PowerShell Scripting: Zero to Hero Guide

Introduction to PowerShell

PowerShell is a task automation framework from Microsoft, consisting of a command-line shell and a scripting language built on .NET. It is mainly used for automating Windows tasks, configuration, and system administration.
It supports cmdlets (small built-in commands), access to .NET libraries, file system, registry, processes, services, and more.

Why PowerShell?

- Native to Windows and supported by Microsoft
- Powerful scripting capabilities
- Integrates deeply with Windows OS
- Automation of tasks like installing software, managing files, configuring network, and more
- Used in DevOps for managing Windows VMs, Azure, and CI/CD pipelines

Variables and Data Types

Variables are created using the $ symbol.

Examples:

```
$name = "Revanth"
$age = 22
```

Common data types:

```
[string]$str = "hello"
[int]$num = 10
[bool]$flag = $true
[float]$pi = 3.14
```

Operators and Expressions

Comparison Operators:

-ge : Greater than or equal to

-le : Less than or equal to

-eq : Equal to

-ne : Not equal to

-gt : Greater than

-lt : Less than

Arithmetic:

+ - * / %

Logical:

-and, -or, -not

Conditional Statements (if-else)

Example:

```
$age = 18
if ($age -ge 18) {
  Write-Output "Adult"
} else {
  Write-Output "Minor"
}
```

Explanation:

- if: starts a conditional block
- -ge: checks if age is greater than or equal to 18
- {}: encloses the code to execute

Loops (for, while)

For Loop:

```
for ($i = 0; $i -lt 5; $i++) {
  Write-Output "i = $i"
}
```

While Loop:

```
$count = 0
while ($count -lt 3) {
  Write-Output $count
  $count++
}
```

Functions

Function Definition:

```
function Greet($name) {
  return "Hello, $name"
}
Greet "Revanth"
```

Explanation:

- function: defines a block of reusable code
- $name: parameter passed to the function

File Operations

Write to File:

```
"Hello World" | Out-File "output.txt"
```

Read from File:

```
Get-Content "output.txt"
```

Explanation:

- Out-File: writes data to a file
- Get-Content: reads data from a file

PowerShell Scripting: Zero to Hero Guide

User Input and Output

Get input:

```
$name = Read-Host "Enter your name"
Write-Output "Hello, $name"
```

Explanation:

- Read-Host: takes input from user
- Write-Output: prints message

System Commands

List Services:

```
Get-Service
```

Stop Process:

```
Stop-Process -Name "notepad" -Force
```

Explanation:

- Get-Service: lists all Windows services
- Stop-Process: stops the specified process

Error Handling

Try-Catch:

```
try {
  Get-Item "missingfile.txt"
} catch {
  Write-Output "File not found!"
}
```

Explanation:

- try: attempts a command

- catch: handles errors

PowerShell Projects (Mini Tasks)

1. Backup Script:

```
Copy-Item -Path "C:\data" -Destination "D:\backup" -Recurse
```

2. User Account Creator:

```
New-LocalUser -Name "testuser" -Password (ConvertTo-SecureString "Pass123" -AsPlainText -Force)
```

3. Service Monitor:

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

4. Startup Script:

Add a script to Task Scheduler to run at boot

Conclusion

PowerShell is a powerful tool for any Windows user or administrator. Learning from variables to real-time automation scripts helps in managing and automating everyday tasks. With practice, you can handle any Windows VM configuration, task scheduling, or DevOps integration easily using PowerShell.