PowerShell Practice Commands

Sections 2.1 to 2.5

Complete Hands-On Guide for Beginners

Created by: Babashaheer

Network Applications (QHO443)

How to Use This Guide

✓ Open PowerShell on your Windows computer

✓ Type each command exactly as shown

✓ Press Enter after each command

✓ Commands marked with ■ should be saved in a .ps1 file

✓ Commands without ■ can be typed directly

Key Symbols

■ = Tip or helpful hint

■■ = Warning or important note

■ = Expected output or result

■ = Must be saved in a .ps1 file

■■ = Type directly in PowerShell

Section 2.1: Parameters - Customising Commands

■■ BASIC COMMANDS (Type and Press Enter)

```
Get-Service
```

■ Shows ALL services on your computer

```
Get-Process
```

■ Shows ALL running programs

```
Get-ChildItem
```

■ Shows all files in current folder

```
Get-LocalUser
```

■ Shows all users on this computer

```
Get-LocalGroup
```

■ Shows all groups on this computer

■■ USING PARAMETERS (Add -Name to be specific)

```
Get-Service -Name Spooler
```

■ Shows ONLY the Spooler service (printer service)

```
Get-Process -Name explorer
```

■ Shows ONLY the explorer process (File Explorer)

```
Get-LocalUser -Name Administrator
```

■ Shows ONLY the Administrator user

```
Get-LocalGroup -Name Administrators
```

■ Shows ONLY the Administrators group

■■ USING WILDCARDS (Search by pattern)

```
Get-ChildItem *.txt
```

■ Shows ONLY files ending with .txt

```
Get-ChildItem *.pdf
```

■ Shows ONLY PDF files

```
Get-Process -Name *chrome*
```

■ Shows all processes with 'chrome' in the name

■■ SWITCH PARAMETERS (On/Off options)

```
Get-ChildItem -Recurse
```

■ Shows files in current folder AND all subfolders

```
Get-ChildItem -Force
```

■ Shows hidden and system files too

```
Get-ChildItem -Recurse -Force
```

■ Shows everything, including hidden files in subfolders

■ TIP: Use TAB to auto-complete parameters. Type 'Get-Service -N' then press TAB!

Section 2.2: Getting Help

■■ USING THE HELP SYSTEM

```
Get-Help Get-Service
```

■ Shows what Get-Service does

```
Get-Help Get-Process
```

■ Shows what Get-Process does

```
Get-Help Get-Service -Examples
```

■ Shows real examples of Get-Service

```
Get-Help Get-Process -Examples
```

■ Shows real examples of Get-Process

```
Get-Help *file*
```

■ Searches for all help topics about files

■■ FINDING COMMANDS

```
Get-Command -Verb Get
```

■ Shows ALL commands that start with 'Get-'

```
Get-Command -Verb Set
```

■ Shows ALL commands that start with 'Set-'

```
Get-Command -Verb New
```

■ Shows ALL commands that start with 'New-'

```
Get-Command -Noun Service
```

■ Shows ALL commands that work with Services

```
Get-Command -Noun Process
```

■ Shows ALL commands that work with Processes

■ TIP: If you forget a command, use Get-Command to search for it!

Section 2.3: Pipelines - Chaining Commands

■■ BASIC PIPELINES (Type the whole line, then press Enter)

```
Get-Service | Sort-Object Status
```

■ Gets all services and sorts them by Status

```
Get-Process | Sort-Object CPU
```

■ Gets all processes and sorts them by CPU usage

```
Get-ChildItem | Sort-Object Name
```

■ Gets all files and sorts them by name

■■ FILTERING WITH WHERE-OBJECT

```
Get-Service | Where-Object Status -eq 'Running'
```

■ Shows ONLY services that are Running

```
Get-Service | Where-Object Status -eq 'Stopped'
```

■ Shows ONLY services that are Stopped

```
Get-Process | Where-Object CPU -gt 10
```

■ Shows processes using more than 10% CPU

```
Get-ChildItem | Where-Object Length -gt 1MB
```

■ Shows files bigger than 1 MB

■■ SELECTING SPECIFIC COLUMNS

```
Get-Service | Select-Object Name
```

■ Shows ONLY the Name column

```
Get-Service | Select-Object Name, Status
```

■ Shows ONLY Name and Status columns

```
Get-Process | Select-Object Name, CPU
```

■ Shows ONLY Name and CPU columns

```
Get-ChildItem | Select-Object Name, Length
```

■ Shows ONLY Name and file size

■■ COMBINING MULTIPLE PIPES

```
Get-Service | Where-Object Status -eq 'Running' | Sort-Object Name
```

■ Gets running services, then sorts them by name

```
Get-Service | Where-Object Status -eq 'Running' | Select-Object Name
```

■ Gets running services, then shows only their names

```
Get-Process | Sort-Object CPU | Select-Object Name, CPU
```

■ Sorts processes by CPU, then shows only Name and CPU columns

■ TIP: The pipe symbol | is above the Enter key (Shift + \)

Section 2.4: Variables

■■ CREATING VARIABLES (Press Enter after each line)

```
$name = 'Jake'
```

■ Stores 'Jake' in a variable called $name

```
$name
```

■ Displays what's stored in $name

```
$age = 20
```

■ Stores the number 20 in $age

```
$age
```

■ Displays what's stored in $age

```
$city = 'London'
```

■ Stores 'London' in $city

```
$country = 'UK'
```

■ Stores 'UK' in $country

■■ USING VARIABLES IN SENTENCES

```
Write-Host 'Hello $name'
```

■ Prints: Hello Jake

```
Write-Host 'I live in $city, $country'
```

■ Prints: I live in London, UK

```
Write-Host 'I am $age years old'
```

■ Prints: I am 20 years old

■■ DOING MATH WITH VARIABLES

```
$num1 = 10
```

■ Store 10

```
$num2 = 25
```

■ Store 25

```
$total = $num1 + $num2
```

■ Add them together

```
$total
```

■ Shows: 35

```
$price = 50
```

■ Store price

```
$quantity = 3
```

■ Store quantity

```
$cost = $price * $quantity
```

■ Multiply

```
Write-Host 'Total: £$cost'
```

■ Shows: Total: £150

■■ STORING COMMAND RESULTS

```
$myProcess = Get-Process -Name explorer
```

■ Stores the explorer process info

```
$myProcess
```

■ Shows all the information

```
$myProcess.ProcessName
```

■ Shows just the name

```
$myProcess.CPU
```

■ Shows just the CPU usage

```
$myService = Get-Service -Name Spooler
```

■ Stores the Spooler service info

```
$myService.Status
```

■ Shows if it's Running or Stopped

```
$today = Get-Date
```

■ Stores today's date and time

```
$today
```

■ Shows the date and time

■ TIP: Variables ALWAYS start with $ (dollar sign)

Section 2.5: Conditional Logic (If/Else)

■■ **IMPORTANT: If/Else statements MUST be saved in a .ps1 file!**

HOW TO CREATE AND RUN SCRIPTS:

1. Open PowerShell
2. Type: notepad test.ps1 (then press Enter)
3. Click 'Yes' when asked to create the file
4. Type your code in Notepad
5. Save the file (Ctrl+S)
6. Close Notepad
7. Type: .\test.ps1 (then press Enter)

■ EXAMPLE 1: Check Age (Save as age.ps1)

```
$age = 20 if ($age -ge 18) { Write-Output "You are an adult" } else { Write-Output
"You are a minor" }
```

■ Output: You are an adult

■ EXAMPLE 2: Check Password Length (Save as password.ps1)

```
$password = "abc123" if ($password.Length -ge 8) { Write-Output "Strong password" }
else { Write-Output "Password too short!" }
```

■ Output: Password too short!

■ EXAMPLE 3: Check Number (Save as number.ps1)

```
$number = 15 if ($number -gt 10) { Write-Output "$number is greater than 10" } else {
Write-Output "$number is 10 or less" }
```

■ Output: 15 is greater than 10

■ EXAMPLE 4: Check Service Status (Save as checkservice.ps1)

```
$service = Get-Service -Name Spooler if ($service.Status -eq "Running") {
Write-Output "Service is working!" } else { Write-Output "Service is stopped" }
```

■ Output depends on whether Spooler is running

■ EXAMPLE 5: Check Temperature (Save as temp.ps1)

```
$temperature = 25 if ($temperature -gt 30) { Write-Output "It's hot!" } else {
Write-Output "Temperature is comfortable" }
```

■ Output: Temperature is comfortable

■ EXAMPLE 6: Using AND (-and) operator (Save as agecheck.ps1)

```
$age = 25 if ($age -ge 18 -and $age -lt 65) { Write-Output "Working age adult" } else
{ Write-Output "Not working age" }
```

■ Output: Working age adult

COMPARISON OPERATORS REFERENCE:

Operator	Meaning	Example	Result
-eq	Equal to	5 -eq 5	True
-ne	Not equal	5 -ne 3	True
-gt	Greater than	10 -gt 5	True
-lt	Less than	5 -lt 10	True
-ge	Greater or equal	5 -ge 5	True
-le	Less or equal	3 -le 5	True

■■ TROUBLESHOOTING: Execution Policy Error

THE MOST COMMON ERROR - Scripts Cannot Run

Error Message You Might See:

```
\1.ps1 : File C:\users\Beastmaster\Downloads\1.ps1 cannot be loaded because running
scripts is disabled on this system. For more information, see
about_Execution_Policies at https://go.microsoft.com/fwlink/?LinkID=135170.

At line:1 char:1

+ .\1.ps1

+ CategoryInfo : SecurityError: (:) [], PSSecurityException

+ FullyQualifiedErrorId : UnauthorizedAccess
```

WHAT IS THIS ERROR?

This is a PowerShell execution policy restriction. Windows is blocking .ps1 scripts by default for security reasons. This is NORMAL and easy to fix!

■ SOLUTION 1: Quick Fix (Recommended - Safe)

Allow scripts ONLY for your user account:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

1. Open PowerShell (normal mode, not admin)
2. Type the command above and press Enter
3. When asked 'Do you want to change the execution policy?', type Y and press Enter
4. Now you can run your scripts with: .\yourscript.ps1

■ SOLUTION 2: One-Time Bypass (Alternative)

Run script without changing policy permanently:

```
powershell -ExecutionPolicy Bypass -File .\yourscript.ps1
```

This bypasses the policy ONLY for this one script run. You need to type this command every time you want to run a script.

WHICH SOLUTION SHOULD YOU USE?

Method	Pros	Cons	Best For
Solution 1 (Set-ExecutionPolicy)	• Set it once, works forever • Only affects your account • Safe and recommended	• Need to run once first	Most users (Recommended)
Solution 2 (Bypass per run)	• No permanent changes • Works immediately	• Must type long command each time • Gets annoying quickly	One-time scripts or shared computers

■■ IMPORTANT NOTES:

1. **RemoteSigned policy is SAFE** - It only allows local scripts you created, not random scripts from the internet

2. **You DON'T need Administrator rights** for Solution 1 when using -Scope CurrentUser

3. **This is a Windows security feature**, not a bug - it protects you from malicious scripts

4. **Never use 'Unrestricted' policy** - even in labs, it's unnecessarily dangerous

PRACTICE EXERCISES - Try These Yourself!

EXERCISE 1: Basic Commands

1. Show all services on your computer
2. Show only the 'Spooler' service
3. Show all running services
4. Show all processes sorted by CPU
5. Count how many services are running (hint: use .Count)

EXERCISE 2: Variables

1. Create a variable with your name
2. Create a variable with your age
3. Print 'My name is [name] and I am [age] years old'
4. Store two numbers and multiply them
5. Store the result of Get-Date in a variable

EXERCISE 3: If/Else (Save in .ps1 files)

1. Create a script that checks if a number is even or odd
2. Create a script that checks if a password is at least 8 characters
3. Create a script that checks if a service is running
4. Create a script that checks if temperature is above 30 degrees
5. Create a script that checks if someone can vote (age 18+)

TROUBLESHOOTING - Common Problems

Problem: Script won't run - 'cannot be loaded because running scripts is disabled'

Solution: Use this command: powershell -ExecutionPolicy Bypass -File .\yourscript.ps1

Problem: 'Get-Service' is not recognized

Solution: Make sure you're using PowerShell, not Command Prompt (cmd)

Problem: Variable not working

Solution: Make sure you typed the $ symbol before the variable name

Problem: If/Else not working when typed line by line

Solution: If/Else MUST be saved in a .ps1 file first, then run with .\filename.ps1

Problem: 'The term '\test.ps1' is not recognized'

Solution: You forgot the dot before the backslash. Use: .\test.ps1 (not \test.ps1)

Problem: File not found

Solution: Make sure you're in the correct folder. Use 'cd' to navigate or save file in current folder

QUICK REFERENCE SHEET

Essential Commands:

- Get-Service → Show all services
- Get-Process → Show all running programs
- Get-ChildItem → Show files in current folder
- Get-Help [command] → Get help about a command
- Get-Command -Verb [verb] → Find commands by verb
- Where-Object → Filter results
- Sort-Object → Sort results
- Select-Object → Choose which columns to show

Key Symbols:

- $ → Marks a variable (e.g., $name)
- | → Pipe symbol, chains commands together
- -eq → Equal to
- -ne → Not equal
- -gt → Greater than
- -lt → Less than
- -ge → Greater than or equal
- -le → Less than or equal

Remember These Key Points!

1. Single-line commands can be typed directly in PowerShell
2. If/Else and Loops MUST be saved in .ps1 files
3. Always use .\filename.ps1 to run scripts (don't forget the dot!)
4. Variables start with $ (dollar sign)
5. Use TAB to auto-complete commands and parameters
6. The pipe | symbol is Shift + \ (above Enter key)
7. Use Get-Help when you forget how a command works

Good luck with your PowerShell learning! Practice makes perfect! ■