Commands

Module 2

Learnings covered in this Unit

- External Commands
- PowerShell Command
- Cmdlet Syntax
- Risk Mitigation
- Aliases

External Commands

External Commands VS Native PowerShell

EXTERNAL COMMAND	POWERSHELL EQUIVALENT
Ping Loopback –a –n 1	Test-Connection –count 1
Shutdown /m \\MS	Restart-Computer –ComputerName “MS”
Netsh interface ip show config	Get-NetIPAddress

- **CMD commands** accessible from PowerShell
- Output of the commands is in **plain text**
- Run in a **separate process**
- No standardization of **naming** or **syntax** conventions

Demonstration

External Command

vs.

Native Commands

PowerShell Native Commands

Cmdlets

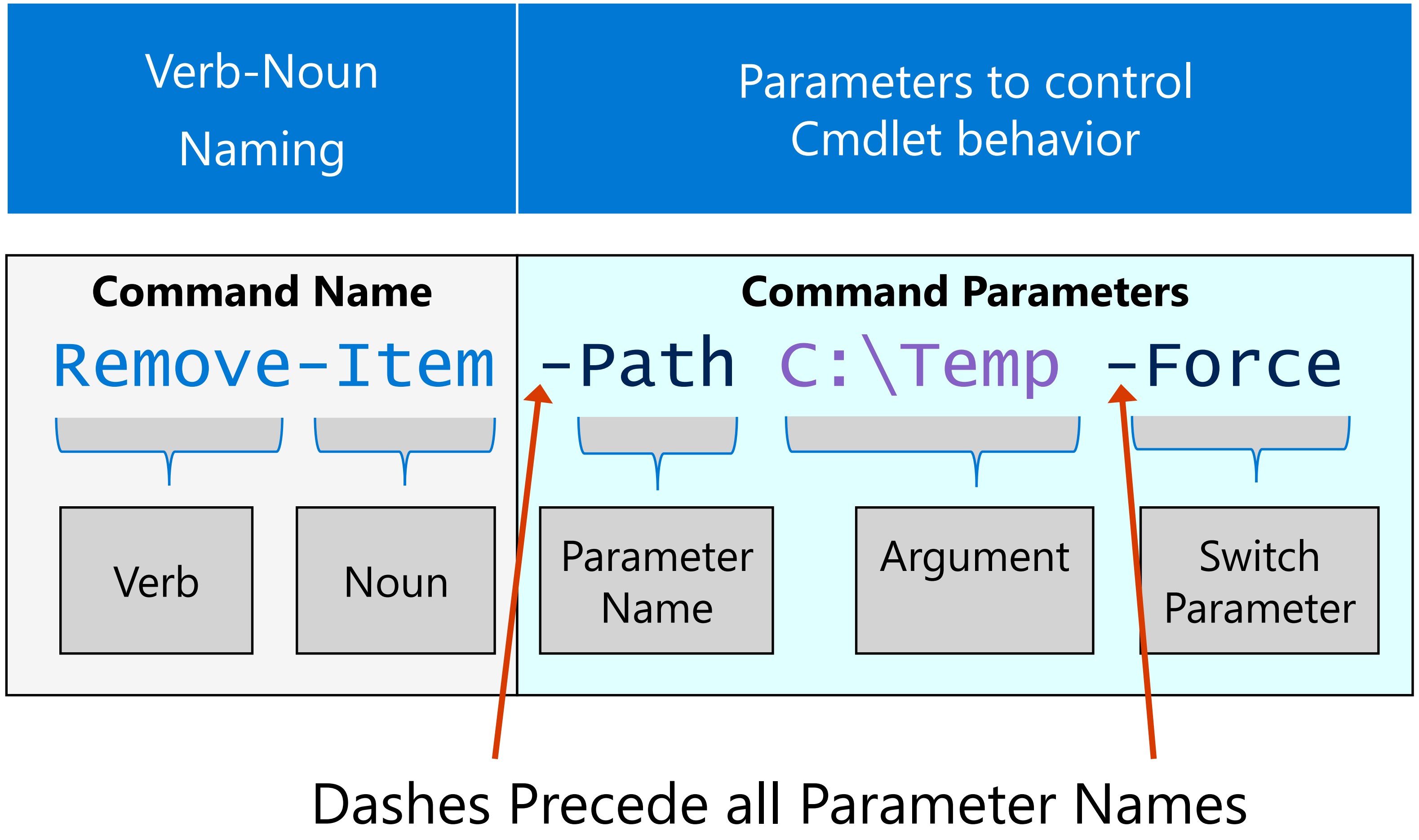
Native PowerShell commands

Standardized naming convention and syntax

Easy to **discover**

Returns **rich data** objects

Cmdlet Standardization


Verb-Noun Naming
Parameters to control Cmdlet behavior
Command Name
Command Parameters
Remove-Item -Path C:\Temp -Force
Verb
Noun
Parameter Name
Argument
Switch Parameter
Dashes Precede all Parameter Names

Cmdlet Examples

```
PS C:\> Get-Process

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
     83       7     1084       4124    45     0.09   7784 armsvc
    179      13     1892       8216    89     0.66   6540 BDAppHost
    143      12     1840       7320    76     0.22  11148 BDExtHost
...

PS C:\> Restart-Service -Name Spooler

PS C:\> Test-Connection -ComputerName 2012R2-MS -Count 1 -Quiet
True

PS C:\> Get-Service -Name Bits

Status   Name DisplayName
------   ---- -----------
Running bits Background Intelligent Transfer Service
```

Get-Command

Discover commands (cmdlets, functions, scripts, aliases, external)

Can show command **syntax**

Many **search options** available (name, verb, noun, module)

Get-Command Examples

```
PS C:\> Get-Command

CommandType     Name                 Definition
-----------     ----                 ----------
Cmdlet          Add-Content          Add-Content [-Path] String[]...
Cmdlet          Add-History          Add-History [[-InputObject] ...
Function        Clear-Host           $space = New-Object System.A...
Alias           exsn                 Exit-PSSession
```

Wildcard in Name

```
PS C:\> Get-Command -Name *user*

CommandType     Name
-----------     ----
Cmdlet          Add-SPOUser
Cmdlet          New-LocalUser
Function        New-UserAdd
Application     quser.exe
Alias           Set-ADUserSettingGAL
```

Get-Command Examples

List cmdlets filtering on Verb

```
PS C:\> Get-Command -Verb Get

CommandType     Name
-----------     ----
Alias           Get-ADUserSnapshot
Alias           Get-AppPackage
Alias           Get-MyIP
Function        Get-DiskImage
Function        Get-DnsClient
Function        Get-DnsClientCache
Cmdlet          Get-Culture
Cmdlet          Get-DAPolicy
Cmdlet          Get-Date
Cmdlet          Get-Delivery
Cmdlet          Get-Event
```

List cmdlets filtering on Noun

```
PS C:\> Get-Command -Noun Service

CommandType     Name
-----------     ----
Cmdlet          Get-Service
Cmdlet          New-Service
Cmdlet          Restart-Service
Cmdlet          Resume-Service
Cmdlet          Set-Service
Cmdlet          Start-Service
Cmdlet          Stop-Service
Cmdlet          Suspend-Service
```

Get-Command Examples

List Cmdlets Only

```
PS C:\> Get-Command -CommandType Cmdlet

CommandType        Name                                  Source
-----------        ----                                  ----------
Cmdlet             Add-ADCentralAccessPolicyMember       ActiveDirectory
Cmdlet             Add-ADComputerServiceAccount          ActiveDirectory
Cmdlet             Add-TeamUser                          MicrosoftTeams
Cmdlet             Add-WindowsImage                      Dism
```

Single Command by name

```
PS C:\> Get-Command -Name Get-Service

CommandType       Name              Version        Source
-----------       ----              -------        ------
Cmdlet            Get-Service       3.1.0.0        Microsoft.Powe…
```

Searching for External Commands

External commands can be found using “**Get-Command**”

```
PS C:\> Get-Command -CommandType 'Application'

CommandType      Name                 Version               Source
-----------      ----                 -------               ------
Application      _MSRSTRT.EXE         0.0.0.0               C:\windows...
Application      AgentService.exe     10.0.19577.1000       C:\windows...
Application      aitstatic.exe        10.0.19577.1000       C:\windows...
Application      alg.exe              10.0.19577.1000       C:\windows...
Application      appidtel.exe         10.0.19577.1000       C:\windows...
Application      AppVClient.exe       10.0.19577.1000       C:\windows...
Application      appverif.exe         10.0.17763.132        C:\windows...
Application      AppVNice.exe         10.0.19577.1000       C:\windows...
Application      AppVShNotify.exe     10.0.19577.1000       C:\windows...
Application      ARP.EXE              10.0.19577.1000       C:\windows...
...
```

Demonstration

Get-Command

PowerShell Cmdlet Syntax

PowerShell Cmdlet Syntax

How to **use** a command

What parameters are **required**

What parameters are **available**

What **data** is expected

Cmdlet Syntax Legend

Verb-Noun	Command name
-<ParameterName>	Required parameter name
<ArgumentType>	Required argument (value)
[-< > < >]	Optional parameter name and argument
[-< >] <ArgumentType>	Required value, Parameter name is optional
<ArgumentType [ ] >	The argument is a list ‘[ ]’ of a type

Command Name

Syntax Definition

```
<Command-Name> -RequiredParameterName <Required Data>
[-OptionalParameterName> <Optional Data>]
[-SwitchParameters]
[-OptionalParameterName>] <Required Data>
```

Syntax Sample

```
PS C:\>  Get-Command -Name Remove-Item -Syntax

Remove-Item -LiteralPath <string[]>
[-Filter <string>]
[-Include <string[]>]
[-Exclude <string[]>]
[-Recurse]
[-Force]
[-Credential <pscredential>]
[-WhatIf]
[-Confirm]
[-UseTransaction]
[-Stream <string[]>]
```

Command Name

Syntax Definition

```
<Command-Name> -RequiredParameterName <Required Data>
[-OptionalParameterName> <Optional Data>]
[-SwitchParameters]
[-OptionalParameterName>] <Required Data>
```

Syntax Sample

```
PS C:\>  Get-Command -Name Remove-Item -Syntax

Remove-Item -LiteralPath <string[]>
[-Filter <string>]
[-Include <string[]>]
[-Exclude <string[]>]
[-Recurse]
[-Force]
[-Credential <pscredential>]
[-WhatIf]
[-Confirm]
[-UseTransaction]
[-Stream <string[]>]
```

Command Name

Syntax Definition

```
<Command-Name> -RequiredParameterName <Required Data>
[-OptionalParameterName> <Optional Data>]
[-SwitchParameters]
[-OptionalParameterName>] <Required Data>
```

Syntax Sample

```
PS C:\>  Get-Command -Name Remove-Item -Syntax

Remove-Item -LiteralPath <string[]>
[-Filter <string>]
[-Include <string[]>]
[-Exclude <string[]>]
[-Recurse]
[-Force]
[-Credential <pscredential>]
[-WhatIf]
[-Confirm]
[-UseTransaction]
[-Stream <string[]>]
```

Command Name

Syntax Definition

```
<Command-Name> -RequiredParameterName <Required Data>
[-OptionalParameterName> <Optional Data>]
[-SwitchParameters]
[-OptionalParameterName>] <Required Data>
```

Syntax Sample

```
PS C:\>  Get-Command –Name Remove-Item –Syntax

Remove-Item -LiteralPath <string[]>
[-Filter <string>]
[-Include <string[]>]
[-Exclude <string[]>]
[-Recurse]
[-Force]
[-Credential <pscredential>]
[-WhatIf]
[-Confirm]
[-UseTransaction]
[-Stream <string[]>]
```

Command Name

Syntax Definition

```
<Command-Name> -RequiredParameterName <Required Data>
[-OptionalParameterName> <Optional Data>]
[-SwitchParameters]
[-OptionalParameterName>] <Required Data>
```

Syntax Sample

```
PS C:\>  Get-Command -Name Get-Content -Syntax

Get-Content [-Path] <string[]>
[-ReadCount <long>]
[-TotalCount <long>]
…
```

This enables a shortcut called a “Positional Parameter”

Demonstration

Native Command Syntax

Parameter Sets
Positional Parameters

Positional Parameters

Positional parameters allow for **shortcuts** by skipping –ParameterName

Many cmdlets have 1-2 positional parameters

Always shown **first** in syntax

Values provided in **order** listed if more than one is present

```
PS C:\> Get-Command –Name Stop-Process –Syntax

Stop-Process [-Id] <int[]> [-PassThru] [-Force] [-WhatIf] [-Confirm]
```

```
PS C:\> Stop-Process –Id 8660
```

vs

```
PS C:\> Stop-Process 8660
```

Parameter Sets

Multiple **options** to run the same cmdlet with **different** data

PowerShell **decides** which set to used based on the **parameters** passed in

Intellisense only shows options for that set

```
PS C:\> Get-Command -Name Stop-Process -Syntax

Stop-Process [-Id] <int[]> [-PassThru] [-Force] [-WhatIf] [-Confirm]
[<CommonParameters>]

Stop-Process -Name <string[]> [-PassThru] [-Force] [-WhatIf] [-Confirm]
[<CommonParameters>]

Stop-Process [-InputObject] <Process[]> [-PassThru] [-Force] [-WhatIf]
[-Confirm] [<CommonParameters>]
```

Demonstration

Parameter Sets & Positional vs. Named Parameters

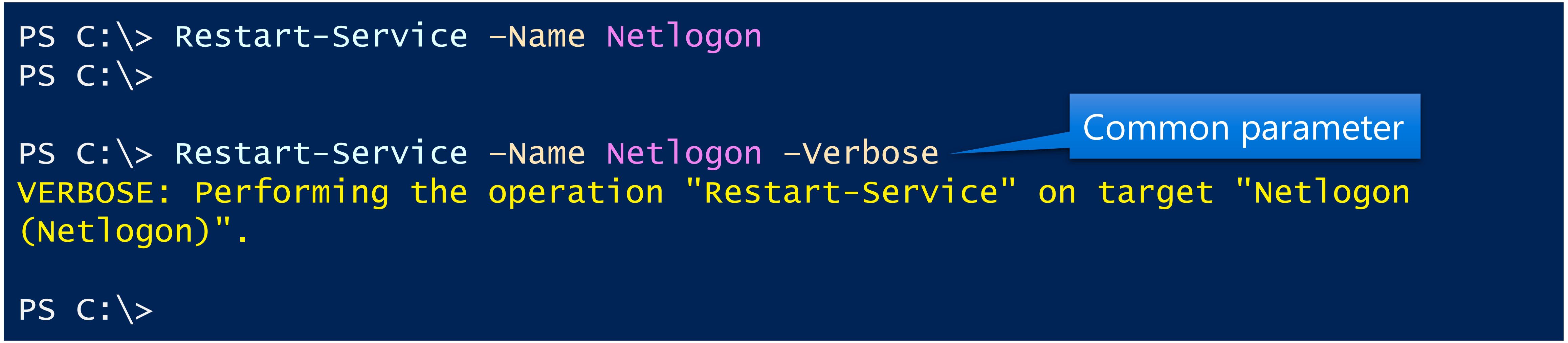
Common Parameters for commands

Cmdlet Common Parameters

- Parameters available on **every** cmdlet
- Implemented by **PowerShell** not cmdlet developer
- Many used to **override** system defaults
- May **not** have an effect even though it exists

Frequently Used Common Parameters

Parameter	Description
-Debug	**Displays** programmer-level detail, if available
-ErrorAction	**Changes** how cmdlet responds to errors
-ErrorVariable	**Stores** error messages in a specified variable
-Verbose	**Displays** detailed information, if available
-WarningAction	**Changes** how cmdlet responds to warnings
-WarningVariable	**Stores** warnings in a specified variable

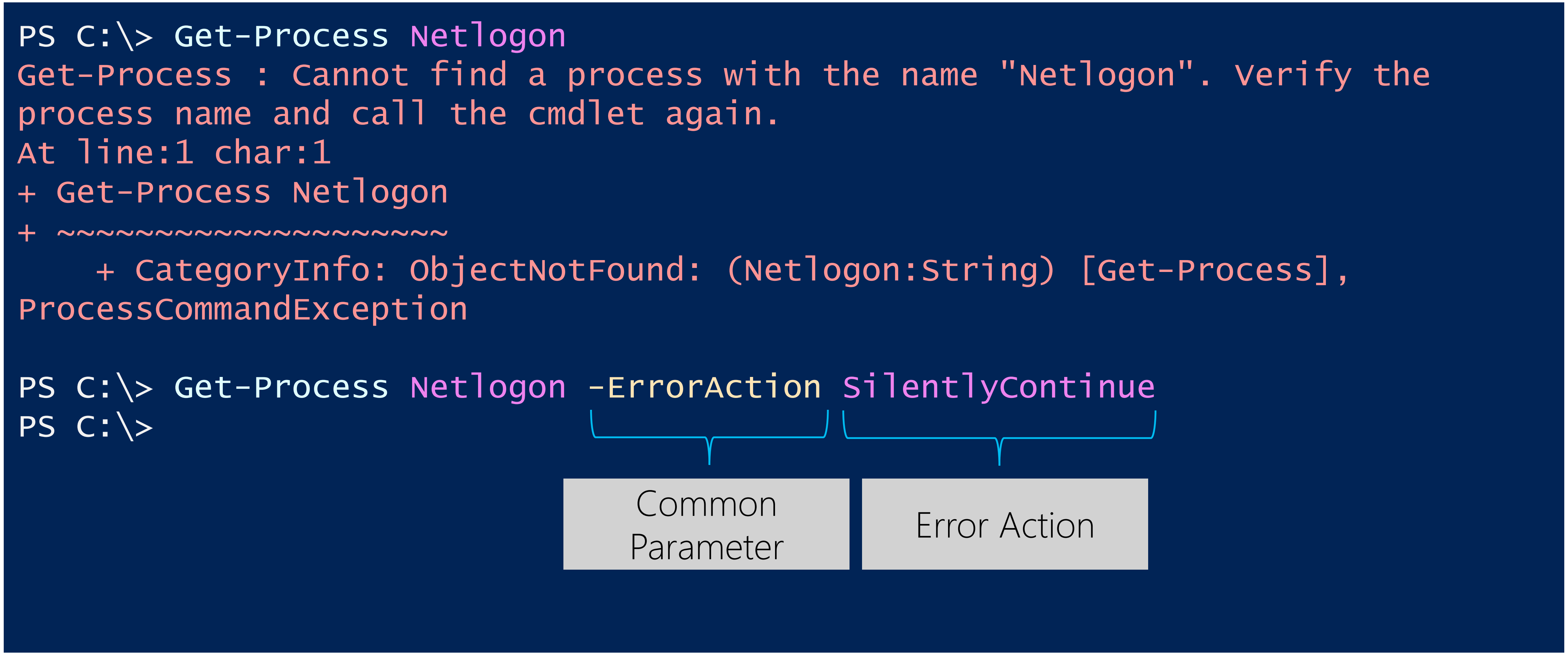


```
PS C:\> Restart-Service -Name Netlogon
PS C:\>

PS C:\> Restart-Service -Name Netlogon -Verbose
VERBOSE: Performing the operation "Restart-Service" on target "Netlogon
(Netlogon)".

PS C:\>
```

Common Parameters - ErrorAction



```
PS C:\> Get-Process Netlogon
Get-Process : Cannot find a process with the name "Netlogon". Verify the
process name and call the cmdlet again.
At line:1 char:1
+ Get-Process Netlogon
+ ~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo: ObjectNotFound: (Netlogon:String) [Get-Process],
ProcessCommandException

PS C:\> Get-Process Netlogon -ErrorAction SilentlyContinue
PS C:\>
```

Demonstration

Common Parameters

Risk Mitigation Parameters

Allows **testing** a cmdlet that deletes or changes data

Prevent **accidental** deletion of data

Add **confirmation** prompts

Available on **many** cmdlets, but not all

Parameter	Description
-WhatIf	**Displays** message describing the command, instead of executing the command
-Confirm	**Prompts** for confirmation before executing command

-WhatIf and Confirm Parameters in Action

```
PS C:\> Stop-Process -Name * -WhatIf

What if: Performing the operation "Stop-Process" on target "AcroRd32 (8160)".
What if: Performing the operation "Stop-Process" on target "AcroRd32 (12756)".
What if: Performing the operation "Stop-Process" on target "armsvc (2468)".
What if: Performing the operation "Stop-Process" on target "atieclxx (3220)".
What if: Performing the operation "Stop-Process" on target "atiesrxx (780)".
...

PS C:\> Start-Service –name xb* -Confirm

Confirm
Are you sure you want to perform this action?
Performing the operation "Start-Service" on target "Xbox Live Auth Manager
(XblAuthManager)".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default
is "Y"):
```

Demonstration

Risk Mitigation Parameters

Command Termination and Line Continuation

Termination Characters

- Used to complete a command – the default is **new line**
- **Semi-colon** used to execute more than one statement on a single line
- Implemented as cross language learning **compatibility**
- Best practice - Only use a **semi-colon** on the **command line** not in scripts

Example - Termination Character

```
PS C:\> Get-Service BITS; Get-Process System; Get-service wsearch


Status   Name               DisplayName
------   ----               -----------
Running  BITS               Background Intelligent Transfer Ser...

Id      : 4
Handles : 1308
CPU     : 1213.59375
Name    : System


Running  wsearch            Windows Search
```

Line Continuation

Allows writing multiline code in the **console**

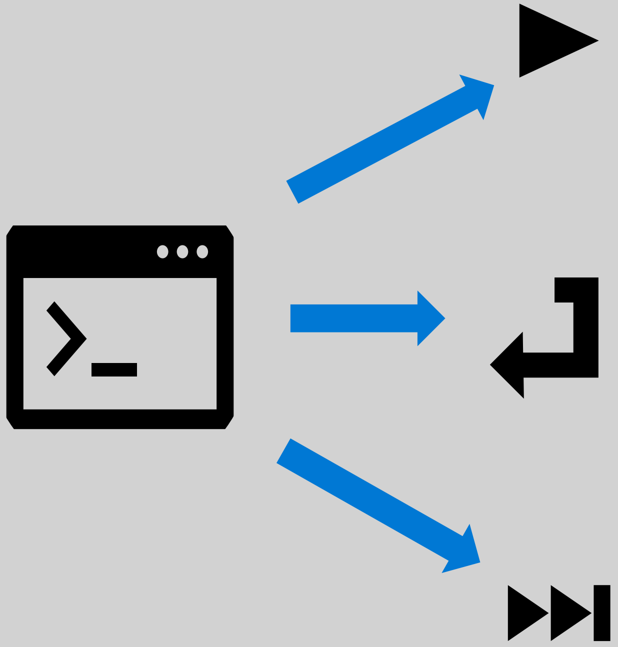
Represented with **>>**

Caused by incomplete **pairs** or certain characters

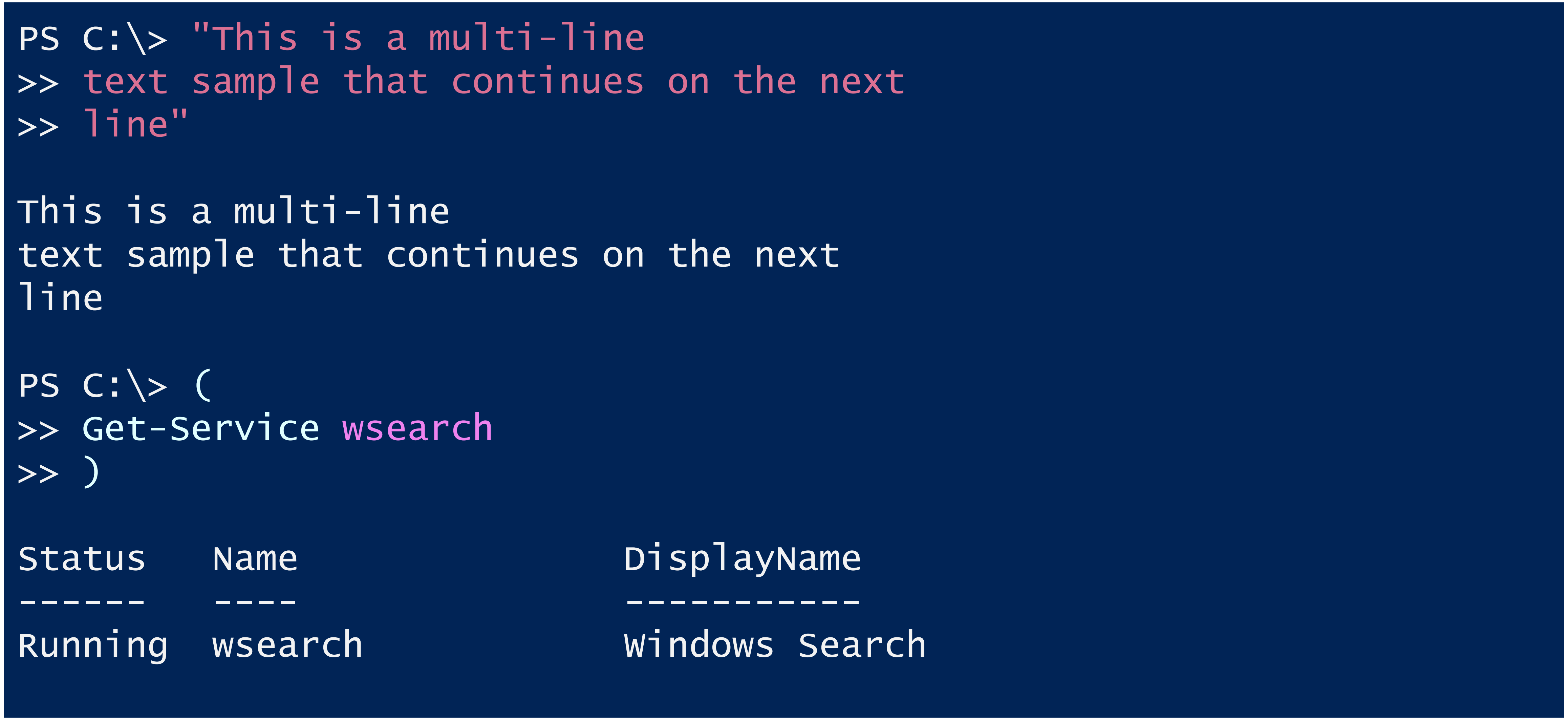
Force with **Shift+Enter**

Can abort with **Ctrl+C** if it is a mistake

Incomplete Matching Pairs	**{ }, ( ), [ ], " ", ' '**
Empty **Pipe Character**	**|**
Backtick or Grave Accent	**`**



Line Continuation - Example



```
PS C:\> "This is a multi-line
>> text sample that continues on the next
>> line"

This is a multi-line
text sample that continues on the next
line

PS C:\> (
>> Get-Service wsearch
>> )

Status   Name               DisplayName
------   ----               -----------
Running  wsearch            Windows Search
```

Demonstration

Command Termination and Line Continuation

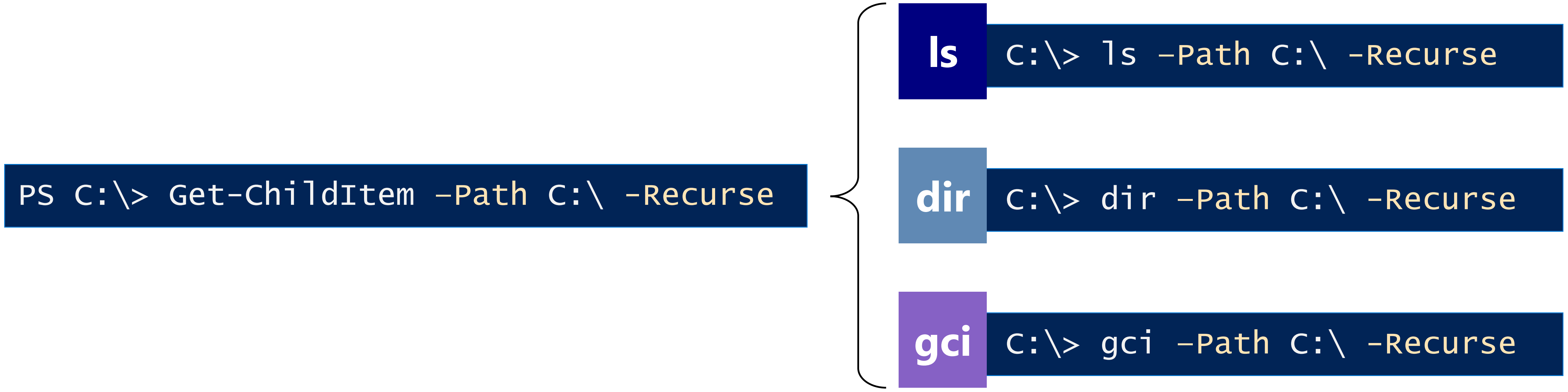
What is an Alias

PowerShell provides **alternative names** for frequently used cmdlets

Many aliases for common **cmd** and **terminal** commands exist for ease of use

Custom aliases can be **created**

Using parameters is **identical** to the cmdlet it references



```
PS C:\> Get-ChildItem -Path C:\ -Recurse
```

```
C:\> ls -Path C:\ -Recurse
```

```
C:\> dir -Path C:\ -Recurse
```

```
C:\> gci -Path C:\ -Recurse
```

Listing all Aliases

```
PS C:\> Get-Alias

CommandType     Name                      ModuleName
-----------     ----                      ----------
Alias           % -> ForEach-Object
Alias           ? -> Where-Object
Alias           ac -> Add-Content
Alias           cd -> Set-Location
Alias           chdir -> Set-Location
...

PS C:\> Get-Command -CommandType Alias

CommandType     Name                            Version     Source
-----------     ----                            -------     ------
Alias           % -> ForEach-Object
Alias           ? -> Where-Object
Alias           ac -> Add-Content
...
```

Finding Alias Definition

```
PS C:\> Get-Alias -Name Dir

CommandType     Name                      Version    Source
-----------     ----                      -------    ------
Alias           dir -> Get-ChildItem

PS C:\> Get-Command -Name Dir

CommandType     Name                      Version    Source
-----------     ----                      -------    ------
Alias           dir -> Get-ChildItem

PS C:\> Get-Alias -Definition Get-ChildItem

CommandType     Name                      Version    Source
-----------     ----                      -------    ------
Alias           dir -> Get-ChildItem
Alias           gci -> Get-ChildItem
Alias           ls -> Get-ChildItem
```

Creating an Alias

Custom aliases only exist in your **current** session

Aliases can be **exported** and **imported**

Aliases can be added in a **profile**

```
PS C:\> New-Alias -Name list -Value Get-ChildItem

PS C:\> list

    Directory: C:\

Mode             LastWriteTime     Length Name
----             -------------     ------ ----
d----            5/09/2013   1:40 PM      Intel
d-r--           21/10/2013   1:31 PM      Program Files
d-r--           10/12/2013  10:26 AM      Program Files (x86)
d----            1/12/2013   1:32 PM      Scripts
```

Demonstration

Built-in Aliases

LAB 1: Introduction to Commands

45 minutes

LAB

Microsoft

© Copyright Microsoft Corporation. All rights reserved.