SHELL

Common types of shell:

1. **CMD (Command Prompt)**

- Operating System**:** Windows
- Use Cases**:** Basic file management, system configuration, running batch scripts, and interacting with the system's internal command-line tools.
- Control/Power:
  - Limited**:** While CMD is simple and fast for basic tasks, it does not offer much control or flexibility when compared to more advanced shells.
  - No scripting support (compared to Bash or PowerShell).
  - CMD does not support advanced features like piping or redirection in the same way more modern shells do.

2. **Bash**

- Operating System**:** Linux, macOS, Windows (via WSL)
- Use Cases**:** File system management, scripting, system administration, automation, and software development.
- Control/Power:
  - Very Powerful**:** Bash is highly flexible, with powerful scripting capabilities (loops, conditions, functions, etc.) and is widely used for managing and automating tasks on Linux-based systems.
  - Advanced Features**:** Supports piping, redirection, background processes, variable manipulation, and command substitution.
  - Bash also allows users to create complex scripts, making it ideal for automation.

3. **PowerShell**

- Operating System**:** Windows (cross-platform versions available for Linux and macOS)
- Use Cases**:** System administration, automation, file system management, and interacting with Windows-specific features (e.g., registry, WMI).
- Control/Power:
  - Extremely Powerful: PowerShell is a task automation and configuration management framework, which includes its own scripting language designed for managing and automating administrative tasks.

- Objects Instead of Text: One of PowerShell's biggest strengths is its ability to work with objects rather than plain text, making it incredibly versatile for interacting with system resources, APIs, and complex data.
- PowerShell integrates tightly with Windows and provides rich access to .NET libraries and Windows Management Instrumentation (WMI), giving users much greater control over system configuration.

4. **Anaconda Prompt**

- Operating System: Windows, Linux, macOS
- Use Cases: Primarily used for managing Python environments, installing packages via conda, and running Python-related tasks.
- Control/Power:
    - Specialised Control**:** The Anaconda Prompt itself is a wrapper around CMD or Bash, depending on the system. Its main focus is Python package management and environment control via the conda package manager.
    - Although it is primarily for managing Python environments, Anaconda Prompt offers limited access to the broader system and is not intended for full system administration tasks.

Feature/Aspect	CMD	Bash	Powershell	Anaconda Prompt
Primary Use	Basic file management, scripts	File system management, automation	System administration, automation	Python environment management
Script Support	Limited	Full scripting support	Full scripting support	Limited (focused on Python)
Syntax	Simple, basic	Unix-like syntax (POSIX)	Object-oriented, similar to .NET	Bash-like with conda commands
Control/Power	Low	High (with rich scripting)	Very high (deep system access)	Limited (focused on Python)
Platform Support	Windows Only	Linux, macOS, Windows (WSL)	Windows, Linux, macOS	Cross-platform
Notable Features	Basic commands, no scripting	Powerful scripting, pipelines, redirection	Object-oriented, deep system integration	Conda package management