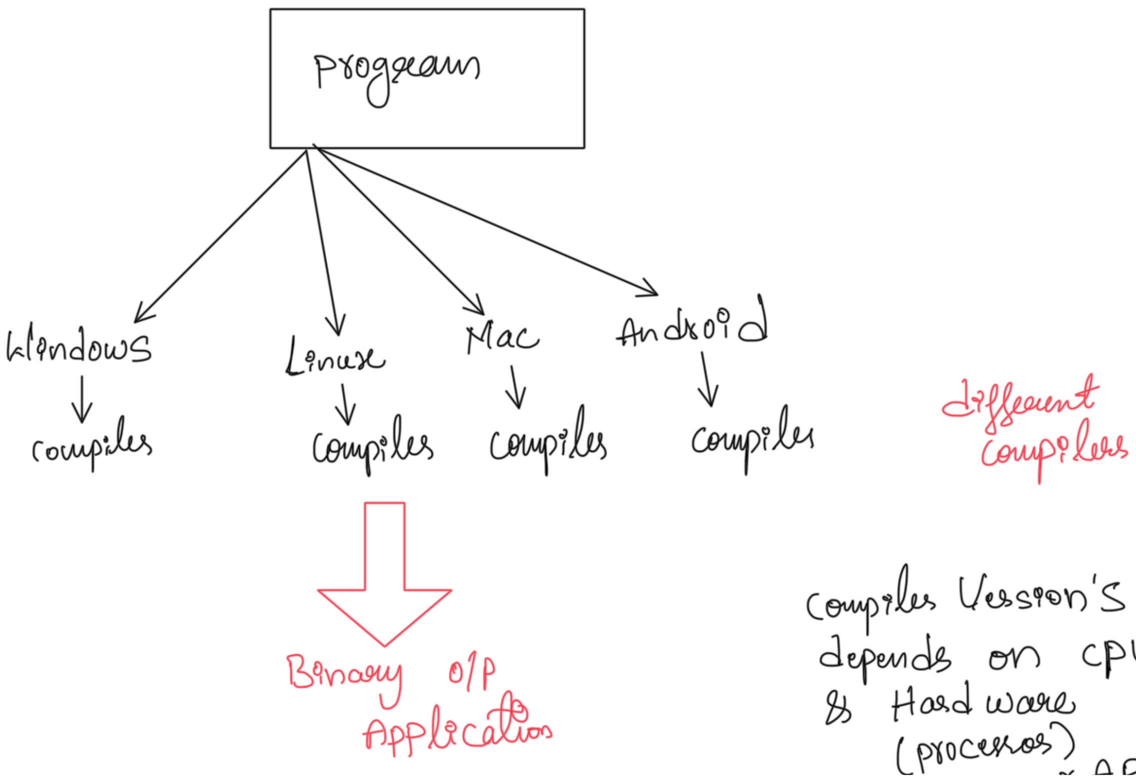
<u>Compiled Language</u>

Tool




Different for different operating system
compiler Library (standard library)
Code
printf ("Namaste");
Application programming Interface
O S
print "Namaste"
Hardware
Compiled Language

The compiler library react with OS API But the OS API is different for different Operating System So the different standard libraries used to Interact with different Operating System.



compiler Version's depends on CPU & Hardware (processor)

ARM

X86 or ARM

32 Bit or 64Bit

underline processor

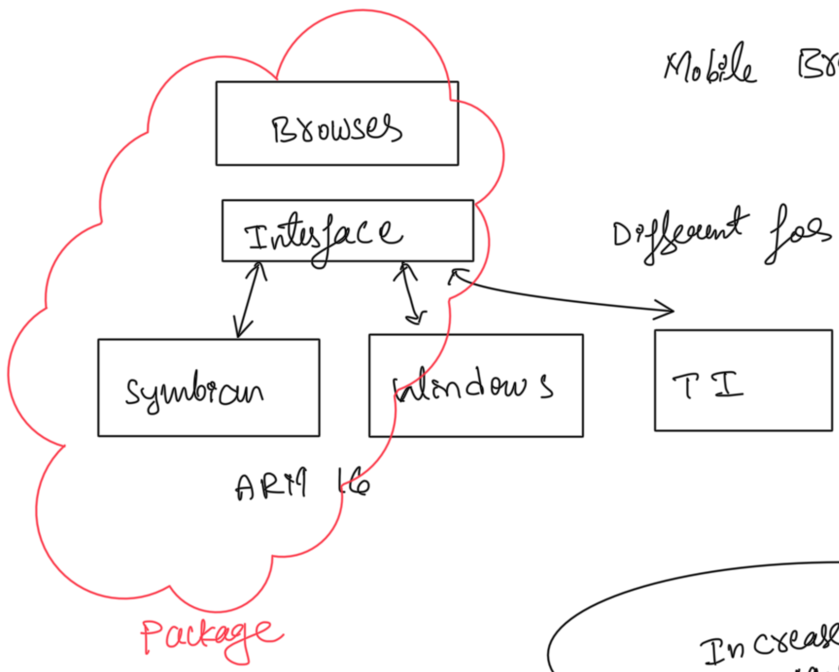
$$\begin{tikzcd}
 & X86 \arrow[dl] \arrow[dr] & & & ARM \arrow[dl] \arrow[dr] & \\
32\ Bit & & 64Bit & 64 & & 32\ Bit
\end{tikzcd}$$

This makes compiled languages are platform Dependent

In year 2005 2006

Mobile Browser

Different for different OS

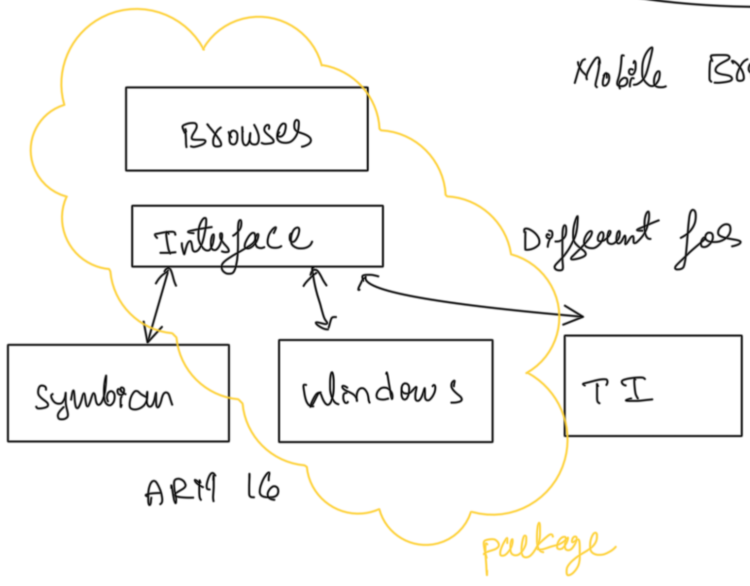


Increases
Maintainance
Bug fix cost

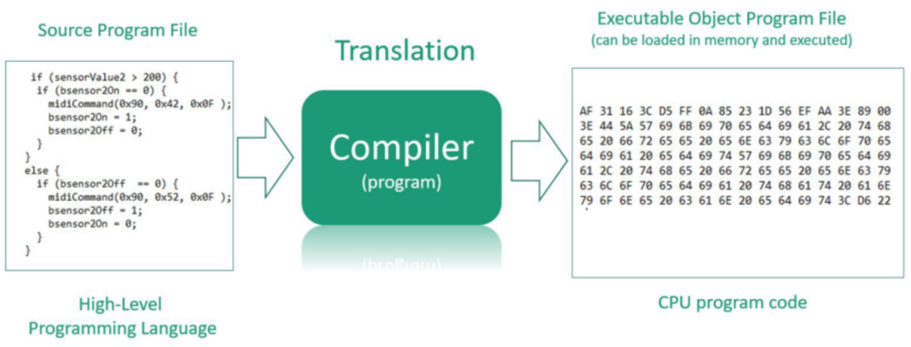
In year 2005 2006

Mobile Browser

Different for different OS



Different packages for Different Operating system needs to Be created and Maintained.



```
if (sensorValue2 > 200) {
  if (bsensor2On == 0) {
    midiCommand(0x90, 0x42, 0x0F );
    bsensor2On = 1;
    bsensor2Off = 0;
  }
}
else {
  if (bsensor2Off  == 0) {
    midiCommand(0x90, 0x52, 0x0F );
    bsensor2Off = 1;
    bsensor2On = 0;
  }
}
```

```
AF 31 16 3C D5 FF 0A 85 23 1D 56 EF AA 3E 89 00
3E 44 5A 57 69 6B 69 70 65 64 69 61 2C 20 74 68
65 20 66 72 65 65 20 65 6E 63 79 63 6C 6F 70 65
64 69 61 20 65 64 69 74 57 69 6B 69 70 65 64 69
61 2C 20 74 68 65 20 66 72 65 65 20 65 6E 63 79
63 6C 6F 70 65 64 69 61 20 74 68 61 74 20 61 6E
79 6F 6E 65 20 63 61 6E 20 65 64 69 74 3C D6 22
```

$$\underbrace{.c \xrightarrow{\text{creates}} .obj \xrightarrow{\text{creates}} .exe \text{ file}}_{\text{Intermediate step}}$$

C.exe

$$\begin{tikzcd} .c \text{ file} \arrow[d] \arrow[d, bend left] \\ .obj \arrow[d] \arrow[d, bend left] \\ .exe \end{tikzcd}$$

IN compiler language the compiler Installed at source

Python

$$\begin{tikzcd} & \text{Source } \boxed{\text{Program}} \arrow[rr, "\text{No changes}"'] & & \text{Destination } \boxed{\text{program}} \arrow[dlll] \arrow[dl] \arrow[d] \arrow[dr] & \\ \text{Window} & & \text{Linux} & \text{Mac} & \text{Android} \end{tikzcd}$$

x86
ARM }
16, 32, 64 Bit processor }

combinations to Build Interpreter

Interpreter software Introduced

Python executes code Line By

SRC

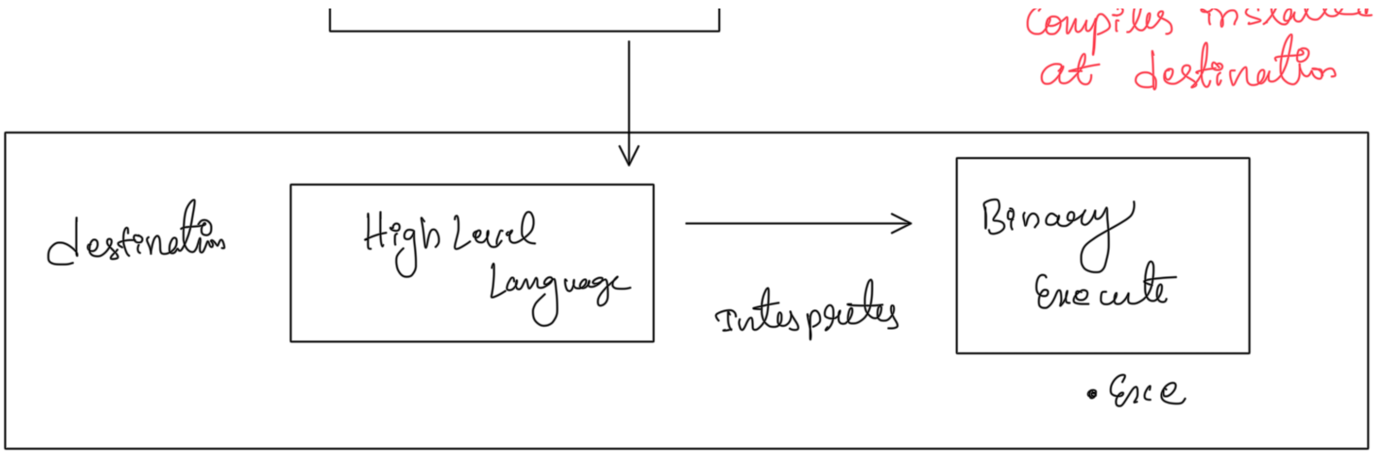
python.exe

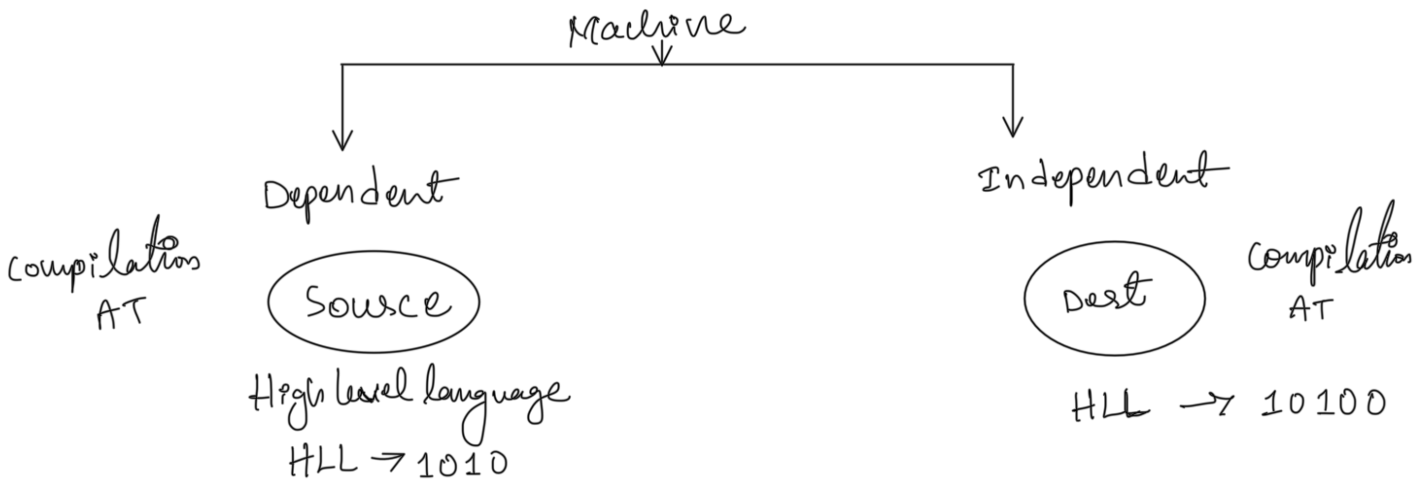
$\boxed{\text{python.py}}$

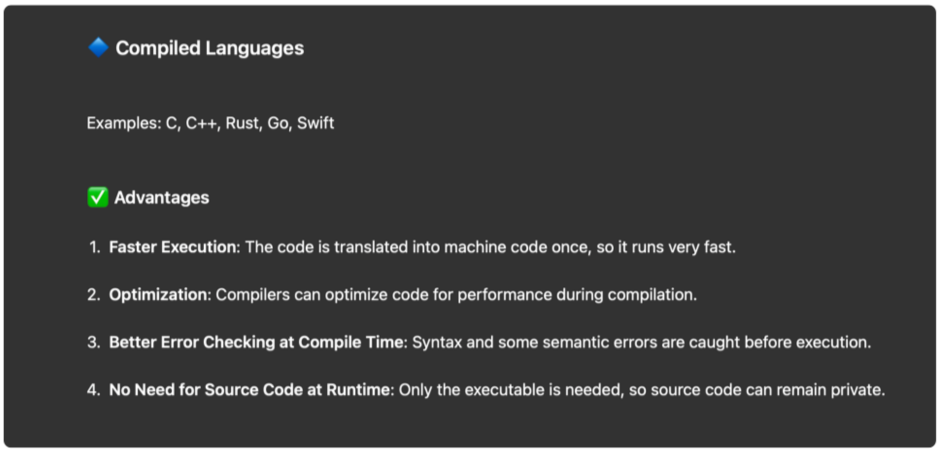
.py

In Interpreter Language the

Compiles installed
at destination





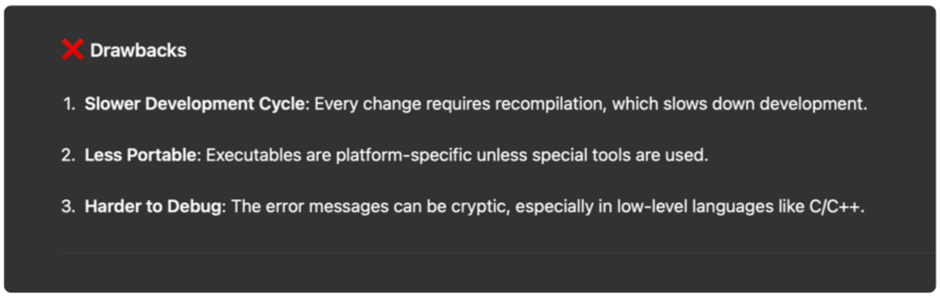

🔷 **Compiled Languages**

Examples: C, C++, Rust, Go, Swift

✅ **Advantages**

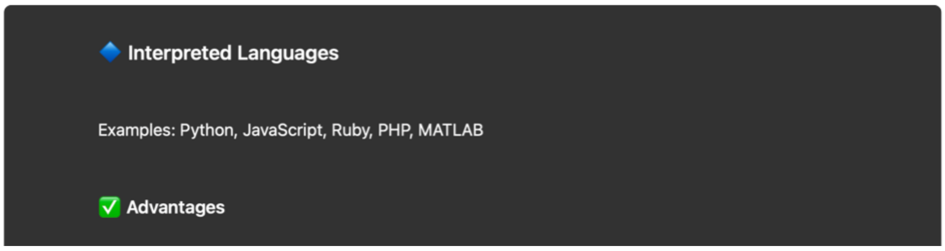
1. **Faster Execution**: The code is translated into machine code once, so it runs very fast.
2. **Optimization**: Compilers can optimize code for performance during compilation.
3. **Better Error Checking at Compile Time**: Syntax and some semantic errors are caught before execution.
4. **No Need for Source Code at Runtime**: Only the executable is needed, so source code can remain private.

pros & cons of
compiled Language


❌ **Drawbacks**

1. **Slower Development Cycle**: Every change requires recompilation, which slows down development.
2. **Less Portable**: Executables are platform-specific unless special tools are used.
3. **Harder to Debug**: The error messages can be cryptic, especially in low-level languages like C/C++.

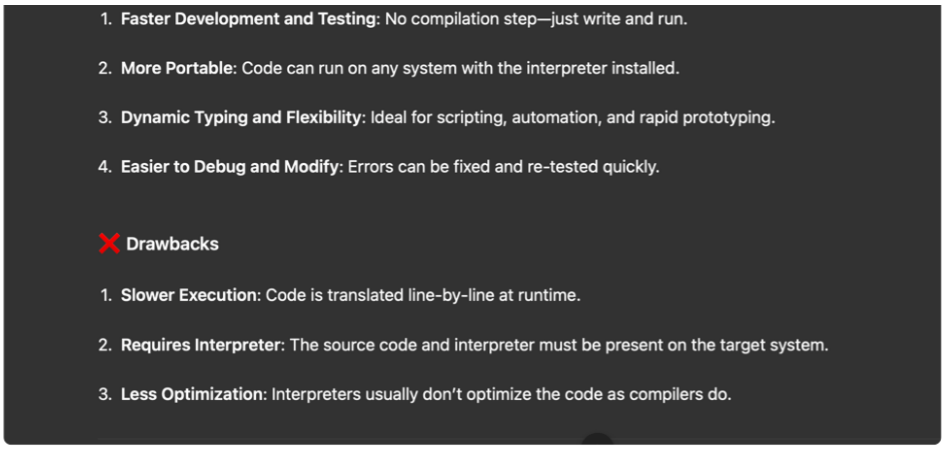
Takes more
Memory


🔷 **Interpreted Languages**

Examples: Python, JavaScript, Ruby, PHP, MATLAB

✅ **Advantages**

pros & cons of
Interpreter language



1. **Faster Development and Testing:** No compilation step—just write and run.
2. **More Portable:** Code can run on any system with the interpreter installed.
3. **Dynamic Typing and Flexibility:** Ideal for scripting, automation, and rapid prototyping.
4. **Easier to Debug and Modify:** Errors can be fixed and re-tested quickly.

❌ **Drawbacks**

1. **Slower Execution:** Code is translated line-by-line at runtime.
2. **Requires Interpreter:** The source code and interpreter must be present on the target system.
3. **Less Optimization:** Interpreters usually don't optimize the code as compilers do.

In Interpreter languages Source code is visible to the client if project file opened.

Every Language have its own Advantags & disadvantages.

Java

Java Compilation

Java is a strictly Typed object oriented, high level platform-independent programming language that supports Both compiled & interpreted Execution.

$$\begin{tikzcd}
\text{program} \arrow[d] \\
\text{HelloWorld.java} \arrow[d, "\text{(javac)}"] \\
\text{Java Compiler} \arrow[d] \\
\text{HelloWorld.class (Bytecode)} \arrow[d] \\
\downarrow \text{ send to destination } \downarrow \arrow[d] \\
\text{[JVM on Target Machine]} \arrow[d, "\text{(Interpreter / JIT)}"] \\
\text{[Native Machine code]} \arrow[d] \\
\text{Program Runs}
\end{tikzcd}$$

Introduction to Coding and Evolution of Programming Languages

Introduction

The most exciting part of our skill development program. In earlier sessions, we covered the software industry, roles, and basics like operating systems and computers. Now, let's get started with how to **code**, and understand the **evolution of programming languages**.

Why Students Struggle with Coding

Many students complete courses and projects but still don't feel confident in their coding skills. After interacting with 50+ students and mentoring 8–10 interns, we found that:

- Many focus too much on one language.
- Courses don't teach **why** something works — just **how** to write it.
- Hands-on practice is missing.

Our Approach

- Learn **multiple languages** (C, Python, Java) side by side.
- Focus on **concepts** that are shared across languages.
- Practice **hands-on coding and environment setup**.

Introduction to Programming Languages

Programming languages allow us to write instructions that a computer can understand. Here's a simplified hierarchy of how they evolved:

1. **Binary (Machine Language)**: 1s and 0s
2. **Assembly Language**: Symbolic representation (e.g., `MOV AX, BX`)
3. **High-Level Languages**: Human-readable (e.g., C, Python, Java)

Learning 3 Key Languages

1. C Language

- **Used for**: System software (e.g., database servers, Memcached).
- **Compiled Language**: Converts to binary before execution.
- **Machine Dependent**: Runs only on the platform it was compiled for.
- **No Object-Oriented Support**: For that, use C++.

```c
#include <stdio.h>
int main() {
    printf("Namaste from India\n");
    return 0;
}
```

2. Python

- **Used for**: Frontend, backend, data science, AI, automation.
- **Interpreted Language**: Source code is read and executed directly.
- **Machine Independent / Portable**: Works across platforms via interpreter.
- **Open Source & Upgradable**: Regular updates and community libraries.

```python
print("Namaste from Bharat")
```

3. Java

- **Used for**: Backend, enterprise apps, Android.
- **Compiled + Portable**: Compiled into bytecode, then run on Java Virtual Machine (JVM).
- **Hybrid**: Combines advantages of compiled and interpreted models.

```java
class Hello {
    public static void main(String[] args) {
        System.out.println("Namaste from Java");
    }
}
```

Compiled vs Interpreted Languages

Feature	C (Compiled)	Python (Interpreted)	Java (Hybrid)
Speed	Fast	Slower	Moderate
Portability	Low	High	High (via JVM)
Requires Compilation	Yes	No	Yes (to bytecode)
Output File Type	.exe/.out	N/A	.class

How Compilation Works (C Example)

1. You write code in a text file (`.c`).

2. Compiler converts it into an **object file** (`.obj`).
3. Linker creates an **executable file** (`.exe`).

Demo:

```
cl namaste.c  # Compiles using Microsoft C Compiler
namaste.exe   # Run the compiled program
```

Even a simple `printf()` creates an executable with hundreds of lines of binary code. This is because standard libraries and OS-level instructions are included during compilation.

Platform Dependence

Programs compiled on one OS (e.g., Windows) cannot run on another (e.g., Android or Linux) due to:

- **Different OS APIs**
- **Different CPU architectures (x86 vs ARM)**
- **32-bit vs 64-bit support**

Analogy:

Trying to run a Windows `.exe` file on an Android phone is like inserting a square peg into a round hole — it just won't fit.

To solve this, developers must use platform-specific compilers for each OS and processor type.

Code Portability: Challenges and Solutions

Problem:

- Platform-dependent software needs **16+ compiler versions** for various combinations of OS and CPU architectures.

Example:

- Android, Windows, Linux, Mac (each with 32/64-bit and x86/ARM).
- Each needs its own binary package.

Industry Solution:

- Use **interface layers** or **abstractions** that isolate OS-specific code.
- Example: In 2005, mobile browser code used one interface layer to support Symbian, Windows, and other platforms.

Binary Files & Compilation Output

Even simple programs generate: - `.obj` (object files) - `.exe` (executables)

These are **binary files**, unreadable by humans but understood by CPUs.

Example:

Open a compiled `.exe` and it shows garbage text. But this binary contains: - Code - Library references - Metadata for the OS

Python: Interpreted Language

Python avoids many headaches of C: - No need to compile for every OS. - Just install a **Python interpreter** for your platform. - Write once, run anywhere (with Python installed).

```
# Example
python hello.py
```

Python interpreters are available for: - Windows (x86/ARM, 32/64-bit) - Linux, Mac, Android, etc.

Summary

- Learn coding concepts, not just syntax.
- Study C, Python, and Java in parallel for a broader understanding.
- Understand compilation vs interpretation.
- Know how OS and CPU affect portability.
- Python simplifies portability using interpreters.

By understanding these fundamentals, you'll build a strong foundation for advanced programming topics.