# SCRIPTING LANGUAGES

# SCRIPTING LANGUAGES

- Scripting languages are programming languages designed to automate tasks, manipulate data, and enable dynamic behavior in applications. They are often interpreted rather than compiled, making them ideal for rapid development and platform-independent execution.

Key Characteristics of Scripting Languages

1. Interpreted Execution: Code is executed directly, without a prior compilation step.
2. Ease of Use: Simpler syntax compared to traditional programming languages.
3. Dynamic Typing: Variables can hold different data types without strict declarations.
4. High-Level Operations: Support for complex operations like string manipulation, file handling, and network communication.
5. Embeddable: Can be embedded into larger applications to provide customization and flexibility.
6. Rapid Development: Ideal for prototyping and automating repetitive tasks.

# TYPES OF SCRIPTING LANGUAGES

1. Client-Side Scripting Languages

Executed in the user's browser to provide interactive user interfaces and dynamic web content.

Examples:

- JavaScript: Most popular for dynamic web applications.
- VBScript: Microsoft's scripting language for web development (deprecated).
- TypeScript: A superset of JavaScript with static typing.

## 2. Server-Side Scripting Languages

A server-side language is a programming language that runs on the web server.  Executed on the server to generate dynamic web pages and manage backend processes.

Examples:

- PHP: Popular for web development.
- Python: Versatile and often used with frameworks like Django or Flask.
- Ruby: Used in frameworks like Ruby on Rails.
- Perl: Known for text processing and system administration tasks.

- Advantages of Scripting Languages

1. Flexibility: Can be used for a variety of tasks.
2. Rapid Development: Faster to write and test.
3. Cross-Platform: Most are platform-independent.
4. Integration: Easy to embed and extend applications.

Disadvantages of Scripting Languages

1. Performance: Slower than compiled languages due to runtime interpretation.
2. Debugging: May be harder to debug due to dynamic typing.
3. Security: More vulnerable if not properly secured (e.g., server-side injection attacks).