PowerShell Assignment

Date: 08-August-2025

Batch: WiproNGA_DWS_B5_25VID2550

Candidate Name: Sooraj B

User ID: **34753**

1. Introduction to Cmdlets

1.1 What are Cmdlets?

Definition: Lightweight, single-function commands in PowerShell (e.g., `Get-Process`, `Set-Date`).

Key Features:

Verb-Noun Naming: `Verb-Action` format (e.g., `Get-`, `Set-`, `New-`).

Object-Oriented: Outputs .NET objects (not just text).

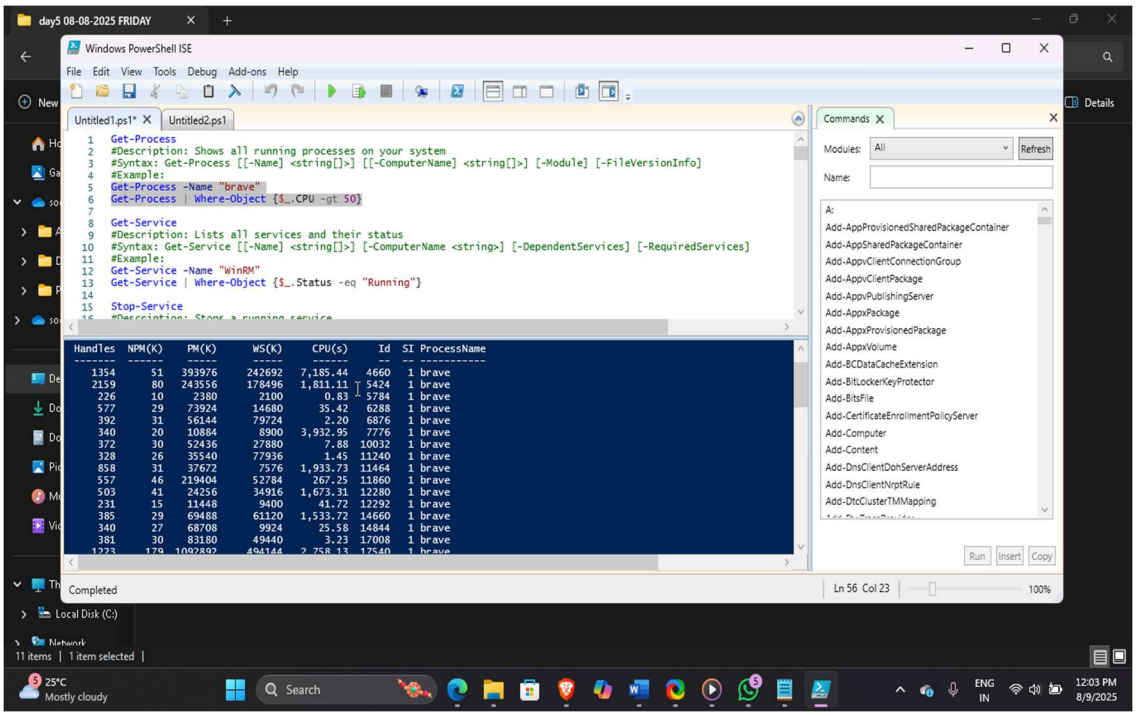
Pipeline Support: Can be chained (e.g., `Get-Service | Where-Object {$_. Status -eq "Running"}`).

1.2 Common Cmdlets

Cmdlet	Purpose	Example
`Get-Command`	Lists all available cmdlets	`Get-Command -Verb Get`
`Get-Help`	Displays help for a cmdlet	`Get-Help Get-Process -Examples`
`Get-Process`	Lists running processes	`Get-Process -Name "chrome"`
`Set-Execution Policy`	Configures script execution permissions	`Set-Execution Policy Remote Signed`

2. The PowerShell Pipeline

2.1 Pipeline Basics



Concept: Passes output of one cmdlet as input to another using `|`.

Example:

PowerShell

```
Get-Process | Where-Object {$_.CPU -gt 100} | Sort-Object CPU -Descending
```

Steps:

1. `Get-Process` → Retrieves all processes.
2. `Where-Object` → Filters processes with CPU > 100.
3. `Sort-Object` → Sorts by CPU usage (descending).

2.2 Filtering & Operators

Comparison Operators**

Operator	Description	Example

| `-eq` | Equal to | `5 -eq 5` → `True` |

| `-gt` | Greater than | `10 -gt 5` → `True` |

| `-like` | Wildcard match | `"file.txt" -like "*.txt"` |

Logical Operators

| Operator | Description | Example |

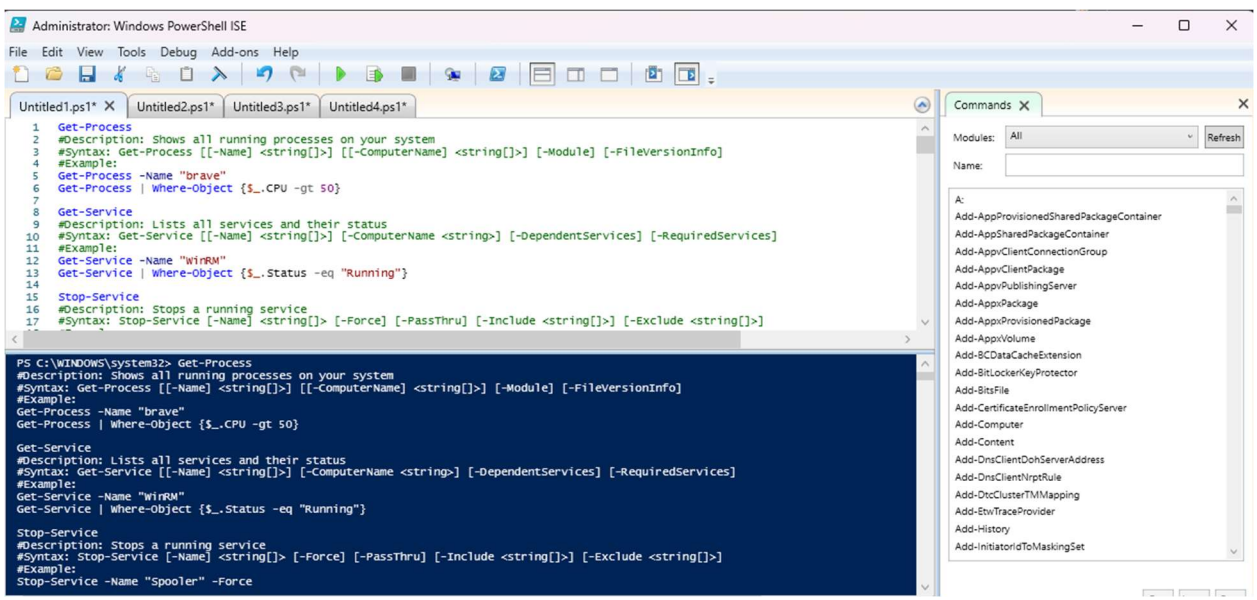
|---|---|---|

| `-and` | Both conditions true | `(5 -gt 2) -and (3 -lt 5)` |

| `-or` | Either condition true | `(5 -lt 2) -or (3 -lt 5)` |

3. WMI & PowerShell**

3.1 WMI Overview



Purpose: Manages Windows systems (hardware, OS, services) locally/remotely.

Key Cmdlets:

- `Get-WmiObject` (Legacy): Retrieves WMI data (e.g., `Get-WmiObject -Class Win32_OperatingSystem`).

- `Get-CimInstance` (Modern): Uses WS-Man protocol (e.g., `Get-CimInstance Win32_Processor`).

3.2 Practical Use Case

PowerShell

Check disk space remotely:

```
Get-CimInstance -ComputerName "Server01" -ClassName Win32_LogicalDisk |
 Where-Object {$_.DriveType -eq 3} |
 Select-Object DeviceID, FreeSpace, Size
```

4. Input, Output & Formatting

4.1 Input Methods

```
$name = Read-Host "Enter your name"
Write-output "Hello. $name"
```

```
PS C:\WINDOWS\system32> $name = Read-Host "Enter your name"
Write-output "Hello. $name"
Enter your name: Sooraj B
Hello. Sooraj B

PS C:\WINDOWS\system32>
```

User Input:

PowerShell

```
$name = Read-Host "Enter your name"
```

Pipeline Input:

PowerShell

```
Get-Service | Where-Object {$_.Status -eq "Stopped"}
```

```
Get-process -Name Notepad
# Get-process • This is the cmdlet, It lists the processes currently running or
#Name This is a parareter. It tells the cmdlet te filter results by process nam
#notepad This the argument (value) you pass to the -Nane parameter – in this ca
# you 're asking for the process named "notepad".
```

```
PS C:\WINDOWS\system32> Get-process -Name Notepad
# Get-process • This is the cmdlet, It lists the processes currently running on your
#Name This is a parareter. It tells the cmdlet te filter results by process name.
#notepad This the argument (value) you pass to the -Nane parameter – in this case,
# you 're asking for the process named "notepad".

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    882      42    48528       8220      42.67  24828   1 Notepad
```

4.2 Output Formatting

Cmdlet	Purpose	Example
`Format-Table` (`ft`)	Displays data in a table	`Get-Process | ft Name, CPU`
`Format-List` (`fl`)	Shows properties vertically	`Get-Service | fl *`
`Out-File`	Saves output to a file	`Get-Process | Out-File "processes.txt"`

5. Scripting Overview

5.1 PowerShell Scripts

```
PS C:\WINDOWS\system32> # Script: SystemInfo. ps1
# Author: Sooraj B
# Description: Displays basic system information
write-Host "Sooraj B"
write-Host "--------------------"
$COMPUTERNAME = $env:COMPUTERNAME
$os = Get-Cimrnstance Win32_OperatingSystem

$cpu = Get-CimInstance Win32_processor
write-Host "computer Name: $Sooraj B"
Write-Host "Operating System: $os"
write-Host "CPU: $cpu"
Sooraj B
```

File Extension: `.ps1` (e.g., `SystemInfo.ps1`).

Example Script:

PowerShell

Display system info

Write-Host "COMPUTER INFORMATION"

$os = Get-CimInstance Win32_OperatingSystem

Write-Host "OS: $($os.Caption)"

5.2 Automation Benefits

Time-Saving: Automate repetitive tasks (e.g., log cleanup).

Scalability: Manage multiple systems simultaneously.

Hands-On Task

Task: Create a Process Monitor Script

1. Script Name: `ProcessMonitor.ps1`

2. Requirements:

- List top 5 CPU-intensive processes.
- Save output to a file.

3. Solution:

PowerShell

```
Get-Process | Sort-Object CPU -Descending | Select-Object -First 5 | Out-File "HighCPUProcesses.txt"
```

Key Takeaways

Cmdlets: Use `Verb-Noun` commands for specific tasks.

Pipeline: Chain commands with `|` for efficient data processing.

WMI: Leverage `Get-CimInstance` for system management.

Scripting: Automate workflows with `.ps1` files.

Next Steps: Practice pipeline chaining and explore `Get-WmiObject` for advanced queries.