Scripting language

scripting language meant:

- A language used to “script” or automate tasks,
- Usually interpreted (not compiled),
- Often embedded inside another system (like shell scripts, game scripts, browser scripts).

 Java is NOT a scripting language.

 It is a **compiled, strongly-typed, object-oriented programming language**.

 You write .java code → compile to **bytecode** → run on the **JVM (Java Virtual Machine)**.

- **JavaScript is a scripting language**.
- Originally created for scripting inside web browsers.
- Interpreted (though modern engines JIT-compile it for speed).
- Today, it’s also used outside browsers (Node.js) for full-stack development.

Originally:

- JavaScript was created (1995) as a **scripting language** for browsers → adding interactivity to web pages.

Now:

- With **Node.js**, JavaScript runs outside browsers → backend servers, APIs, automation scripts.
- With frameworks, it powers **web apps (React, Angular, Vue), mobile apps (React Native, Ionic), desktop apps (Electron), AI/ML (TensorFlow.js), even IoT**.

Because it’s used in **many domains**, not just one, it qualifies as **general-purpose**.

- **TypeScript is NOT a scripting language.**
- It is a **programming language** that is a **superset of JavaScript**.

- It adds **static typing, interfaces, enums, generics, and advanced tooling** on top of JavaScript.
- TypeScript code (.ts) must be **compiled (transpiled)** into JavaScript (.js) before running in browsers or Node.js.

TypeScript doesn't run directly anywhere — the TypeScript compiler (tsc) must turn it into JavaScript first.

 Python started as a scripting language (easy automation, small programs).

 Interpreted, dynamically typed.

 But now, it's much more: used for AI/ML, web apps, automation, etc.

 So Python is both a **scripting language** *and* a **general-purpose programming language**.

A **general-purpose programming language** is a language designed to build *all kinds of software*, not restricted to a single domain.

- You can use it for **web apps, desktop apps, mobile apps, AI, system software, games, scripts, etc.**
- Examples: **Python, Java, C, C++, JavaScript, Go, Rust**.

🔹 **Opposite: Domain-Specific Language (DSL)**

A **domain-specific language** is designed for a *specific kind of task*.

- Examples:
  - **SQL** → database queries.
  - **HTML** → structuring web pages.
  - **MATLAB** → mathematical & engineering computations.
  - **VHDL/Verilog** → hardware design.

Simple Example:

- If you pick **SQL**, you can't build a game or an AI model → it's domain-specific.
- If you pick **Python**, you can build a game, AI model, website, automation script, or data analysis → it's general-purpose.

Anaconda python

https://www.youtube.com/watch?v=GfH4QL4VqJ0

Jupyter Notebook or Anaconda: ML

Modern AI & Deep Learning Era (2010s – now)

- **Python** → By far the most widely used language for AI/ML today because of:
    - Huge ecosystem: TensorFlow, PyTorch, scikit-learn, NumPy, Pandas, etc.
    - Easy syntax → fast prototyping + research.
    - Integration with C/C++ backends for performance.
- **C++** → Still crucial in high-performance ML (deep learning frameworks' core is in C++).
- **Julia** → Gaining popularity in scientific computing and ML (fast like C, but easy like Python).
- **JavaScript (Node.js, TensorFlow.js)** → For ML in the browser/web apps.
- **Rust & Go** → Emerging for efficiency, but still niche compared to Python.

https://www.blender.org/

TIOBE programming community index

Dropbox uses python