***Name:- Raushan Kumar Sharma***

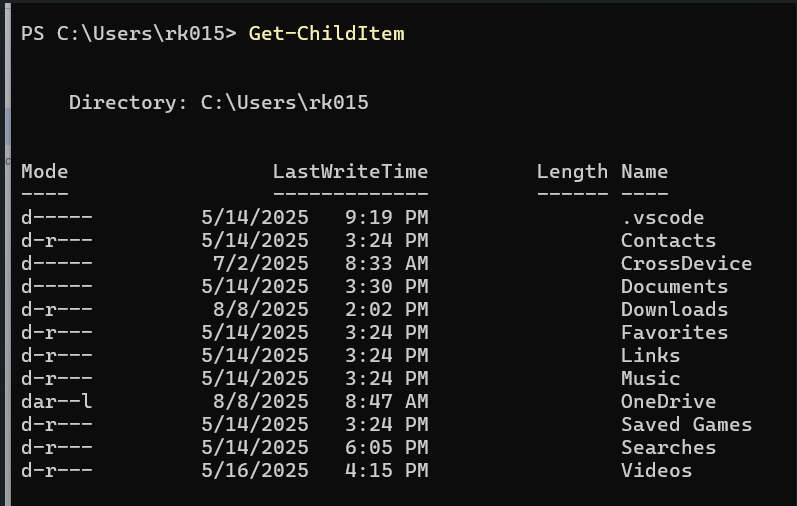
***User I’D:- 34745***

***E-mail I’D:-*** [***raushankumarsharma01582000@gmail.com***](http://raushankumarsharma01582000@gmail.com)

*Assignment Topics:-*

* ***Introducing to Cmdlets***
* ***The PowerShell Pipeline***
* ***Key Cmdlets***
* ***WMI & PowerShell***
* ***Pipeline Filtering & Operators***
* ***Input, Output & Formatting***
* ***Scripting Overview***
* ***Introducing to Cmdlets:-***

***Cmdlets are the fundamental building blocks of PowerShell, acting as specialized commands for performing specific tasks. They follow a consistent verb-noun naming convention (e.g., Get-ChildItem) to clearly indicate their function.***



* ***The PowerShell Pipeline:-***

***The PowerShell pipeline is a fundamental concept in PowerShell that enables the chaining of commands, allowing the output of one command to serve as the input for the next. This mechanism facilitates efficient and flexible data processing and task automation.***

***Key characteristics of the PowerShell pipeline:***

***Piping Operator (|):***

The pipeline operator, represented by the vertical bar (|), connects commands in a sequence.

***Object-Oriented Nature:***

Unlike traditional shell pipelines that often deal with text streams, the PowerShell pipeline passes objects.

***Pipeline Input:***

Cmdlets (PowerShell commands) are designed to accept input from the pipeline, either by value (the entire object) or by property name (specific properties of the object).

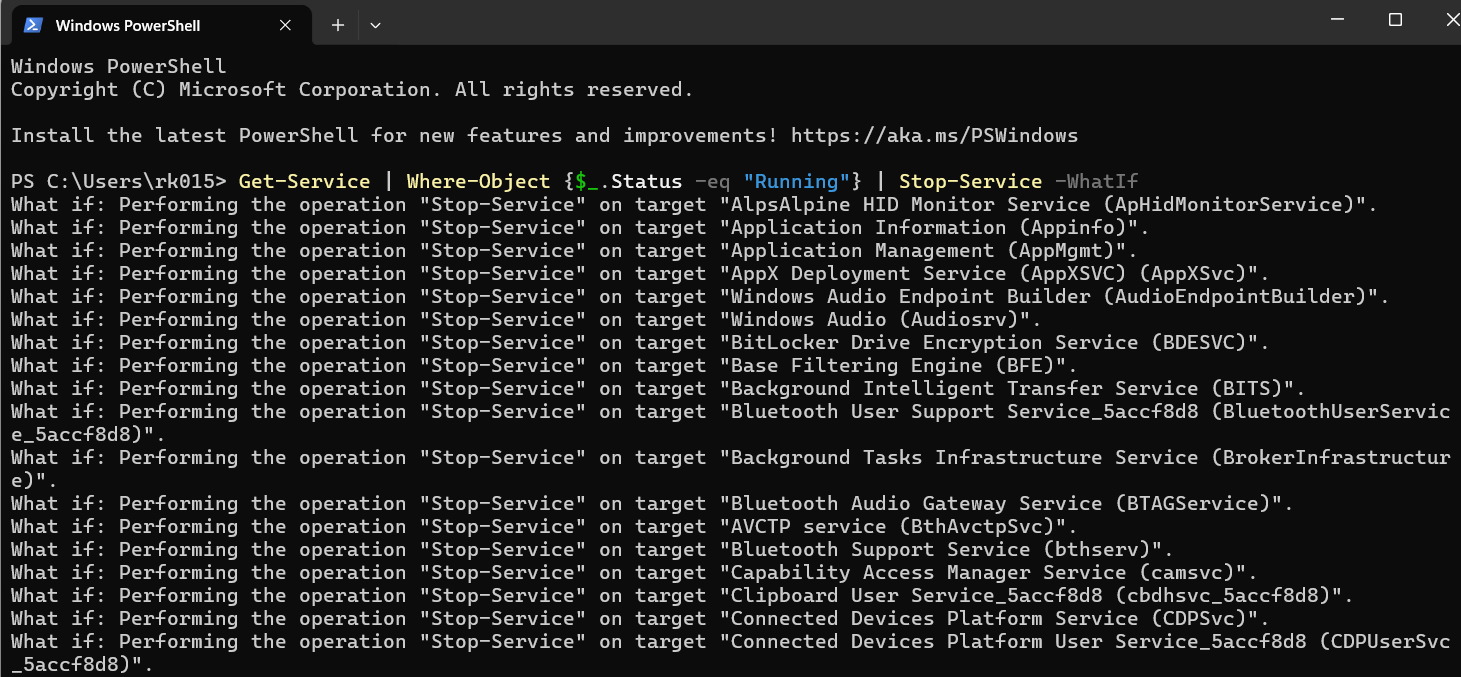
***Efficiency and Readability:***

By eliminating the need for intermediate variables to store command outputs, the pipeline promotes cleaner, more concise, and often more efficient code.

***Chaining Commands:***

Multiple commands can be chained together in a single pipeline to perform complex operations, such as retrieving a list of services, filtering them based on a condition, and then performing an action (e.g., stopping or starting) on the filtered services.

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



* ***Key Cmdlets:-***

Cmdlets are specialized commands in PowerShell, designed for specific tasks and following a Verb-Noun naming convention (e.g., Get-Process). They are the building blocks of PowerShell scripting and automation, enabling users to interact with the system and manage various aspects of the operating environment.

Key Concepts:

Verb-Noun Naming:

Cmdlets follow a structured naming pattern, like Get-Process or Set-Content, making them easily discoverable and understandable.

Specialized Tasks:

Each cmdlet is designed to perform a specific action, such as retrieving information, modifying settings, or managing objects.

PowerShell Environment:

Cmdlets are executed within the PowerShell runtime, which provides the necessary context for their operation.

Identity and Atomicity:

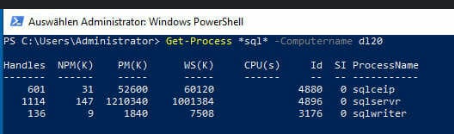
Cmdlets should have an Identity parameter to specify the object they act upon and should operate atomically, meaning they either succeed completely or fail without leaving the system in an inconsistent state.

Extensibility:

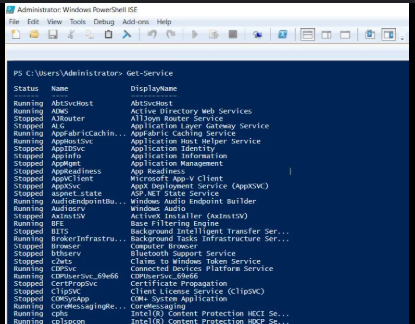
PowerShell cmdlets are designed to be extensible, allowing users to create custom cmdlets or leverage existing ones from modules.

Examples of Cmdlets:

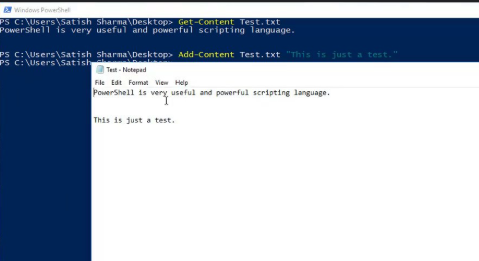
Get-Process: Retrieves a list of running processes.



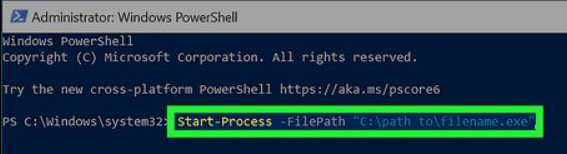
Get-Service: Retrieves a list of services.



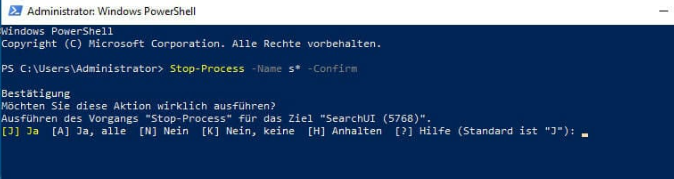
Add-Content: Writes or replaces the content of a file.



Start-Process: Starts a process.



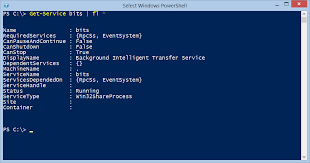
Stop-Process: Stops a running process.



* ***WMI & PowerShell:-***

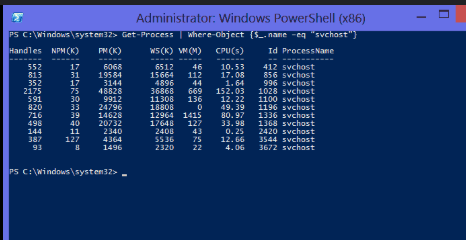
Windows Management Instrumentation (WMI) is a core component of the Windows operating system that provides a standardized way to manage and monitor system components, applications, and networks. It acts as a management infrastructure based on the Common Information Model (CIM) and allows administrators to query system information, configure settings, and invoke methods on various managed resources.

PowerShell, Microsoft's object-oriented shell and scripting language, offers robust integration with WMI, making it an incredibly powerful tool for system administration and automation. PowerShell cmdlets like Get-WmiObject (or its newer equivalent, Get-CimInstance), Invoke-WmiMethod, Set-WmiInstance, and Register-WmiEvent enable direct interaction with WMI.



* ***Pipeline Filtering & Operators:-***

Pipeline filtering and operators are fundamental concepts in data processing and software architecture. They involve structuring a sequence of operations where the output of one stage becomes the input for the next, enabling efficient and modular data manipulation. This concept is applied in various contexts, including data pipelines, software design patterns, and programming languages.

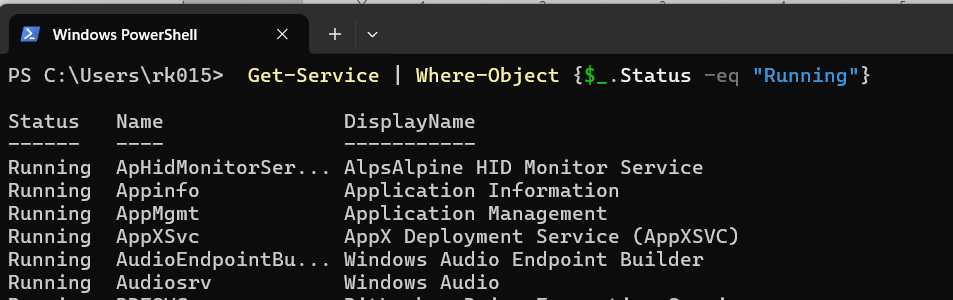


* ***Input, Output & Formatting:-***

Input:

Piping: The most common method, where the output of one cmdlet serves as the input for another using the pipeline operator (|).

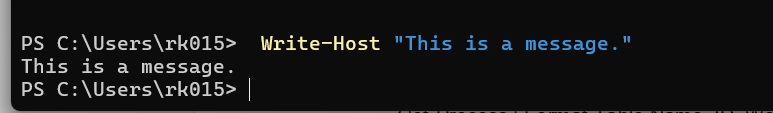
Get-Service | Where-Object {$\_.Status -eq "Running"}



Output:

Write-Host: Displays strings directly to the console. This is primarily for displaying information to the user.

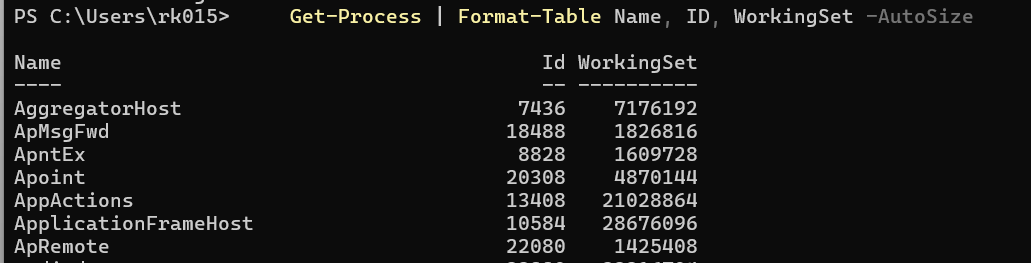
Write-Host "This is a message."



Formatting:

Format-Table: Formats output as a table, allowing selection of specific properties and customization of column layout.

Get-Process | Format-Table Name, ID, WorkingSet -AutoSize



* ***Scripting Overview:-***

PowerShell scripting involves using the PowerShell scripting language to automate tasks and manage systems, primarily within Microsoft Windows environments, but also cross-platform. It combines the functionality of a command-line interface with a robust scripting language built on the .NET framework.

Scripting Language Features:

* PowerShell includes a full-fledged scripting language with support for:
* Variables: Storing and manipulating data.
* Conditional Statements: If-Else, Switch for decision-making.
* Loops: For, ForEach, While for repetitive tasks.
* Functions: Creating reusable blocks of code.
* Error Handling: Try-Catch-Finally blocks for robust script execution.

