Week 2: Getting Started with PowerShell

Contents

Time required: 60 minutes

Video Walkthrough: Getting Started with PowerShell on Windows Server 2025

Tutorial 1: Hello World

Start Server1. We will create and run our PowerShell scripts on our Server1 VM.

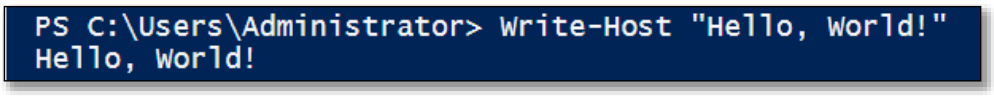
It is traditional when learning any programming language to create a Hello World program. This confirms that everything is working properly.

Write-Host is a command that is used to display information directly to the console or terminal. It's often used to output messages, text, or other data to the screen so that the user can see the information.

1. Click **Start** → Type **Powershell**
2. Click **PowerShell ISE**.
3. Go to the PS prompt in the bottom screen. Type in the Write-Host line.

```
Write-Host "Hello, World!"
```

Example run:



Run Our First Script

1. In PowerShell ISE → **File** → **New**
2. Type the following in the file.

```
Write-Host "Hello, World!"
```

3. File → Save As → Name the file **HelloWorld**
   PowerShell will automatically add the .ps1 extension.
4. Press **F5** to run the script.

Example Run:

```
PS C:\Users\Administrator> C:\Users\Administrator\Documents\HelloWorld.ps1
Hello, World!
```

PowerShell Version

Let's discover the version of PowerShell we are using.

```
# Find out the PowerShell version
$PSVersionTable
```

You should see something like this.

```
PS C:\Windows\system32> $PSVersionTable

Name                           Value
----                           -----
PSVersion                      5.1.19041.1682
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.19041.1682
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
```

Tutorial 2: User Input and Output

Read-Host is a command that allows you to interactively prompt the user to enter input from the keyboard while the script is running. It's used to gather data or information directly from the user.

Create a PowerShell script named **input_output.ps1**

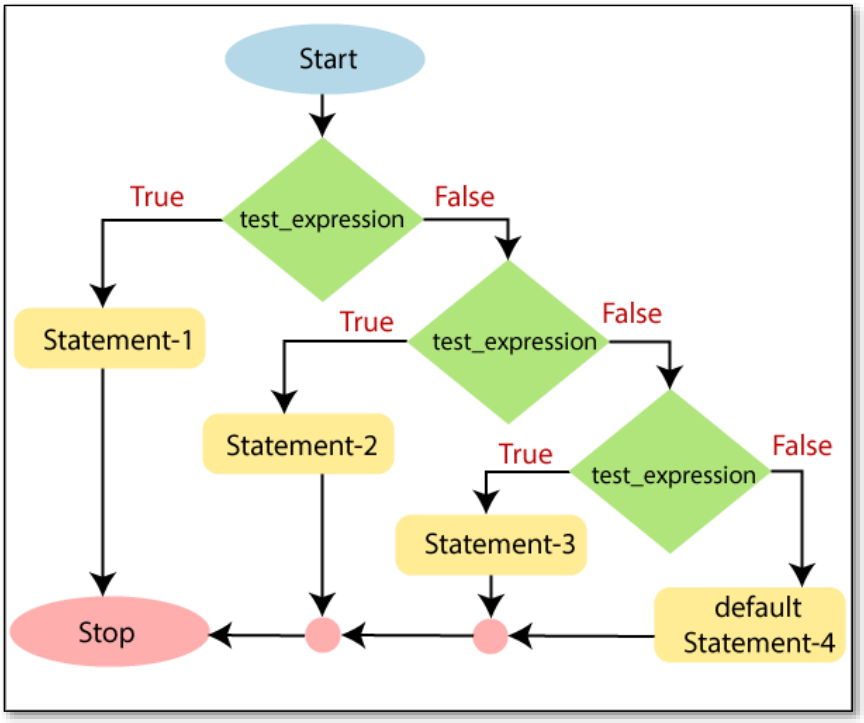
```powershell
#    Name: input_output.ps1
#    Author:
#    Created: 08/26/23
#    Purpose:

# Prompt the user for their name
# Variables in PowerShell start with $
# Assign user input to the variable $name
$name = Read-Host "Please enter your name:"

# Display a personalized greeting using the $name variable
Write-Host "Hello, $name! Welcome to PowerShell scripting."
```

Example run:

```
PS C:\Users\Administrator> C:\Users\Administrator\Documents\input_output.ps1
Please enter your name:: Bill
Hello, Bill! Welcome to PowerShell scripting.
```

Tutorial 3: Conditional Statements

i**f** statements are used for conditional branching in your scripts. They allow you to make decisions and execute different code blocks based on whether a certain condition is true or false. An **if** statement can be followed by one or more optional **elseif** blocks and an optional **else** block, providing a way to handle various scenarios.




```
if (condition) {
    # Code to execute if the condition is true
}
elseif (condition) {
    # Code to execute if the first condition is false and this condition is
true
}
else {
    # Code to execute if none of the previous conditions are true
}
```

Example conditional script

```
[int]$age = Read-Host "Enter your age:"
# -lt is less than
if ($age -lt 18) {
    Write-Host "You are underage."
}
# -ge is greater than or equal to
elseif ($age -ge 18 -and $age -lt 65) {
    Write-Host "You are an adult."
}
else {
    Write-Host "You are a senior citizen."
}
```

In this example, the script prompts the user to enter their age using Read-Host. [int] converts the string input into an integer. The if statement then checks the entered age against different conditions:

- If the age is less than 18, the script displays "You are underage."
- If the age is 18 or greater and less than 65, the script displays "You are an adult."
- If the age is 65 or greater, the script displays "You are a senior citizen."

Create a PowerShell script named **positive_negative.ps1**

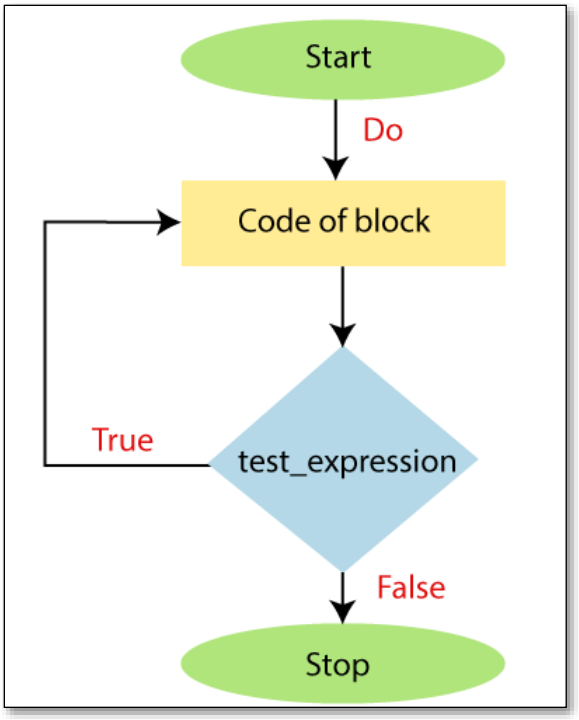
```powershell
#   Name: if_positive_negative.ps1
#   Author:
#   Created: 08/26/23
#   Purpose:

# Ask the user to type in a number
# Read input into a variable
[int]$number = Read-Host "Enter a number"

# Test if the number entered is greater than 0
# -gt tests greater than
if ($number -gt 0) {
    Write-Host "The number is positive."
}
# If the number is not greater than 0, check if it's less than 0
# -lt tests for less than
elseif ($number -lt 0) {
    Write-Host "The number is negative."
}
# If the number is not greater than 0 and not less than 0, it must be 0
else {
    Write-Host "The number is zero."
}
```

Example run:

```
PS C:\Users\Administrator> C:\Users\Administrator\Documents\positive_negative.ps1
Enter a number:: -5
The number is positive.
```

Tutorial 4: For Loop

A **for** loop is a control structure that allows you to repeat a block of code a specific number of times. It's especially useful when you want to iterate through a sequence of values or perform an action a known number of times.



The basic structure of a **for** loop in PowerShell:

```
for (initialization; condition; iteration) {
    # Code to execute in each iteration
}
```

Let's break down each part of the `for` loop:

Initialization: This is where you set up the initial value of a loop control variable. This variable will be used to keep track of the loop's progress.

Condition: This is a condition that is evaluated before each iteration of the loop. As long as the condition is true, the loop continues to execute. If the condition becomes false, the loop stops.

Iteration: This is where you specify how the loop control variable is updated after each iteration. It's often used to increment or decrement the variable.

Here's an example of a simple **for** loop that counts from 1 to 5 and displays the current value:

```
for ($i = 1; $i -le 5; $i++) {
    Write-Host "Current value: $i"
}
```

In this example:

($i = 1; initializes the loop control variable `$i` to start at 1.

$i -le 5; is the condition. If `$i` is less than or equal to 5, the loop continues.

$i++) is the iteration part. After each iteration, the value of `$i` is incremented by 1.

The loop will run 5 times, displaying "Current value: 1" through "Current value: 5" in the console.

Create a PowerShell script named **for_loop.ps1**

```
#   Name: for_loop.ps1
#   Author:
#   Created: 08/26/23
#   Purpose:

# Ask the user to type in a number
# Read input into a variable
[int]$number = Read-Host "Enter a number:"

# Displaying numbers from 1 to $number using a loop
for ($counter = 1; $counter -le $number; $counter++) {
    Write-Host $counter
}
```

Example run:

```
PS C:\Users\Administrator> C:\Users\Administrator\Documents\for_loop.ps1
Enter a number:: 10
1
2
3
4
5
6
7
8
9
10
```

Assignment Submission

1. Attach the program files.
2. Attach screenshots showing the successful operation of the programs.
3. Submit in Blackboard.