BATCH INFORMATION:

* Name: Sohini Sri Lenka
* User ID: 34747
* Batch Start Date: 04/08/2025
* Batch name: WiproNGA\_DWS\_B5\_25VID2550
* Batch ID: 25VID2550
* Assignment date: 8/8/25 - 9/8/25

INTRODUCTION TO Cmdlets:

Cmdlets (command-lets) are small, lightweight commands used in PowerShell to perform specific tasks.

Think of them like building blocks — each cmdlet does one focused job, and you can combine them to create bigger scripts or automate tasks.

**Key points about Cmdlets:**

* Used in PowerShell — not regular Command Prompt.
* It is Case-insensitive.
* Special syntax:  
  They follow a Verb-Noun format, like:

Get-Process

Set-Date

New-Item

Remove-Item

POWERSHELL PIPELINE :

In PowerShell, the pipeline is a way to send the output of one command directly as the input to another command — almost like an assembly line where each station does its part and passes it on.

Think of it like this:

* **Without pipeline:** You run one command, get the result, and then manually feed it into the next command.
* **With pipeline:** PowerShell automatically connects the output of the first command to the input of the second, without you storing it in a variable.

We will use the **|** (pipe) symbol between commands.

For example:

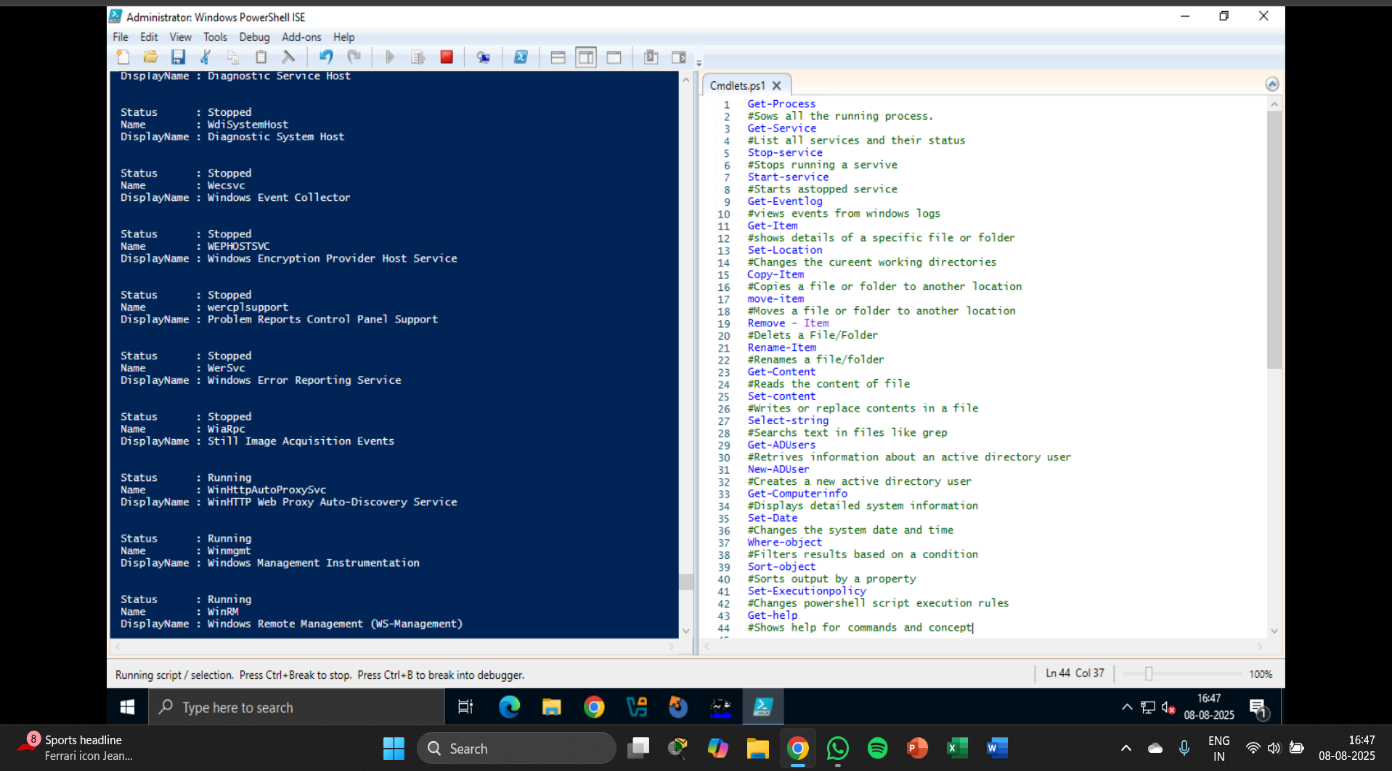
(Get-Process | Where-Object { $\_.CPU -gt 100 } | Sort-Object CPU -Descending)

KEY cmdlets:

Here r the all commands in one go:

Get-Process ,Get-Service , Stop-Service, Start-Service , Get-EventLog, Get-Item , Set-Location Copy-Item , Move-Item, Remove-Item, Rename-Item , Get-Content, Set-Content , Select-String, Get-ADUser , , New-ADUser , , Get-ComputerInfo , Set-Date , Where-Object, Sort-Object,

Set-ExecutionPolicy , Get-Help ,Get-ChildItem ,Remove-Item ,Test-Connection ,Get-NetIPAddress ,Get-Command



WMI&POWERSHELL :

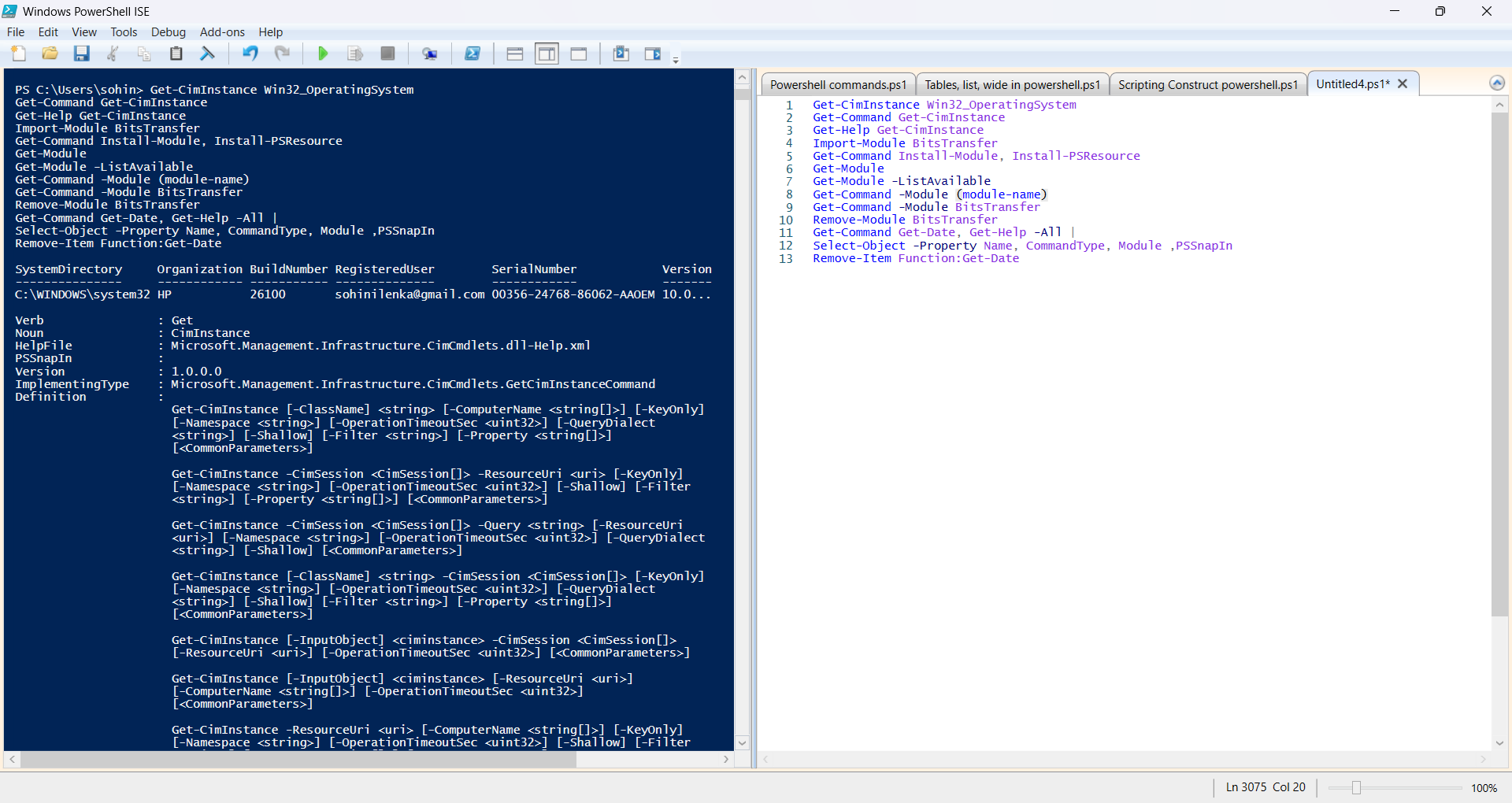
* Think of it as Windows’ “control center” for system information and management.
* It allows you to query, monitor, and control almost any aspect of the Windows operating system.
* You can use it to get info like:
  + CPU details
  + Installed software
  + System processes
  + Network configurations
* Works a bit like a database for system information — you can ask questions (“queries”) and get answers.

Example:

* “Hey WMI, give me the list of all running processes.”
* WMI will fetch that info from the system.

**PowerShell can talk to WMI using special cmdlets.  
This makes it easy to get or set system information without clicking through Windows menus.**

**Common Cmdlets:**

****

PIPELINE FILTERING & OPERATORS:

In PowerShell, pipeline filtering and operators are about controlling how data flows through a command sequence and applying conditions to decide what passes through. The **pipeline (|)** sends the output of one command as input to another. Filtering means allowing only the data that matches certain conditions to continue in the pipeline.

**Filtering Cmdlets**

* **Where-Object** - Keeps only the objects that match a condition.

**(Get-Process | Where-Object {$\_.CPU -gt 100})**

*Keeps only processes where CPU time is greater than 100 seconds.*

* **Select-Object** - Chooses specific properties (or the first few items).

**(Get-Service | Select-Object Name, Status)**

*Shows only the Name and Status of each service.*

OPERATORS:

Get-Service |

Where-Object {$\_.Status -eq 'Running'} |

Sort-Object Name |

Select-Object -First 5

INPUT, OUTPUT , FORMATTING:

PowerShell handles input and output through several cmdlets and techniques, including reading from the console, writing to the console, and formatting output for readability.

**Input**

* **Read-Host**: This cmdlet prompts the user for input and stores it as a string.

EX: $name = Read-Host "Please enter your name"

    Write-Host "Hello, $name!"

* **Get-Content**: Reads the content of a file.

EX: $content = Get-Content -Path "C:\example.log"

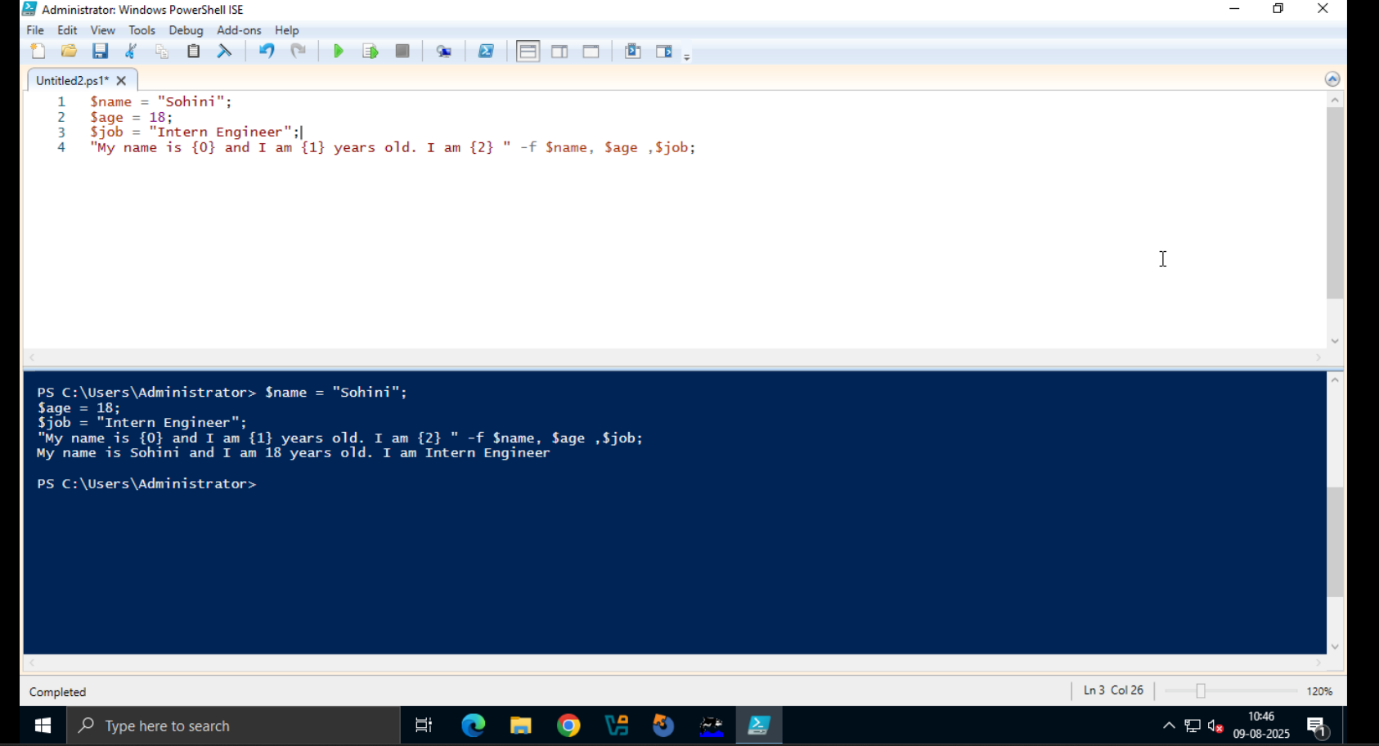
**Output**

* **Write-Host**: Displays output directly to the console.

    EX :Write-Host "This is a message."

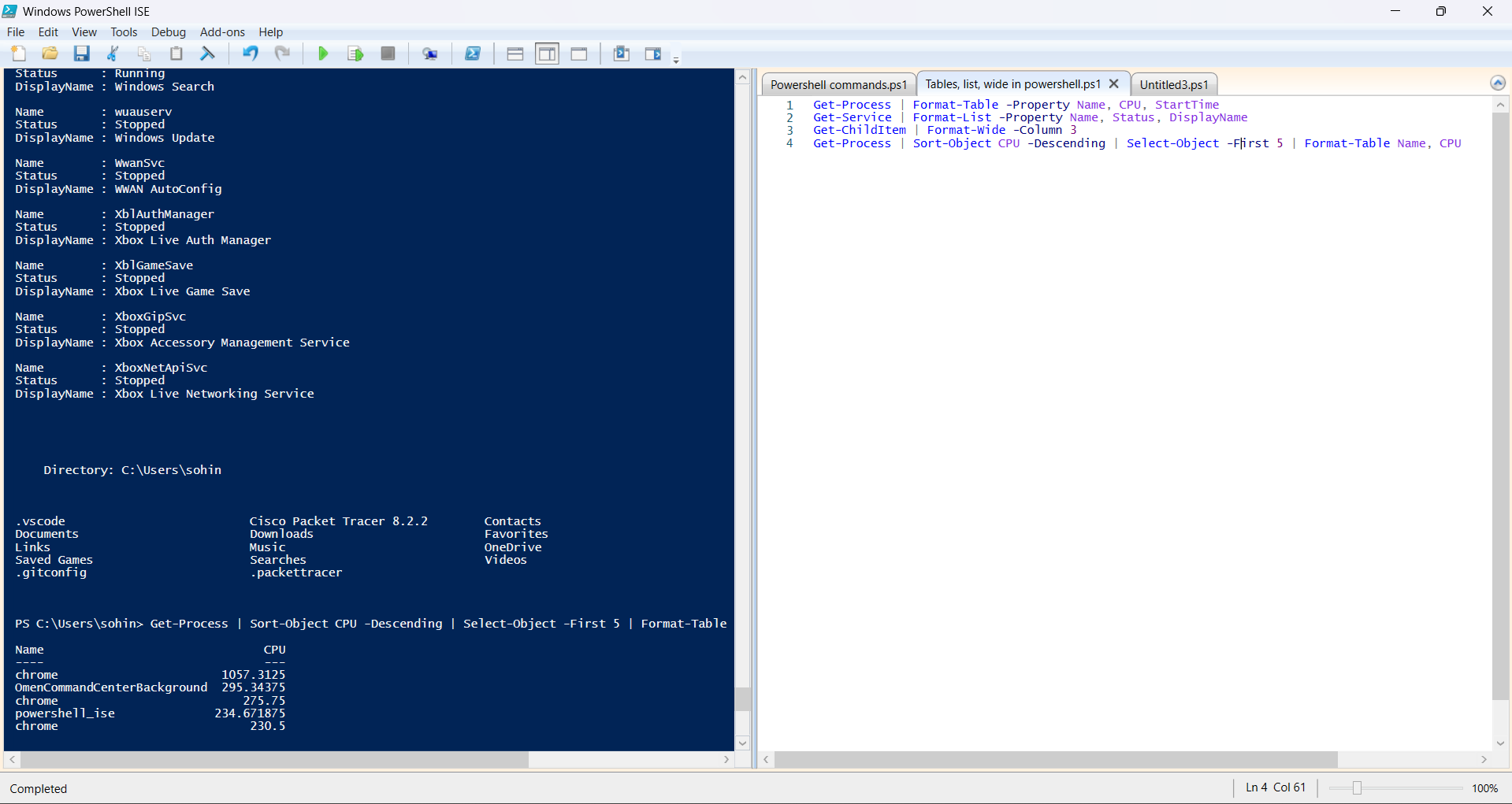
* **Out-File**: Writes output to a file.

    Get-Process | Out-File -FilePath "processes.txt"



FORMATTING:

* **Format-Table**: Displays output in a table format.
* **Format-List**: Displays output as a list of properties.
* **Format-Wide**: Displays output in a wide format, showing only one property per line.



SCRIPTING CONSTRUCT IN POWERSHELL:

In **PowerShell**, **scripting constructs** are the essential elements you use to write scripts that perform tasks.  
**1. Variables**

* Store data values in memory.
* Always start with a $ symbol.
* Example:

$name = "Sohini"

$age = 20

**2. Data Types**

* Define what kind of value a variable holds.
* Common types: String ("text"), Integer (123), Boolean ($true / $false), Array (@(1,2,3)).
* Example:

$price = 99.99 # Decimal

$isAvailable = $true # Boolean

**3. Operators**

* Perform actions like math, comparison, and logic.
* **Arithmetic**: +, -, \*, /
* **Comparison**: -eq (equal), -ne (not equal), -gt (greater than), -lt (less than)
* **Logical**: -and, -or, -not
* Example:

$sum = 5 + 3

if ($sum -eq 8) { "Correct!" }

**4. Conditional Statements**

* Control script flow based on conditions.
* Example:

if ($age -ge 18) {

"Adult"

} elseif ($age -ge 13) {

"Teenager"

} else {

"Child"

}

**5. Loops**

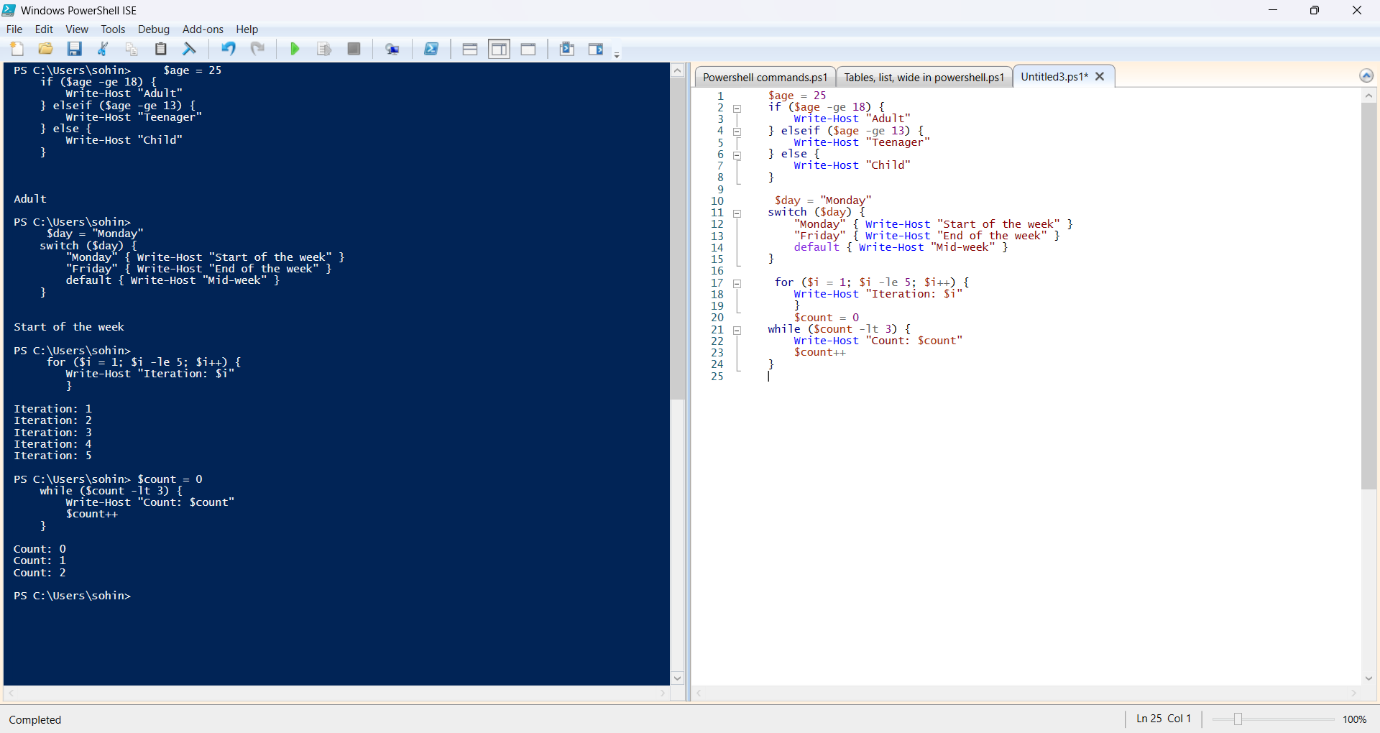
* Repeat actions until a condition is met.

**For Loop**

for ($i = 1; $i -le 5; $i++) {

"Number: $i"

}



POWERSHELL OBJECTS , VARIABLES , ARRAYS:

**1. PowerShell Objects**

* In PowerShell, everything is an object — like a box that holds information plus details about it.
* Each object has:
  + **Properties** → facts about it (like Name, Size, Status).
  + **Methods** → actions it can do (like Start, Stop, Delete).

Example:

Get-Process

* Each process is an **object** with properties (CPU, Memory) and methods (Kill).

**2. Variables**

* A variable is a named container that stores data.
* Always starts with $ in PowerShell.

Example:

$Name = "Sohini"

$Age = 18

**3. Arrays**

* An array is a collection of values stored in one variable.
* Let’s us keep multiple items together.

Example:

$Fruits = @("Apple", "Mango", "Banana")

* $Fruits[0] → Apple
* $Fruits[1] → Mango

MODULARIZATION:

Modularization in PowerShell means breaking code into smaller, reusable parts called modules instead of keeping everything in one big script.

**TYPES OF MODULES:**

**Script Module (.psm1):**

A PowerShell script file containing functions, variables, etc. Think of it like: A notebook where you’ve written your favorite recipes — you can open it anytime and follow them.

**Manifest Module (.psd1):**

A metadata file describing a module — lists its version, author, what files it includes, etc.

Think of it like: A catalog or index of your toolbox so others know what’s inside and how to use it.

**Manifest Module (.psd1):**

What it is: A metadata file describing a module — lists its version, author, what files it includes, etc. Think of it like: A catalog or index of your toolbox so others know what’s inside and how to use it.

**Using a Module:**

Import the module: Use the Import-Module cmdlet to load the module into the current session.

Use the module's functions: Once imported, you can call the functions defined within the module.