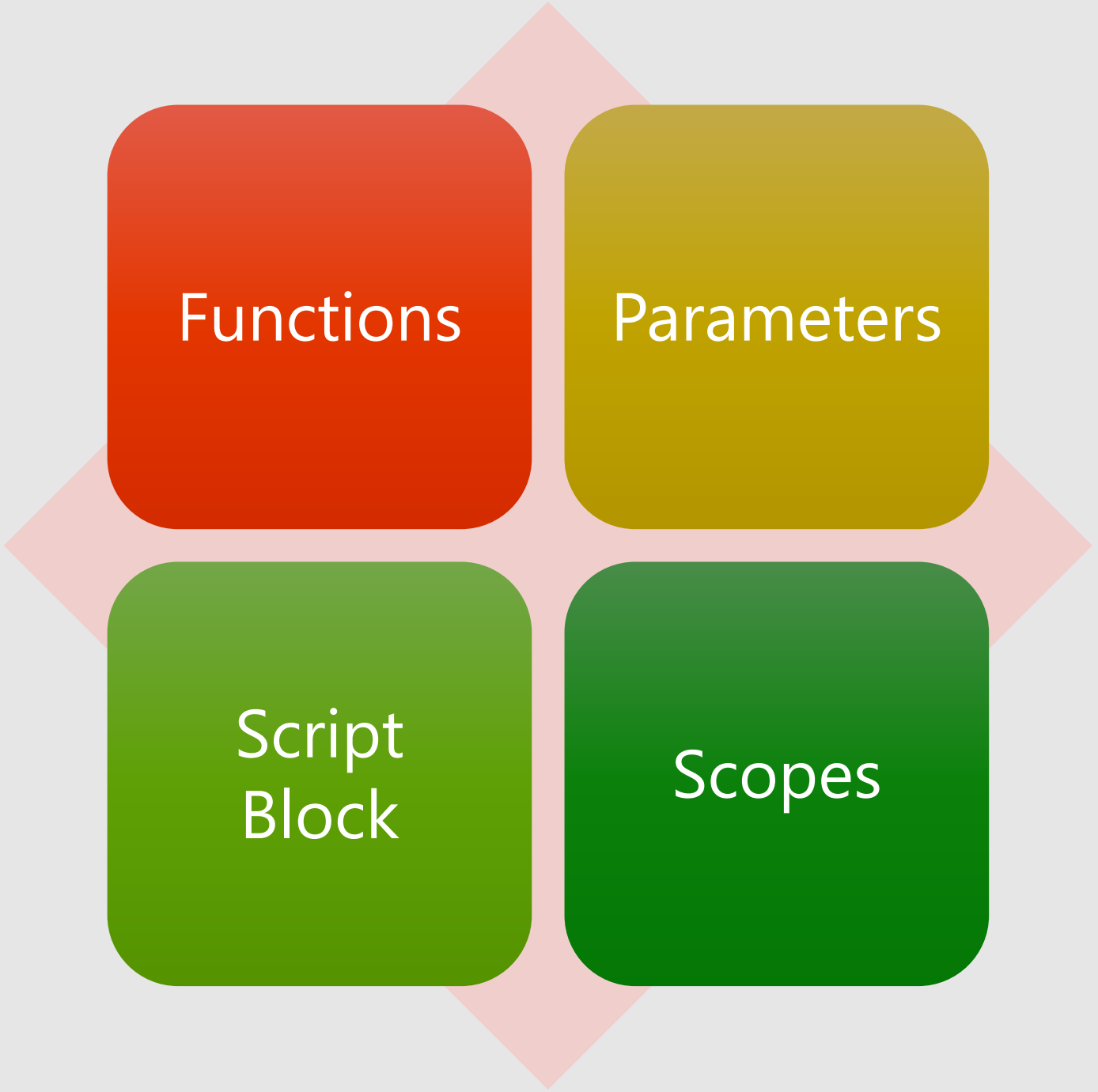
Microsoft

Introduction to Functions

Module 5

Learning Units

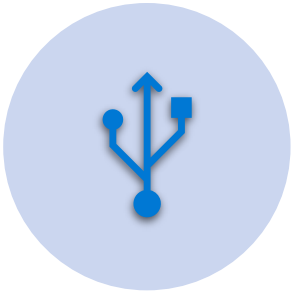


Functions

Function Overview

Reusable block of PowerShell code

Reduces size of code and increases reliability

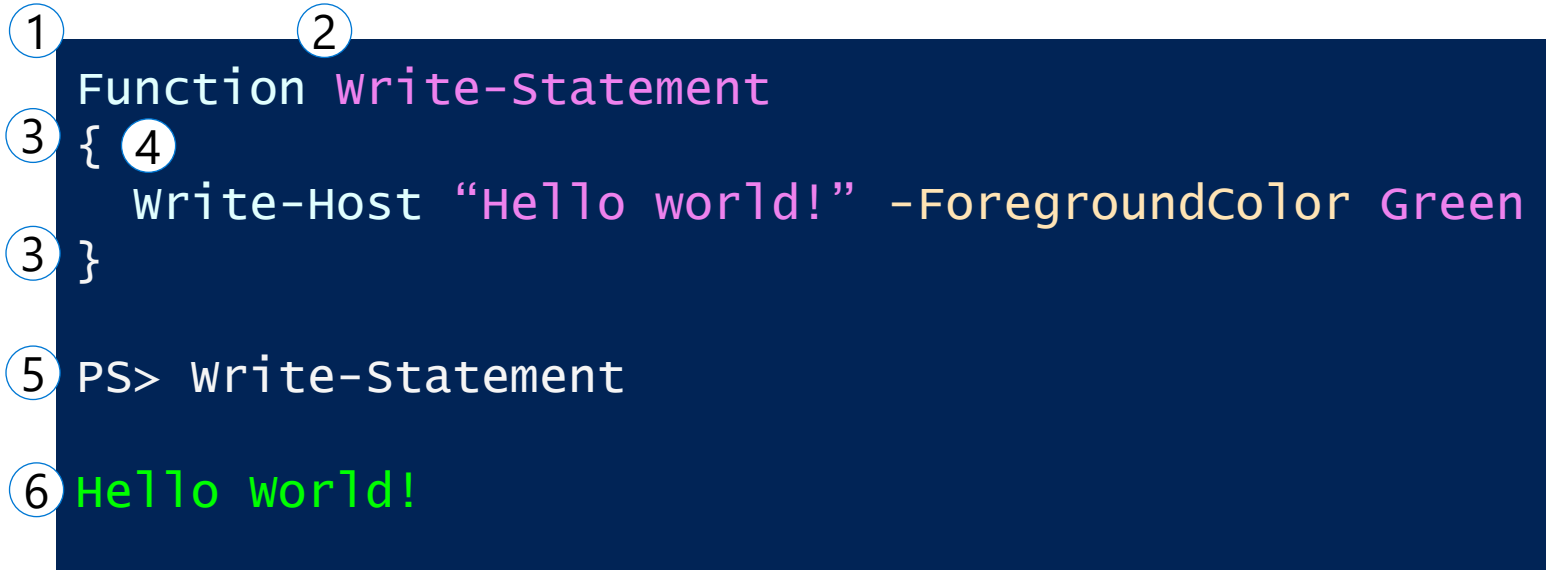


Can accept **parameter** values and return output

Advanced Functions behave like **Cmdlets**, including help content

What Does a Function Look Like?

1. Function Keyword
2. Function Name
3. Curly Brace Pair
4. PowerShell Commands
5. Call the Function
6. Function output



Creating a Function

Multiple commands can be contained within a function

Allows for large code blocks to be reused – no need to copy and paste

Maintains consistency between repeated uses of code when edited

```
Function Write-ServiceStatus
{
  $SVC = Get-Service -Name WinRM
  $Name = $SVC.DisplayName
  $Status = $SVC.Status
  Write-Host "The Service $Name is currently $Status" -ForegroundColor Green
}

PS> Write-ServiceStatus
The Service Windows Remote Management (WS-Management) is currently Stopped
```

Demonstration

Simple Function

Parameters

Parameters in a function

- Must be the **first line** of code in the function
- Defined using **param ( )** statement
- When used, the parameter name is preceded by a **hyphen**
- By default, accepts **any** data type, is **positional**, and is **optional**
- **Optional** advanced attributes can override default parameter behavior

Add Parameters to a Function

1. Function Keyword
2. Function Name
3. Curly Brace Pair
4. PowerShell Commands
5. Call the Function
6. Function output

```
Function Write-Statement
{
  Write-Host "Hello World!" -ForegroundColor Green
}

PS> Write-Statement

Hello World!
```

Add Parameters to a Function

1. Function Keyword
2. Function Name
3. Curly Brace Pair
4. Parameter statement
5. PowerShell Commands
6. Call the Function
7. Function output

```
Function Write-Statement
{
  Param($Statement)
  Write-Host $Statement -ForegroundColor Green
}

PS> Write-Statement -Statement "Hello World!"

Hello World!
```

Default Parameter Values

Can assign a value like any other variable

Function parameters will override the default value

```
Function Write-Statement
{
  Param($Statement = "Good morning!")
  Write-Host $Statement -ForegroundColor Green
}

PS> Write-Statement

Good morning!
```

Adding Multiple Parameters

Functions can accept multiple parameters separated by commas

Supports line breaks between parameters

```
Function Write-ServiceStatus
{
  Param ($Service,
         $Color = "Green")
  $SVC = Get-Service -Name $Service
  $Name = $SVC.DisplayName
  $Status = $SVC.Status
  Write-Host "The Service $Name is currently $Status" -ForegroundColor $Color
}

PS> Write-ServiceStatus  -Service WinRM -Color Yellow
The Service Windows Remote Management (WS-Management) is currently Stopped
```

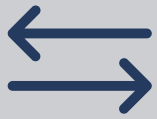
Demonstration: Parameters in functions

Script Blocks

What is a Script Block?

A collection of statements listed in curly brackets “{ }”

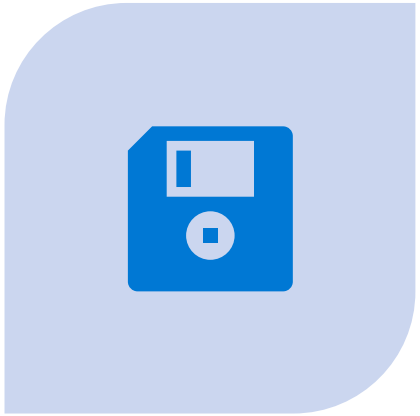
Used by Cmdlets, Functions, Automation, and other advanced features



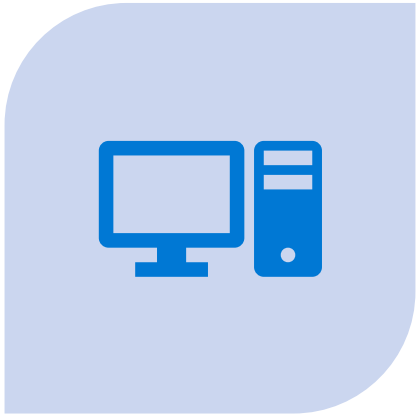
Can accept parameter values and return output

* Script blocks will continue to be used throughout this course

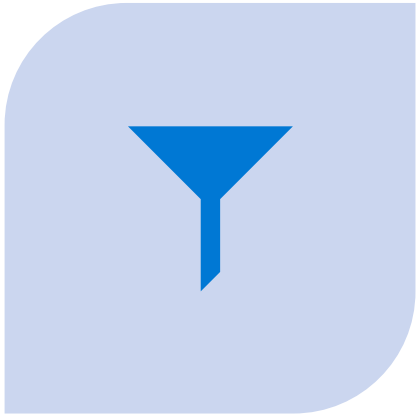
When to a use script blocks



Save code for reuse with functions and automation



Remotely sending commands to another machine



Complex queries and filters on the pipeline

Using Script Blocks

Measure-Command uses the Expression parameter

- Expression parameter accepts the script block

Measures the time it takes commands to run

- Also executes the commands

Many other cmdlets have similar parameters that use Script Blocks

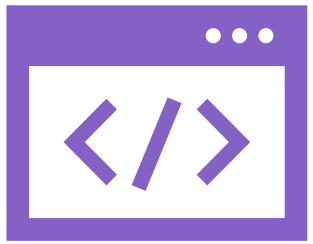
- Identify other commands with: *Get-Command –ParameterType ScriptBlock*

```
PS> Measure-Command -Expression {Get-Process}

Days              : 0
Minutes           : 0
Seconds           : 2
Milliseconds      : 933
```

Script Blocks

Script Blocks can be saved in variables

```
PS> $ScriptBlock = {Get-Service -Name WinRM}
PS> $ScriptBlock

Get-Service -Name WinRM
```



Script block code is contained but not executed

Invoking Script Blocks

Using a Cmdlet: **Invoke-Command –ScriptBlock $ScriptBlock**

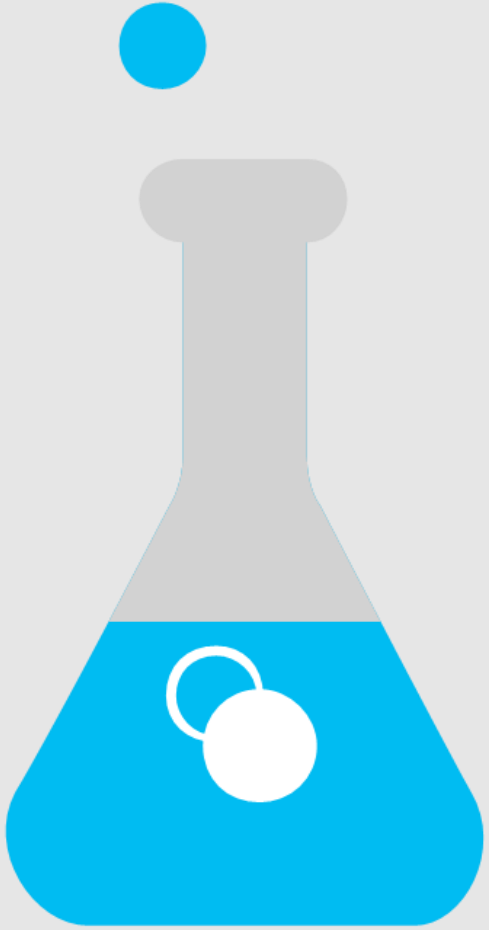
Using a Method: **$ScriptBlock.Invoke()**

```
PS> $ScriptBlock = { Get-Service -Name WinRM }
PS> Invoke-Command -ScriptBlock $ScriptBlock

Status   Name   DisplayName
------   ----   -----------
Stopped  WinRM  Windows Remote Management (WS-Management)

PS> $ScriptBlock.Invoke()

Status   Name   DisplayName
------   ----   -----------
Stopped  WinRM  Windows Remote Management (WS-Management)
```

Demonstration: Script Blocks

LAB 4:
Functions Basic

60 Minutes



LAB