Differences between programming languages

Here are some common differences between programming languages:

1. **Purpose & Usage**
    - Some languages are general-purpose (e.g., **Python, Java**), while others are domain-specific (e.g., **SQL for databases, MATLAB for scientific computing**).
    - **Web development** languages include JavaScript, HTML, and CSS.
    - **System programming** uses languages like C and Rust.
2. **Programming Paradigm**
    - **Procedural languages** (e.g., **C, Pascal**) follow a step-by-step approach.
    - **Object-oriented languages** (e.g., **Java, C++**) use classes and objects.
    - **Functional languages** (e.g., **Haskell, Lisp**) focus on immutability and pure functions.
3. **Level of Abstraction**
    - **Low-level languages** (e.g., **Assembly, C**) provide direct hardware access.
    - **High-level languages** (e.g., **Python, JavaScript**) are easier to read and write but abstract away hardware details.
4. **Compiled vs. Interpreted**
    - **Compiled languages** (e.g., **C, C++**) convert code to machine language before execution.
    - **Interpreted languages** (e.g., **Python, JavaScript**) execute line by line at runtime.
    - Some languages (e.g., **Java**) use a **hybrid model** with a compiler and an interpreter (JVM).
5. **Syntax and Readability**
    - **Python** has a simple, readable syntax.

- **Perl** and **C++** have more complex syntax.

6. **Performance**
   - **C, C++**, and **Rust** offer high performance due to direct hardware access.
   - **Python** and **JavaScript** trade performance for ease of use.
7. **Memory Management**
   - **Manual memory management** (e.g., **C, C++**) requires developers to allocate and free memory.
   - **Garbage-collected languages** (e.g., **Java, Python**) handle memory automatically.

Feature	C	Python	Java	JavaScript	C++	Rust	SQL
Type	Compiled	Interpreted	Compiled + JVM	Interpreted	Compiled	Compiled	Query Language
Paradigm	Procedural	Multi-paradigm	OOP	Functional + OOP	Multi-paradigm	Multi-paradigm	Declarative
Usage	System programming, Embedded	AI, Data Science, Web	Enterprise apps, Mobile (Android)	Web Development	Game Development, System Apps	System programming, Safe Concurrency	Databases
Performance	High	Moderate	Moderate	Moderate	High	High	High (for queries)

Memory Management	Manual (malloc/free)	Automatic (Garbage Collected)	Automatic (Garbage Collected)	Automatic	Manual or Smart Pointers	Manual + Ownership Model	Managed by DB Engine
Syntax Complexity	Moderate	Easy	Moderate	Easy	Complex	Moderate	Simple (SQL Queries)
Portability	High	High	Very High	Very High	High	Moderate	High
Compilation	Yes (gcc, clang)	No (Interpreted)	Yes (JVM Bytecode)	No (Interpreted)	Yes (g++, MSVC)	Yes (rustc)	No (Executed in DB)
Garbage Collection	No	Yes	Yes	Yes	No	No	No
Best For	Performance-critical applications	AI, automation, scripting	Large-scale enterprise apps	Web development	Game engines, High-performance software	Memory-safe system programming	Managing relational databases