**Today Assignment – Date: August 8,2025**

**Name: 34733-Prasang Kumar Singh**

**Gmail:** [**thakurprasang23@gmail.com**](mailto:thakurprasang23@gmail.com)

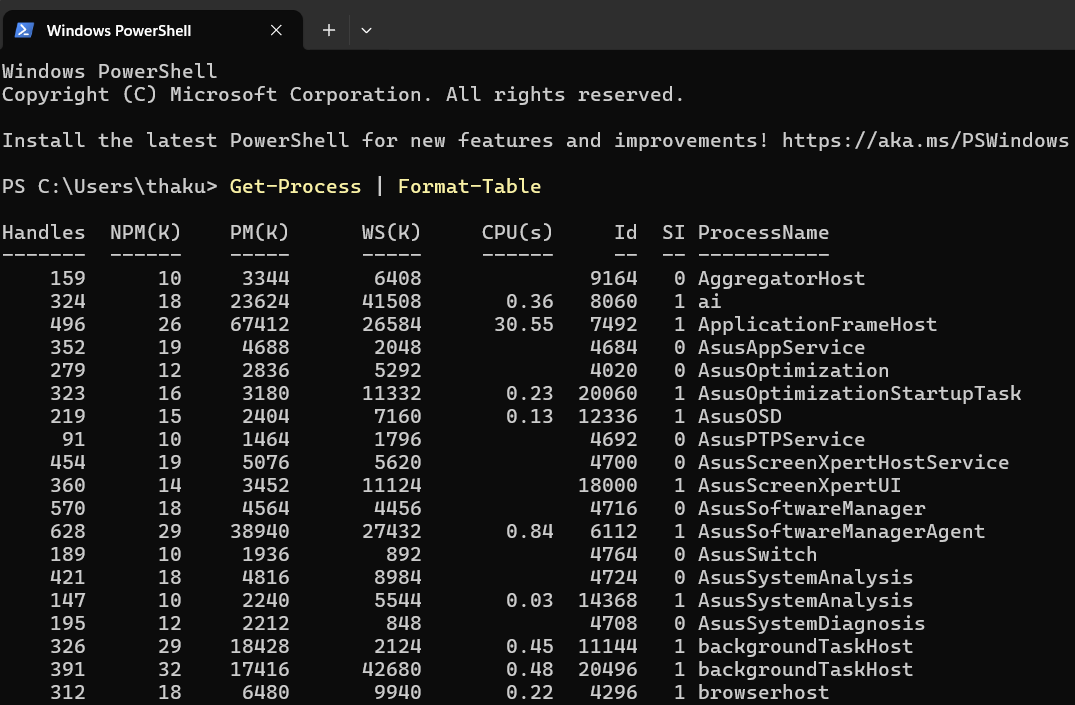
**Today topic:-**

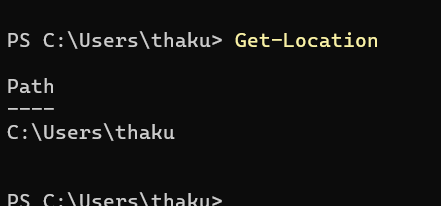
* Introducing to Cmdlets
* The PowerShell Pipeline
* Key Cmdlets
* WMI & PowerShell.
* Pipeline Filtering & Operators
* Scripting Overview
* Project: Exploring Cmdlet Syntax
* Project: Automate a Task with a Cmdlet Script
* Project: Create a PowerShell Cmdlet Cheat Sheet

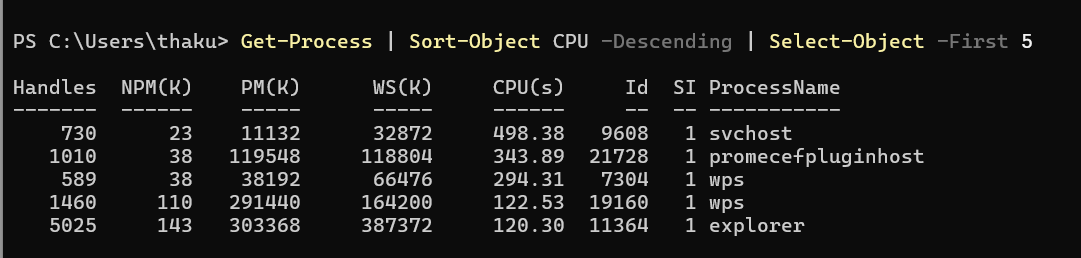
**Introducing to Cmdlets**

1. Cmdlets are built-in commands in PowerShell used for performing specific tasks.
2. They follow a “Verb-Noun” format (e.g., Get-Process, Start-Service) for clarity.
3. Cmdlets can be connected using the pipeline (|) to create complex command sequences.
4. They work with objects instead of plain text, making scripts more advanced and flexible.
5. PowerShell includes many cmdlets by default, and users can create their own.
6. Cmdlets are not separate programs—they are part of the PowerShell environment.
7. Each cmdlet takes input, processes it, and passes the output to the next cmdlet if needed.

**Some hands on practice Screenshots**







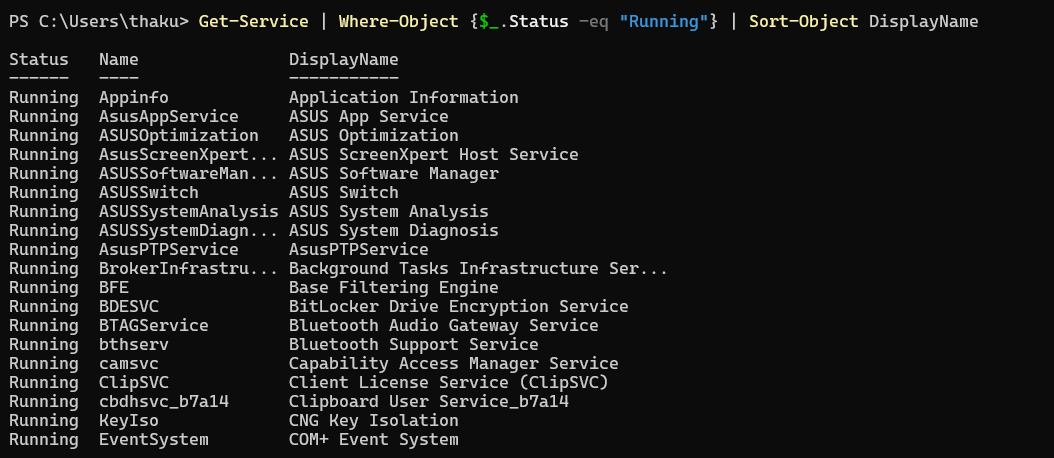
**The PowerShell Pipeline**

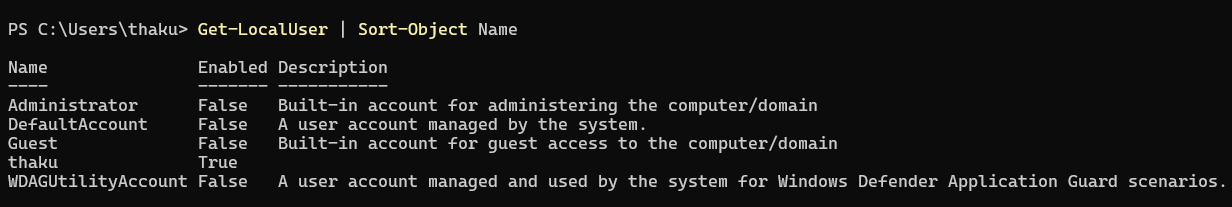
**The PowerShell pipeline** is a method of connecting multiple commands (cmdlets) in a sequence so that the output of one command becomes the input for the next. It uses the | (pipe) symbol to pass data between commands efficiently.

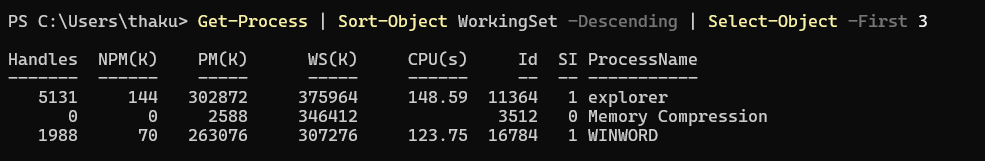
The result of Cmdlet1 goes to Cmdlet2, then its result goes to Cmdlet3

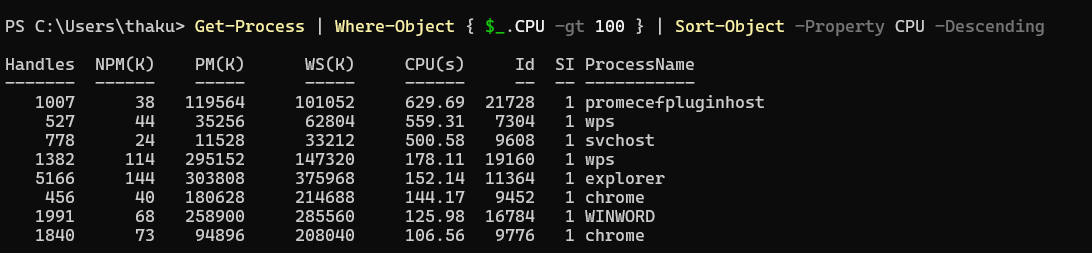
* The pipeline connects cmdlets to work together.
* Uses the | symbol to link commands.
* Passes **data as objects**, not plain text.
* Example: Get-Process | Sort-Object CPU – gets and sorts processes by CPU usage.
* Helps in building **step-by-step powerful scripts**.
* Makes tasks easier, cleaner, and more efficient.
* Works with **rich structured data**.
* Widely used for managing systems, files, and services.

**Some hands on practice Screenshots**

****

****

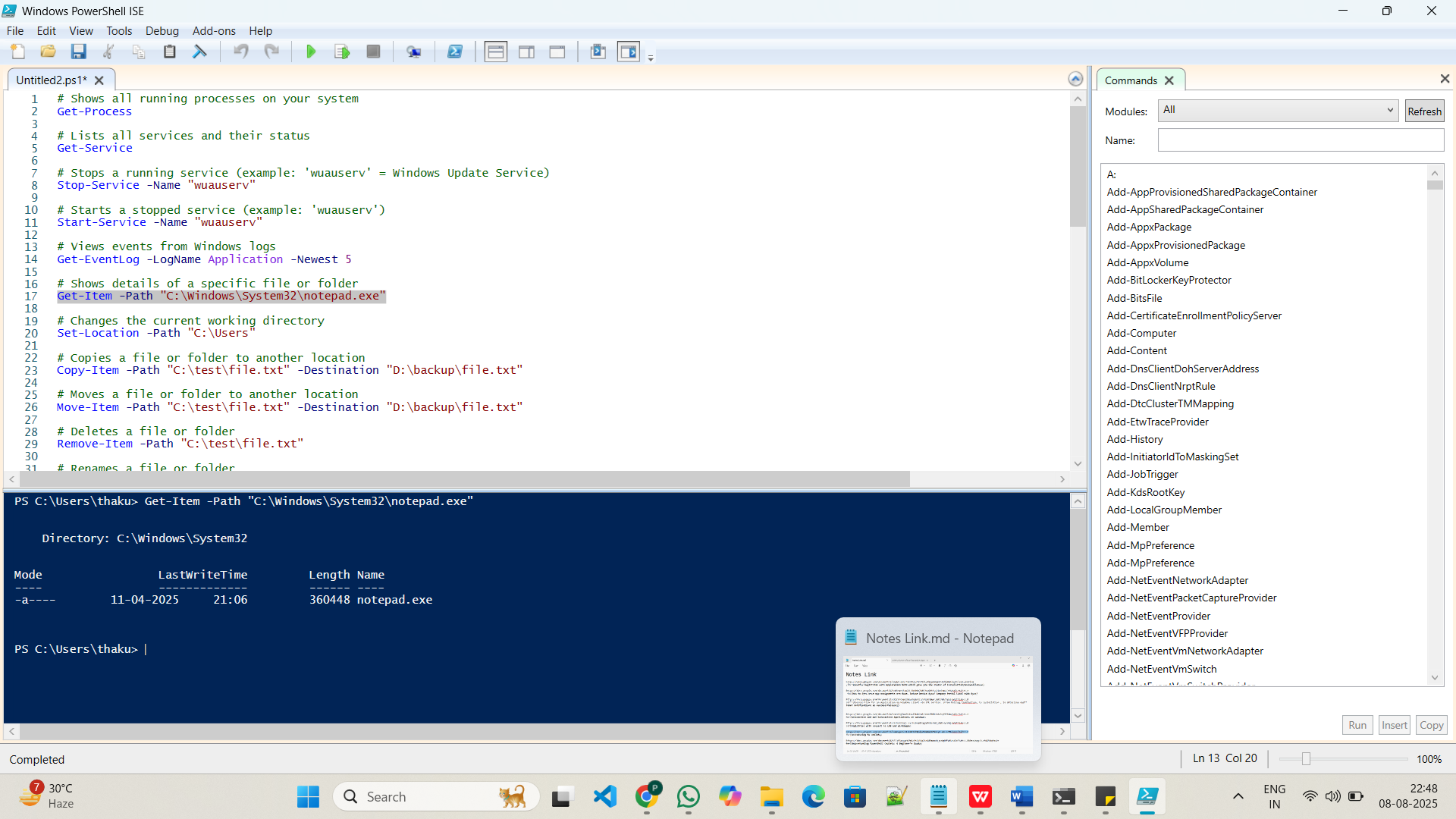
****

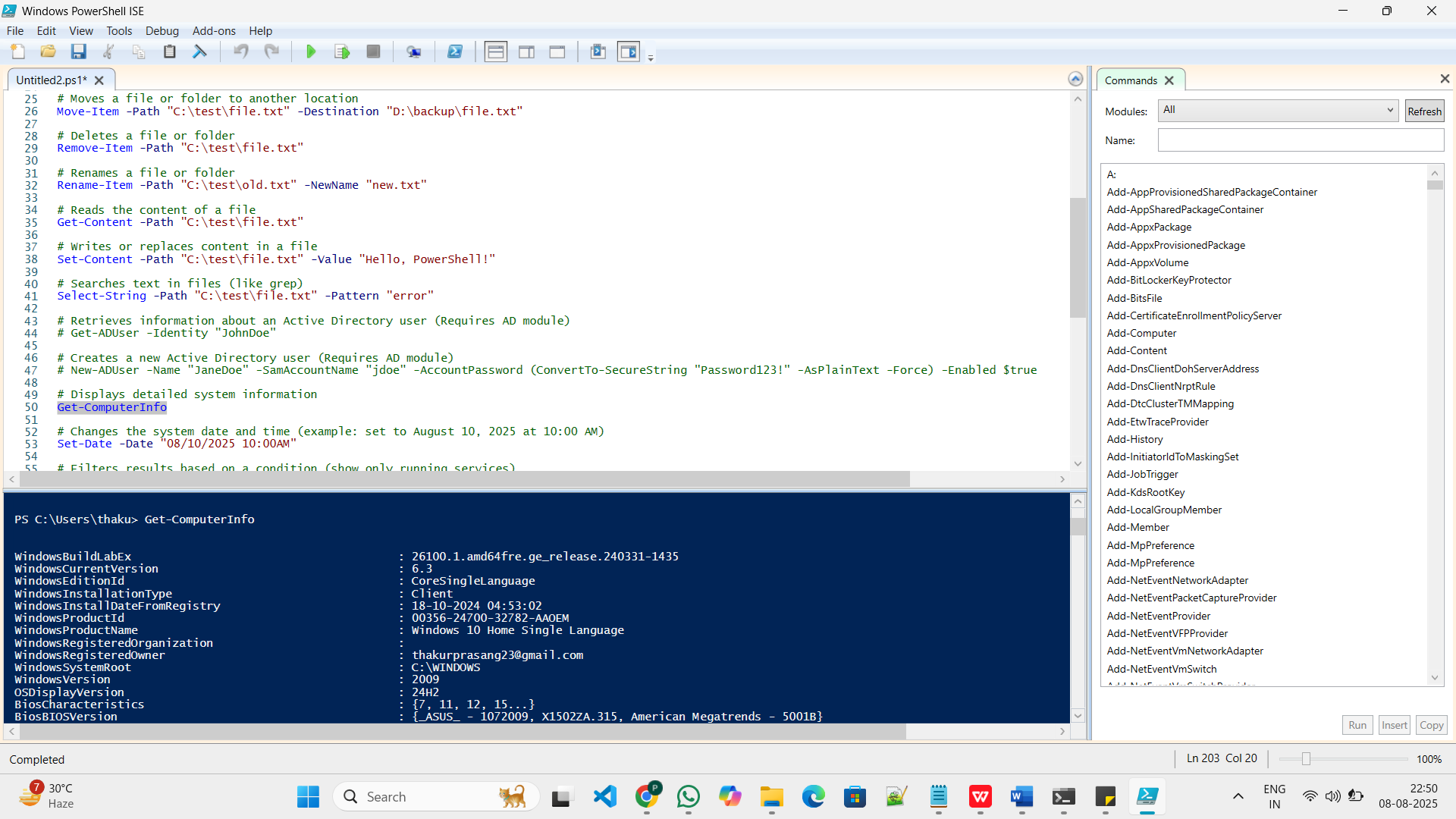
****

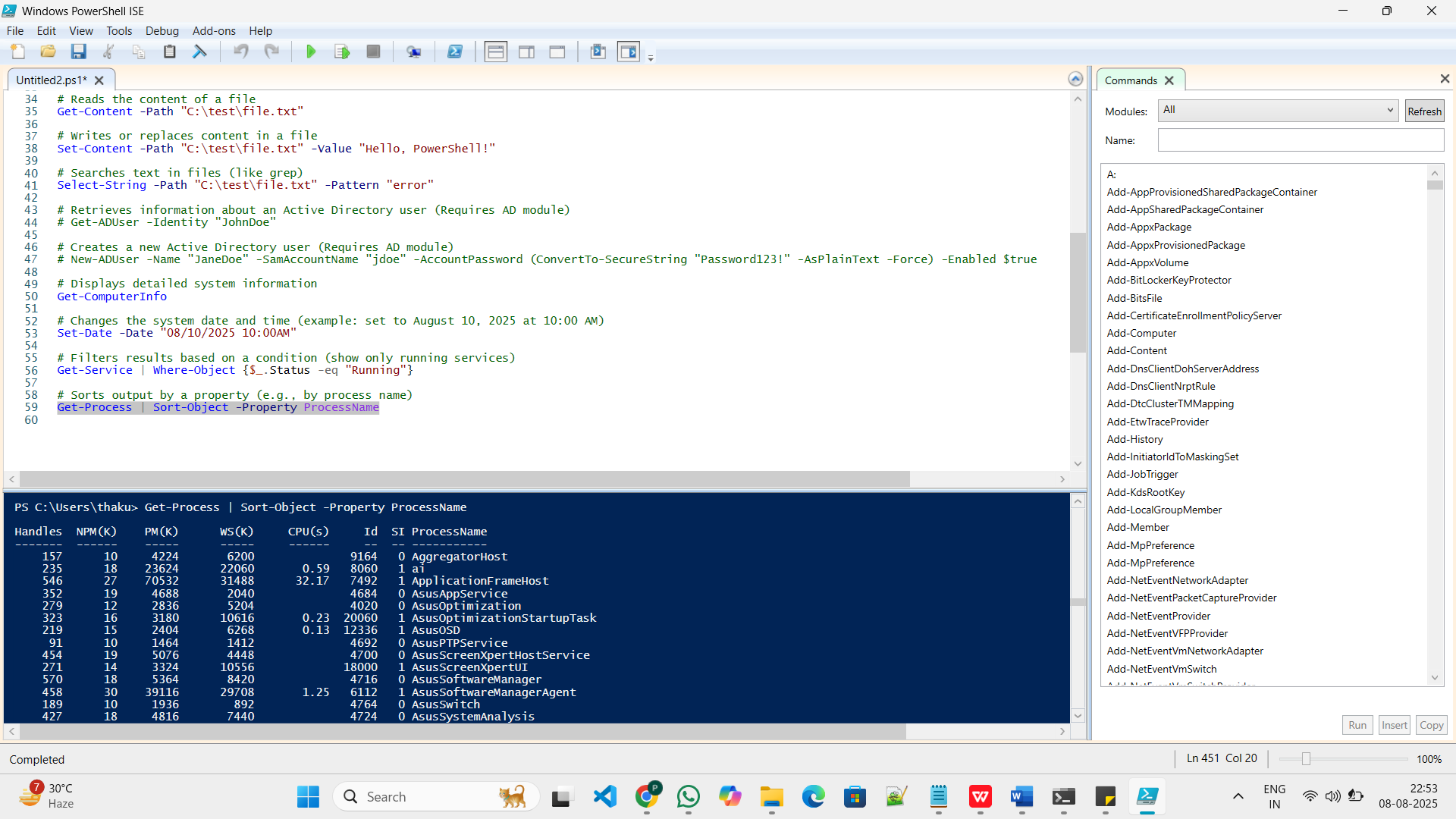
**Key Cmdlets**

**Get-Process** – Shows all running processes on your system.  
**Get-Service** – Lists all services and their status.  
**Stop-Service** – Stops a running service.  
**Start-Service** – Starts a stopped service.  
**Get-EventLog** – Views events from Windows logs.  
**Get-Item** – Shows details of a specific file or folder.  
**Set-Location** – Changes the current working directory.  
**Copy-Item** – Copies a file or folder to another location.  
**Move-Item** – Moves a file or folder to another location.  
**Remove-Item** – Deletes a file or folder.  
**Rename-Item** – Renames a file or folder.  
**Get-Content** – Reads the content of a file.  
**Set-Content** – Writes or replaces content in a file.  
**Select-String** – Searches text in files (like grep).  
**Get-ADUser** – Retrieves information about an Active Directory user.  
**New-ADUser** – Creates a new Active Directory user.  
**Get-ComputerInfo** – Displays detailed system information.  
**Set-Date** – Changes the system date and time.  
**Where-Object** – Filters results based on a condition.  
**Sort-Object** – Sorts output by a property.

**Some hands on practice Screenshots**



****

****

**WMI & PowerShell.**

WMI stands for Windows Management Instrumentation.

It lets you access system information like hardware, software, and services.

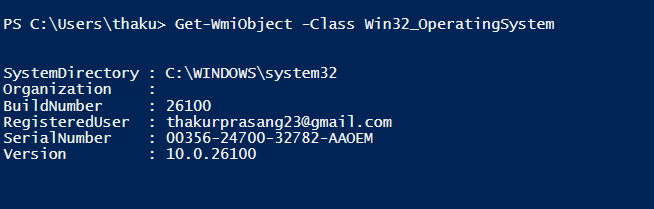
PowerShell can use WMI to get or change this info easily.

Example:

Get-WmiObject -Class Win32\_OperatingSystem

This shows details about your Windows OS like version, build number, and more.

WMI is useful for tasks like checking disk space, system info, or managing services remotely — all through PowerShell.

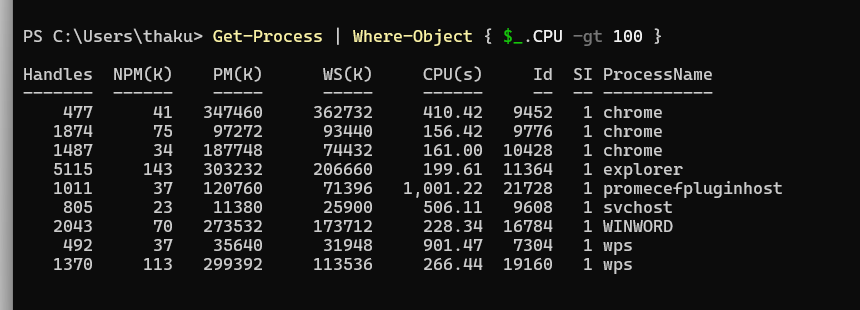


**Pipeline Filtering & Operators**

**1. Pipeline (|)**

* The pipeline symbol (|) is used to pass the output of one cmdlet as the input to another cmdlet.
* It allows chaining multiple commands together to perform complex tasks step-by-step.

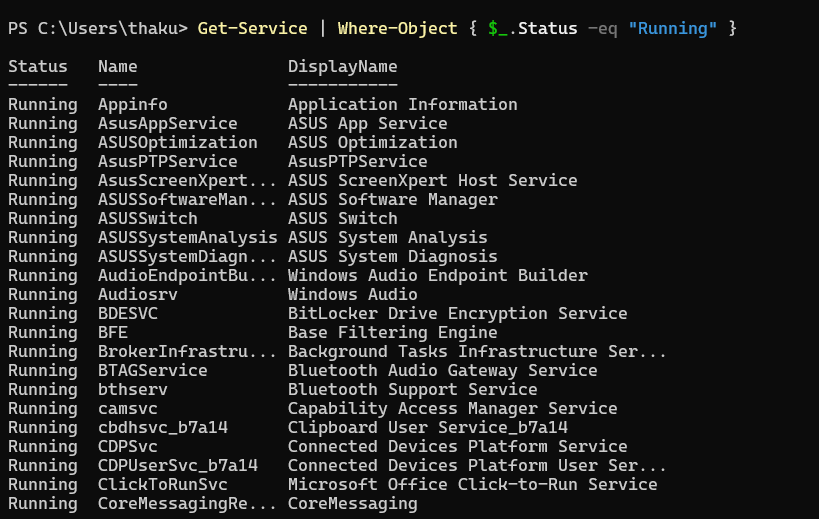
Example:- Get-Process | Where-Object { $\_.CPU -gt 100 }



**Filtering**

* Filtering helps extract specific items from a large set of data based on certain conditions.
* In PowerShell, Where-Object is the most common cmdlet used for filtering.
* Syntax:

**<Command> | Where-Object { $\_.Property -operator value }**



**Operators**

1. -eq means **equal to** (e.g., Status equals "Running").
2. -ne means **not equal to** (e.g., Name is not "Admin").
3. -gt means **greater than** (e.g., CPU usage is more than 100).
4. -lt means **less than** (e.g., File size is less than 1MB).
5. -ge means **greater than or equal to** (e.g., Age is 18 or more).
6. -le means **less than or equal to** (e.g., Score is 50 or below).
7. -like means **matches a wildcard pattern** (e.g., Name like "*Test*").
8. -match means **matches a regular expression pattern** (e.g., Name starts with "A").

**Scripting Overview.**

PowerShell scripting means writing a series of commands in a .ps1 file to automate tasks.

Instead of typing each command one by one, you save them in a script and run all at once.

Why we use scripts?

* Save time
* Repeat tasks easily
* Reduce manual errors
* Automate system tasks, app installs, reports, etc.

**Project: Exploring Cmdlet Syntax**

**Step 1:** Open PowerShell ISE or Code Editor

* PowerShell ISE:  
  Press Windows + S → Search for PowerShell ISE → Click to open.

save with .ps1 extension.

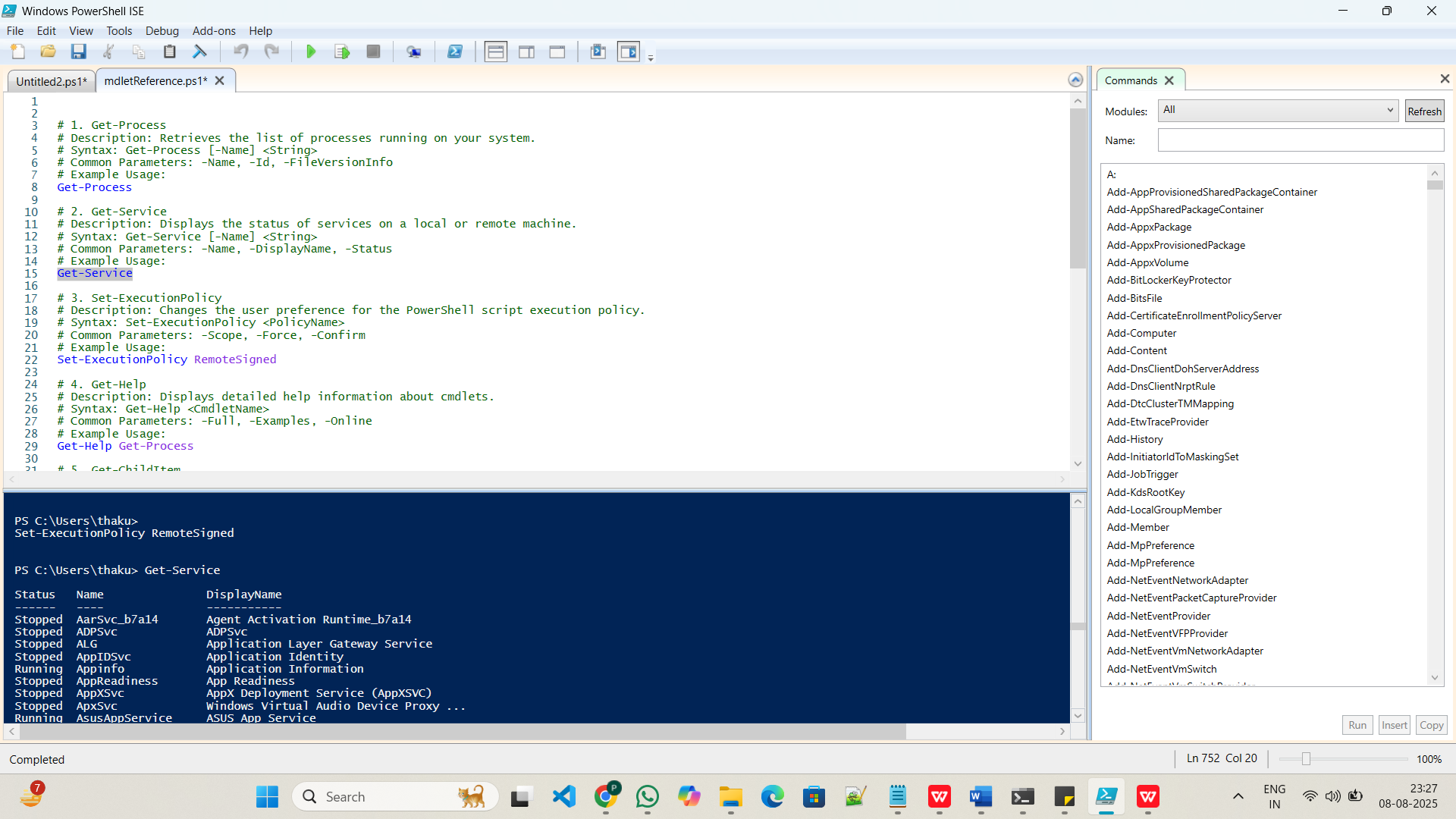
**Step 2: Create a New Script File**

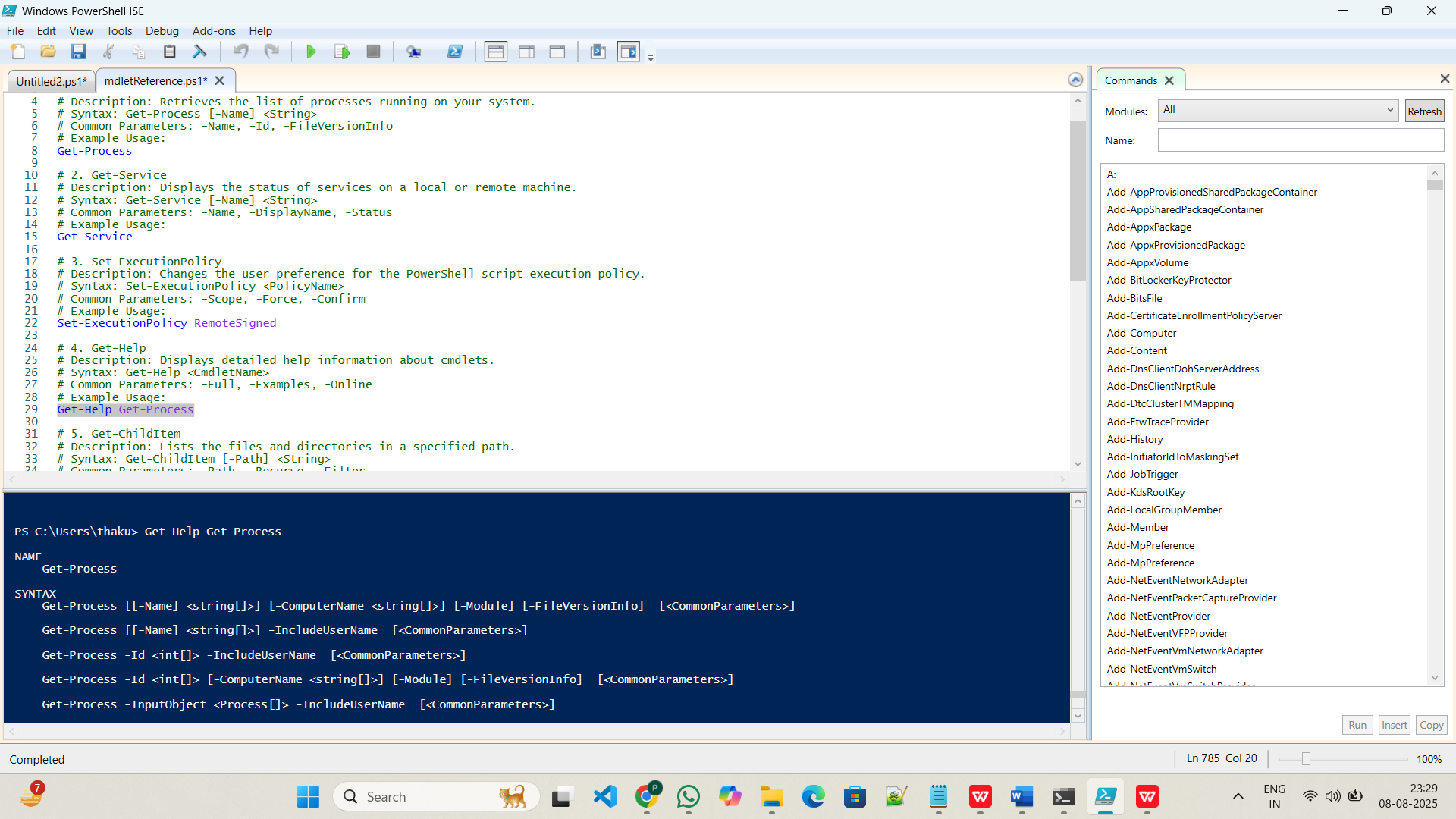
* File Name: CmdletReference.ps1
* In PowerShell ISE:  
  File > New, then File > Save As > CmdletReference.ps1

**Step 3: Choose Five Cmdlets**

Let’s use:

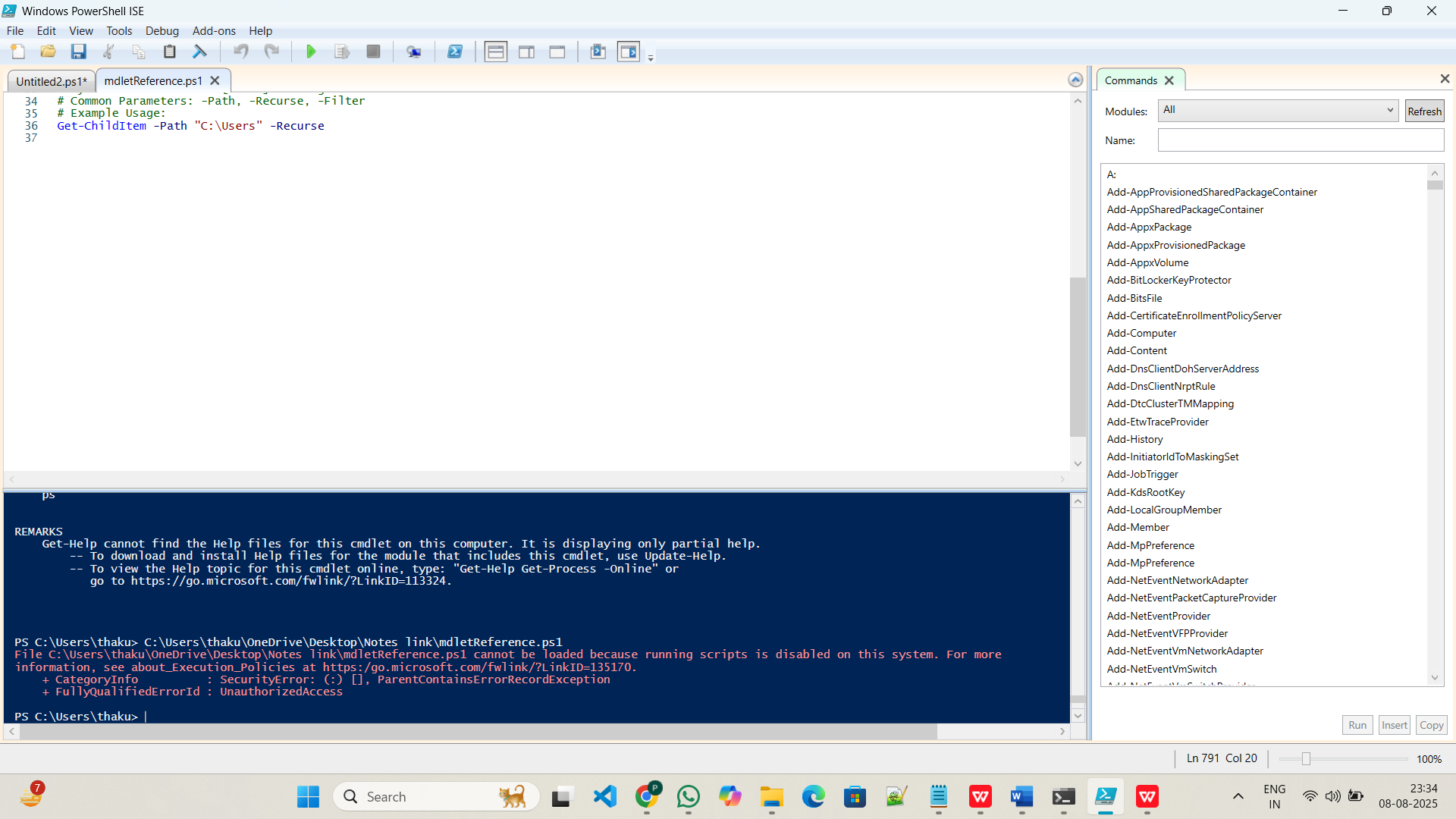
1. Get-Process
2. Get-Service
3. Set-ExecutionPolicy
4. Get-Help
5. Get-ChildItem

****

****

**Step 5: Save and Run Your Script**

1. Press **Ctrl + S** to save.
2. Click **Run Script (F5)** in PowerShell ISE.  
   This will execute each command and show the output in the console.



It cannot be loaded because running scripts is disabled on this system.

Scripts make PowerShell powerful and great for automation.

**Project: Automate a Task with a Cmdlet Script**

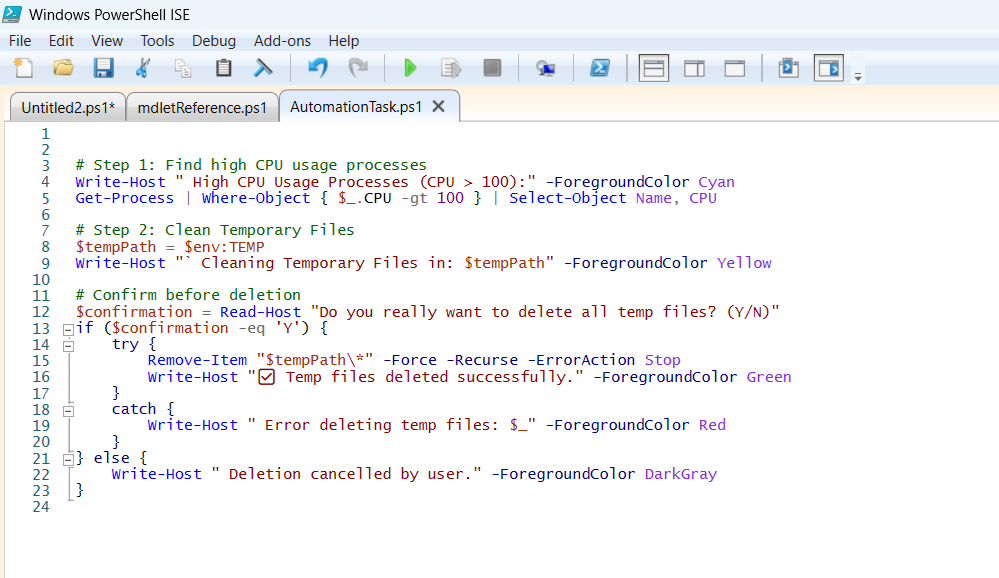
Step 1: Choose Tasks to Automate

* Task 1: Show processes using more than 100 CPU units
* Task 2: Clean temp files from %temp% folder

Step 2: Create a New Script File

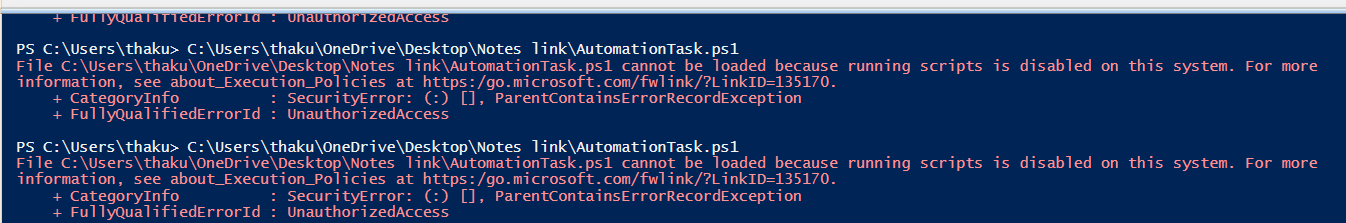
* File Name: AutomationTask.ps1
* Open PowerShell ISE or any text editor
* Save the script with .ps1 extension

**Step 3:-**

****

Step 4: Run with Proper Permissions

1. Right-click PowerShell ISE → Run as Administrator
2. Press F5 to run the script
3. It will:
   * Show high-CPU processes
   * Ask for your permission to clean temp files
   * Delete files if you say **Y**



It cannot be loaded because running scripts is disabled on this system.

Scripts make PowerShell powerful and great for automation.



**Project 3: Create a PowerShell Cmdlet Cheat Sheet**

**PowerShell Cmdlet Cheat Sheet**

**Basic Cmdlets**

Commonly used cmdlets for getting help, discovering commands, and working with content.

|  |  |  |
| --- | --- | --- |
| Cmdlet | Description | Example Usage |
| Get-Help | Shows help information about commands | Get-Help Get-Process |
| Get-Command | Lists all available commands | Get-Command |
| Get-Content | Displays the content of a file | Get-Content C:\example.txt |
| Set-Content | Writes or replaces content in a file | Set-Content C:\example.txt "Hi" |
| Select-String | Searches text/files for specific patterns | Select-String -Pattern "error" log.txt |
| Where-Object | Filters objects based on a condition | Get-Process | Where-Object {$\_.CPU -gt 100} |
| Sort-Object | Sorts objects by property | Get-Process | Sort-Object CPU |
| Out-File | Sends output to a file | Get-Process | Out-File processes.txt |
| Get-Variable | Gets variables in the current session | Get-Variable |
| Set-Variable | Creates or changes variables | Set-Variable -Name MyVar -Value 1 |
| Clear-Host | Clears the screen | Clear-Host |
| Measure-Object | Measures properties of objects | Get-ChildItem | Measure-Object |
| Write-Output | Sends output to the pipeline or console | Write-Output "Hello, world!" |
| Write-Host | Displays colored output on console | Write-Host "Success" -ForegroundColor Green |
| Get-History | Shows command history | Get-History |
| Invoke-History | Runs a command from history | Invoke-History 3 |
| Get-Alias | Gets command aliases | Get-Alias |
| Set-Alias | Creates an alias for commands | Set-Alias ll Get-ChildItem |
| ForEach-Object | Performs an operation on each item in a collection | Get-Process | ForEach-Object { $\_.Name } |
| Get-Location | Shows the current directory | Get-Location |

**File System Cmdlets**

Cmdlets to work with files and folders (items).

|  |  |  |
| --- | --- | --- |
| Cmdlet | Description | Example Usage |
| New-Item | Creates new files or directories | New-Item -Path "C:\Test" -ItemType Directory |
| Copy-Item | Copies files or folders | Copy-Item C:\file.txt D:\Backup\file.txt |
| Move-Item | Moves files or folders | Move-Item C:\file.txt D:\Archive\file.txt |
| Remove-Item | Deletes files or folders | Remove-Item C:\Temp\\* -Recurse |
| Rename-Item | Renames a file or folder | Rename-Item "old.txt" "new.txt" |
| Get-ChildItem | Lists files and folders in a directory | Get-ChildItem C:\Temp |
| Test-Path | Checks if a path exists | Test-Path C:\Temp |
| Get-Item | Gets a specific file or folder | Get-Item C:\file.txt |
| Set-Item | Changes the value or content of an item | Set-Item -Path Env:Path -Value $newPath |
| Join-Path | Combines strings into a path | Join-Path C:\Temp "file.txt" |
| Split-Path | Gets part of a path | Split-Path C:\Temp\file.txt -Parent |
| Clear-Content | Clears the content of a file | Clear-Content C:\log.txt |
| Get-ItemProperty | Gets properties of a file or folder | Get-ItemProperty C:\file.txt |
| Set-ItemProperty | Sets properties of a file or folder | Set-ItemProperty C:\file.txt -Name IsReadOnly -Value $true |
| Copy-ItemProperty | Copies properties from one item to another | Copy-ItemProperty C:\file1.txt C:\file2.txt |
| Get-PSDrive | Lists drives available (file system, registry) | Get-PSDrive |
| New-PSDrive | Creates a new drive | New-PSDrive -Name X -PSProvider FileSystem -Root C:\ |
| Remove-ItemProperty | Deletes property of an item | Remove-ItemProperty -Path C:\file.txt -Name IsReadOnly |
| Get-ChildItem -Recurse | Lists all items in directory and subdirectories | Get-ChildItem -Recurse C:\Temp |
| Format-Table | Formats output as a table | Get-ChildItem | Format-Table |

**Network Cmdlets**

Cmdlets for checking and managing network configurations.

|  |  |  |
| --- | --- | --- |
| Cmdlet | Description | Example Usage |
| Test-Connection | Checks network connectivity (like ping) | Test-Connection google.com |
| Get-NetIPAddress | Shows IP address information | Get-NetIPAddress |
| Get-NetIPConfiguration | Displays IP configuration details | Get-NetIPConfiguration |
| Get-NetAdapter | Gets network adapter information | Get-NetAdapter |
| Enable-NetAdapter | Enables a network adapter | Enable-NetAdapter -Name "Ethernet" |
| Disable-NetAdapter | Disables a network adapter | Disable-NetAdapter -Name "Wi-Fi" |
| Set-DnsClientServerAddress | Sets DNS servers for a network adapter | Set-DnsClientServerAddress -InterfaceAlias "Ethernet" -ServerAddresses ("8.8.8.8") |
| Get-DnsClientServerAddress | Gets DNS server addresses for adapters | Get-DnsClientServerAddress |
| Test-NetConnection | Tests network connection with detailed info | Test-NetConnection google.com -InformationLevel Detailed |
| Resolve-DnsName | Performs DNS query | Resolve-DnsName google.com |
| Get-NetRoute | Gets IP route information | Get-NetRoute |
| New-NetIPAddress | Adds a new IP address to an interface | New-NetIPAddress -InterfaceAlias "Ethernet" -IPAddress 192.168.1.10 -PrefixLength 24 |
| Remove-NetIPAddress | Removes IP address from interface | Remove-NetIPAddress -IPAddress 192.168.1.10 |
| Get-NetFirewallRule | Lists firewall rules | Get-NetFirewallRule |
| Enable-NetFirewallRule | Enables a firewall rule | Enable-NetFirewallRule -Name "FPS-Rule" |
| Disable-NetFirewallRule | Disables a firewall rule | Disable-NetFirewallRule -Name "FPS-Rule" |
| Show-NetConnectionProfile | Displays network connection profile | Get-NetConnectionProfile |
| Get-NetIPAddress -AddressFamily IPv6 | Shows IPv6 addresses | Get-NetIPAddress -AddressFamily IPv6 |
| Restart-NetAdapter | Restarts a network adapter | Restart-NetAdapter -Name "Wi-Fi" |
| Get-NetNeighbor | Displays neighbor cache entries | Get-NetNeighbor |