**-:Cmdlets Commands basic:-**

**-:PowerShell Advanced Briefing:-**

**1. Introducing to Cmdlets :-**

**What is a Cmdlet?**

A Cmdlet is a lightweight, built-in PowerShell command used to perform specific tasks.

* Pronounced as **"command-let"**
* Always written in the format:

Verb-Noun

* E.g., Get-Process, Set-Location, New-Item

**Key Properties of Cmdlets:**

| **Feature** | **Description** |
| --- | --- |
| Built-in | Part of PowerShell, not external |
| Verb-Noun | Naming convention |
| Pipeline Aware | Can pass and receive data from pipelines |
| Object-Oriented | Works with .NET objects, not just plain text |

**Hands-On Practice:**

1. See all available Cmdlets:

Get-Command

1. Search for all Cmdlets starting with Get:

Get-Command -Verb Get

1. Update Help files (run as Admin):

Update-Help

**Real-World Use Case:**

**Task:** List all running services

Get-Service | Where-Object { $\_.Status -eq "Running" }

**Task:** Stop a specific service

Stop-Service -Name "Spooler"

**Key Cmdlet Examples to Remember:**

| **Cmdlet** | **Purpose** |
| --- | --- |
| Get-Process | Lists running processes |
| Stop-Process | Kills a process |
| Get-Service | Lists services |
| Set-ExecutionPolicy | Changes script policy |
| Get-Help | Displays documentation |
| New-Item | Creates files/folders |

**Note :-**

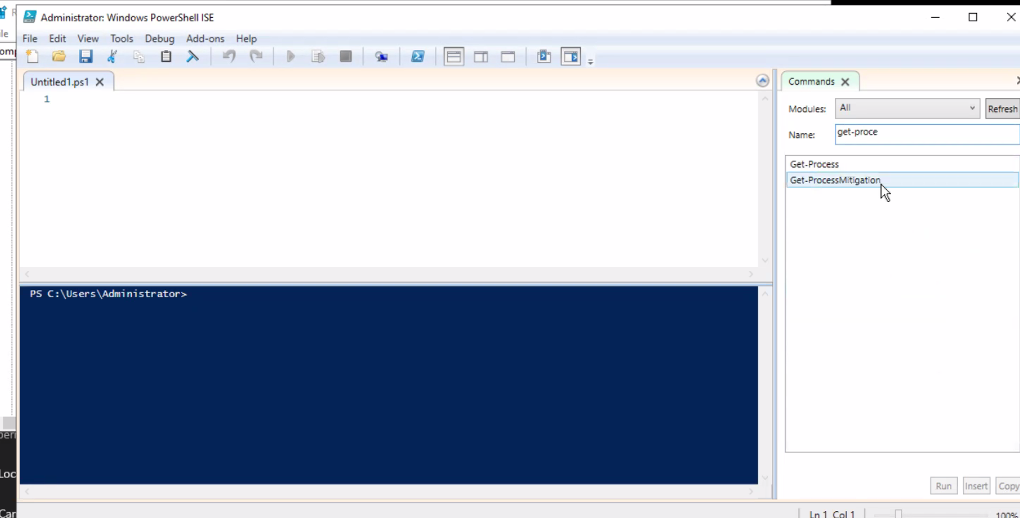
* **PowerShell Cmdlets return objects, not plain text.**
* **Aliased Cmdlets (e.g., ls = Get-ChildItem)**
* **Cmdlets are not functions, they are compiled code from .NET.**

**2. The PowerShell Pipeline (|) :-**

The pipeline (|) in PowerShell passes the output of one command into the input of another.

It’s like connecting building blocks to form a chain of tasks.

like this: Command1 | Command2 | Command3.

Each command gets the output of the previous command as its input.

**Example 1:** List all services and filter for running ones

* Get-Service | Where-Object { $\_.Status -eq "Running" }

Here:

* Get-Service lists all services.
* Where-Object filters based on the Status property.

**Example 2:** Get top 5 processes by memory usage

* Get-Process | Sort-Object -Property WS -Descending | Select-Object -First 5

**Step-by-Step Explanation**

**Get-Process**

* This **cmdlet** retrieves a list of all running processes on your system.
* Each process returned has properties like:
  + Name – process name (e.g., chrome, notepad)
  + Id – process ID
  + CPU – CPU time used
  + WS – **Working Set memory** (RAM currently used in bytes)

…and many more.

Example:

* Get-Process

will show something like:

scss

Handles NPM(K) PM(K) WS(K) CPU(s) Id SI ProcessName

1024 58 56732 120000 14.53 1204 1 chrome

321 24 15000 25000 1.32 4540 1 notepad

**| (Pipeline)**

* The **pipe operator** sends the output of one command as input to the next.
* Here, the list of processes from Get-Process is sent to the next cmdlet.

**Sort-Object -Property WS -Descending**

* Sort-Object sorts the list **based on a property** of each object.
* -Property WS means sort by the **Working Set memory**.
* -Descending means show the **largest memory usage first**.
* Without -Descending, it would sort smallest to largest.

Example:

* Get-Process | Sort-Object -Property WS -Descending

will list processes **starting with the ones using the most RAM**.

**Select-Object -First 5**

* This picks only the **first 5** objects from the sorted list.
* Since we sorted by memory usage in descending order, this means:  
  **Top 5 memory-hogging processes.**

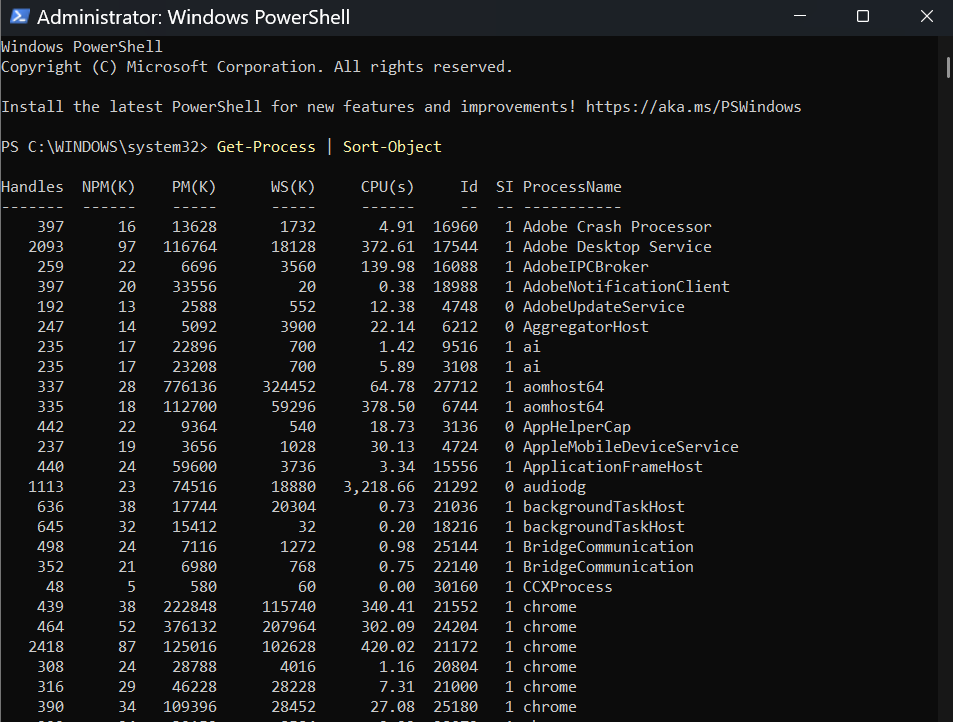
**Putting It All Together**

our command:

* Get-Process | Sort-Object -Property WS -Descending | Select-Object -First 5

**Translation in plain English:**

“Show me the top 5 running processes that are using the most RAM on my computer.”



**Example 3:** Show files larger than 100MB in C:\

* Get-ChildItem -Path C:\ -Recurse -File -ErrorAction SilentlyContinue |

Where-Object { $\_.Length -gt 100MB }

**Step-by-Step Explanation**

**1.Get-ChildItem -Path C:\**

* This **cmdlet** lists items (files and folders) in the given path.
* -Path C:\ means start from the **root of the C drive**.

**2. -Recurse**

* This tells PowerShell to **search inside all subfolders** of the given path.
* Without -Recurse, it would only show files/folders in C:\ directly.

**3. -File**

* This ensures **only files** are listed, not folders.

**4. -ErrorAction SilentlyContinue**

* This tells PowerShell **not to display error messages** (e.g., "Access Denied" when it can’t open certain system folders).
* Without this, you might see a lot of red text when scanning C:\.

**5. | (Pipeline)**

* Passes the list of files found to the **next command**.

**6. Where-Object { $\_.Length -gt 100MB }**

* Where-Object filters items **based on a condition**.
* { $\_.Length -gt 100MB } means:
  + $\_ → represents the current file in the pipeline.
  + .Length → the file’s size in **bytes**.
  + -gt 100MB → **greater than 100 megabytes**.
* So this **keeps only files larger than 100MB**.

**Translation :-**

“Search through the entire C drive (including all subfolders), list only files, ignore errors, and show only those files that are bigger than 100MB.”

We can also do this :-

* Get-ChildItem -Path C:\ -Recurse -File -ErrorAction SilentlyContinue |

Where-Object { $\_.Length -gt 100MB } |Sort-Object Length -Descending

**Explanation of the pipeline step-by-step:**

Let’s break down this pipeline:

* Get-Process | Where-Object { $\_.CPU -gt 100 } | Select-Object Name, CPU
* Get-Process → Gets all process objects.
* Where-Object → Filters only those with CPU usage > 100.
* Select-Object → Projects only the name and CPU usage.

Here, $\_ is a **special variable** that refers to the current object in the pipeline.

Example:-Find top 3 services that are set to auto-start.

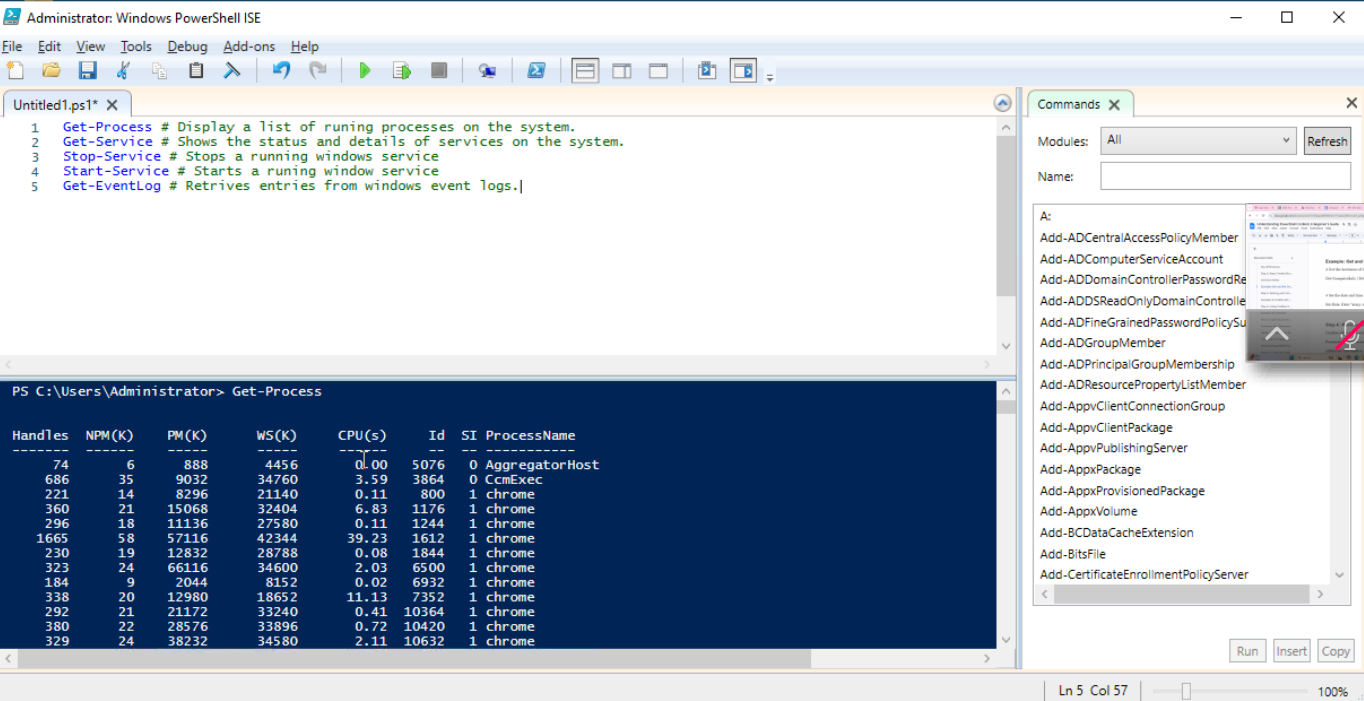
* Get-Service | Where-Object { $\_.StartType -eq "Automatic" } | Select-Object -First 3

| **Tip** | **Description** |
| --- | --- |
| Use $\_ inside script blocks | It refers to the current object |
| Objects are passed, not text | This is key PowerShell power |
| Chain unlimited commands | Keep adding more commands using ` |
| Avoid unnecessary Select-Object early | It may cut off properties you need later |

**Questions:-**

**Q: What does the PowerShell pipeline do?**  
A: It allows the output of one cmdlet to be passed as input to another, maintaining object structure.

**Q: What's the role of $\_ in the pipeline?**  
A: It refers to the current object in the pipeline for filtering or transforming.



**3. Key Cmdlets in PowerShell**

PowerShell has **200+ built-in cmdlets**, but some are used **90% of the time** in admin, automation, and scripting work.

**A. File & Folder Management Cmdlets**

| **Cmdlet** | **Purpose** | **Example** |
| --- | --- | --- |
| Get-ChildItem (ls, dir) | Lists files/folders | Get-ChildItem C:\ |
| New-Item | Creates file/folder | New-Item -Path C:\temp\test.txt -ItemType File |
| Remove-Item (rm, del) | Deletes file/folder | Remove-Item C:\temp\test.txt |
| Copy-Item | Copies file/folder | Copy-Item C:\test.txt C:\Backup\ |
| Move-Item | Moves file/folder | Move-Item C:\test.txt D:\ |

**Practise:-**

**# Create a folder and a file in it**

* New-Item -Path C:\Demo -ItemType Directory
* New-Item -Path C:\Demo\notes.txt -ItemType File -Value "PowerShell Rocks!"

**# List files inside C:\Demo**

* Get-ChildItem C:\Demo

**Real-world example:** Automating backup scripts by copying files daily.

**B. System Information Cmdlets**

| **Cmdlet** | **Purpose** | **Example** |
| --- | --- | --- |
| Get-Process | Lists running processes | Get-Process |
| Stop-Process | Stops a process | Stop-Process -Name notepad |
| Get-Service | Lists Windows services | Get-Service |
| Start-Service / Stop-Service | Controls services | Start-Service -Name Spooler |
| Get-EventLog | Reads event logs | Get-EventLog -LogName System -Newest 10 |

**Practise:-**

**# See all running processes sorted by memory usage**

* Get-Process | Sort-Object -Property WS -Descending | Select-Object -First 5

**# Start Print Spooler service**

* Start-Service -Name Spooler

**Real-world example:** Monitoring and restarting crashed services automatically.

**C. Networking Cmdlets**

| **Cmdlet** | **Purpose** | **Example** |
| --- | --- | --- |
| Test-Connection | Ping equivalent | Test-Connection google.com |
| Get-NetIPConfiguration | Shows IP settings | Get-NetIPConfiguration |
| Get-NetAdapter | Network interface info | Get-NetAdapter |

**Practise:-**

**# Ping Google 4 times**

* Test-Connection google.com -Count 4

**# Show network adapters**

* Get-NetAdapter

**Real-world example:** Network troubleshooting scripts for multiple servers.

**D. Security & Permissions Cmdlets**

| **Cmdlet** | **Purpose** | **Example** |
| --- | --- | --- |
| Get-Acl | View file permissions | Get-Acl C:\Demo\notes.txt |
| Set-Acl | Modify permissions | *(More advanced, needs ACL object)* |
| Get-LocalUser | Lists local users | Get-LocalUser |
| New-LocalUser | Creates a local user | New-LocalUser TestUser -Password (Read-Host -AsSecureString) |

**Real-world example:** Setting up user accounts and managing file permissions.

**E. Scripting & Output Cmdlets**

| **Cmdlet** | **Purpose** | **Example** |
| --- | --- | --- |
| Write-Output | Sends output to pipeline | Write-Output "Hello PowerShell" |
| Write-Host | Prints directly to console | Write-Host "Hello World" |
| Out-File | Writes output to file | `Get-Process |
| Export-Csv | Saves output as CSV | `Get-Service |
| Import-Csv | Reads CSV into objects | Import-Csv services.csv |

**Practise :-**

**# Export process list to CSV**

* Get-Process | Export-Csv processes.csv -NoTypeInformation

**# Import the CSV and view**

* Import-Csv processes.csv

**Real-world example:** Automated reporting scripts that export data for managers.

**F. System Configuration Cmdlets**

| **Cmdlet** | **Purpose** | **Example** |
| --- | --- | --- |
| Set-ExecutionPolicy | Controls script execution | Set-ExecutionPolicy RemoteSigned |
| Get-ExecutionPolicy | Shows execution policy | Get-ExecutionPolicy |
| Get-Module | Lists loaded modules | Get-Module |
| Import-Module | Loads a module | Import-Module ActiveDirectory |

**Real-world example:** Enabling PowerShell scripts in a fresh Windows installation.

**Key takeaway:**Mastering these cmdlets is like knowing the alphabet of PowerShell — you’ll use them in almost every script.