- ✓ Cmdlets & Help
- ✓ The Pipeline
- ✓ Objects & Properties
- ✓ Variables & Types
- ✓ Conditions & Loops
- ✓ Functions & Params
- ✓ Files · CSV · JSON
- ✓ Modules & Automation

PS>

FULL COURSE

12 Lessons · Beginner to Advanced

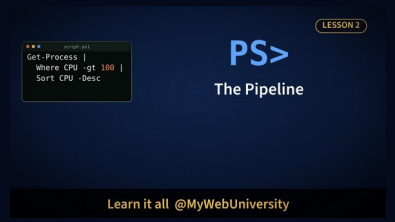
Free · English & Dari

Learn it all @MyWebUniversity

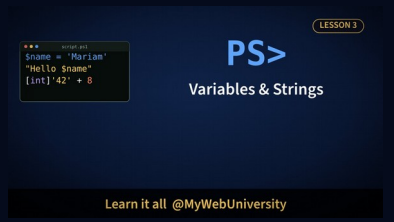
The 12 Lessons



1. Welcome to PowerShell



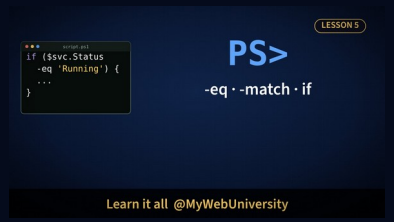
2. The Object Pipeline



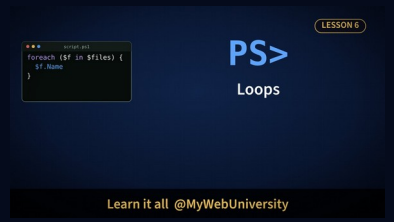
3. Variables, Types and Strings



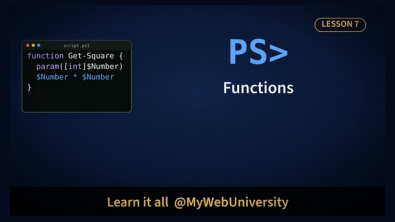
4. Arrays, Hashtables and Custom Objects



5. Operators and Conditions



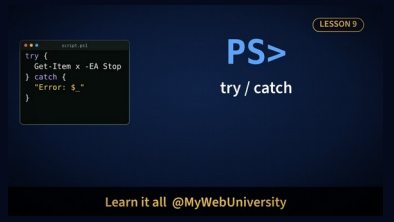
6. Loops



7. Functions and Parameters



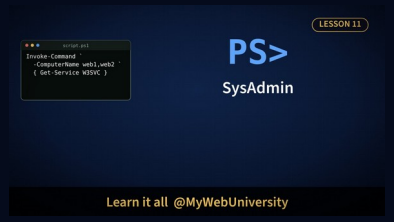
8. Files, CSV and JSON



9. Error Handling



10. Scripts, Modules and Profiles



11. System Administration with PowerShell



12. Best Practices and Next Steps



A FREE MYWEBUNIVERSITY COURSE

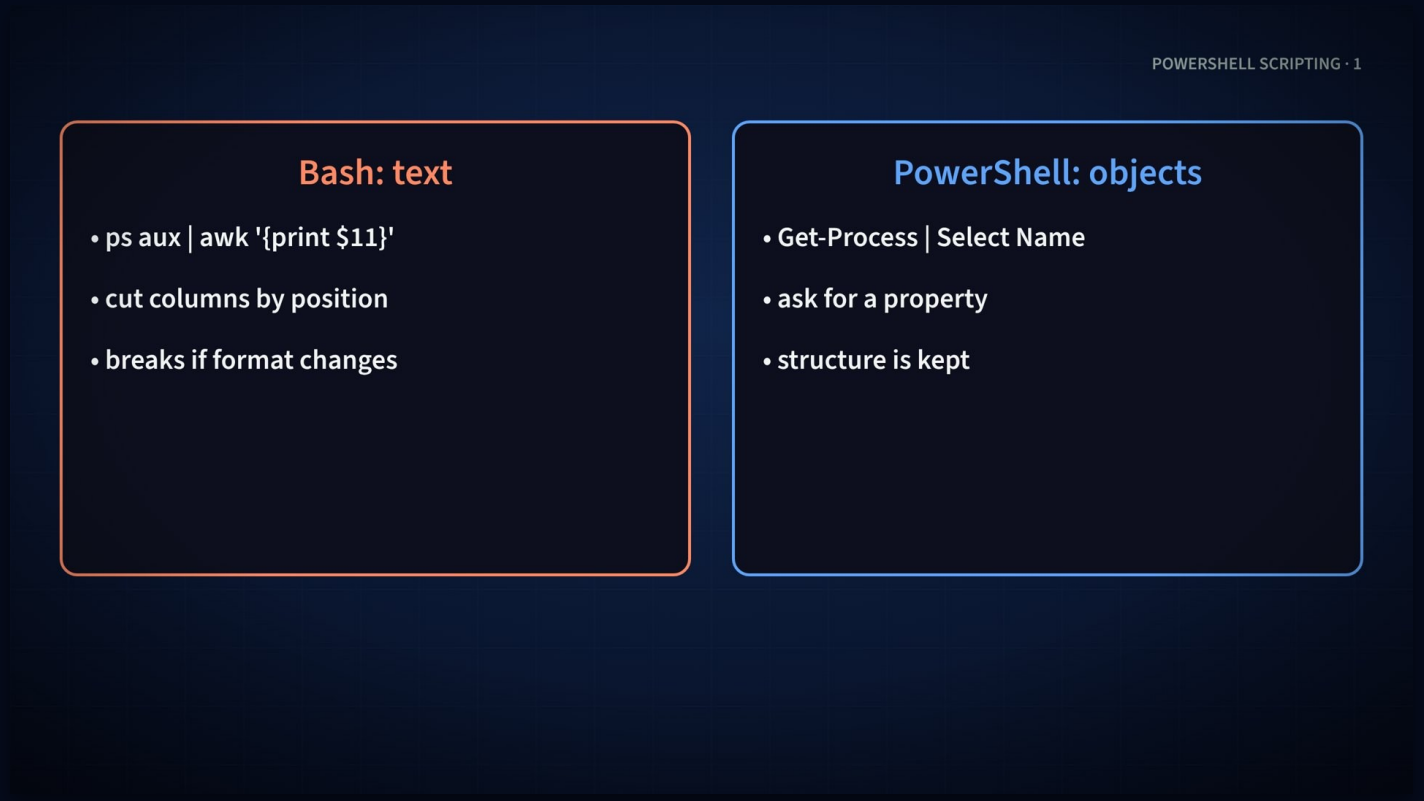
POWERSHELL SCRIPTING

LESSON 1

Welcome to PowerShell

Objects not text, pwsh on every platform, verb-noun commands and Get-Help

What Is PowerShell?



- PowerShell is a shell and scripting language from Microsoft. Its big idea: commands output …
- So you never cut columns out of text. You ask for the property you want, like the Name or …

1.01

PowerShell 7 Everywhere

terminal

```
PS> winget install Microsoft.PowerShell      # Windows
$ sudo snap install powershell --classic      # Linux
$ brew install powershell                     # macOS

PS> $PSVersionTable.PSVersion
Major  Minor  Patch
7      4      5
```

- Windows PowerShell 5.1 comes with Windows. The modern version, PowerShell 7, called pwsh, …
- This course uses PowerShell 7. Install it, then type pwsh in any terminal to start.

1.02

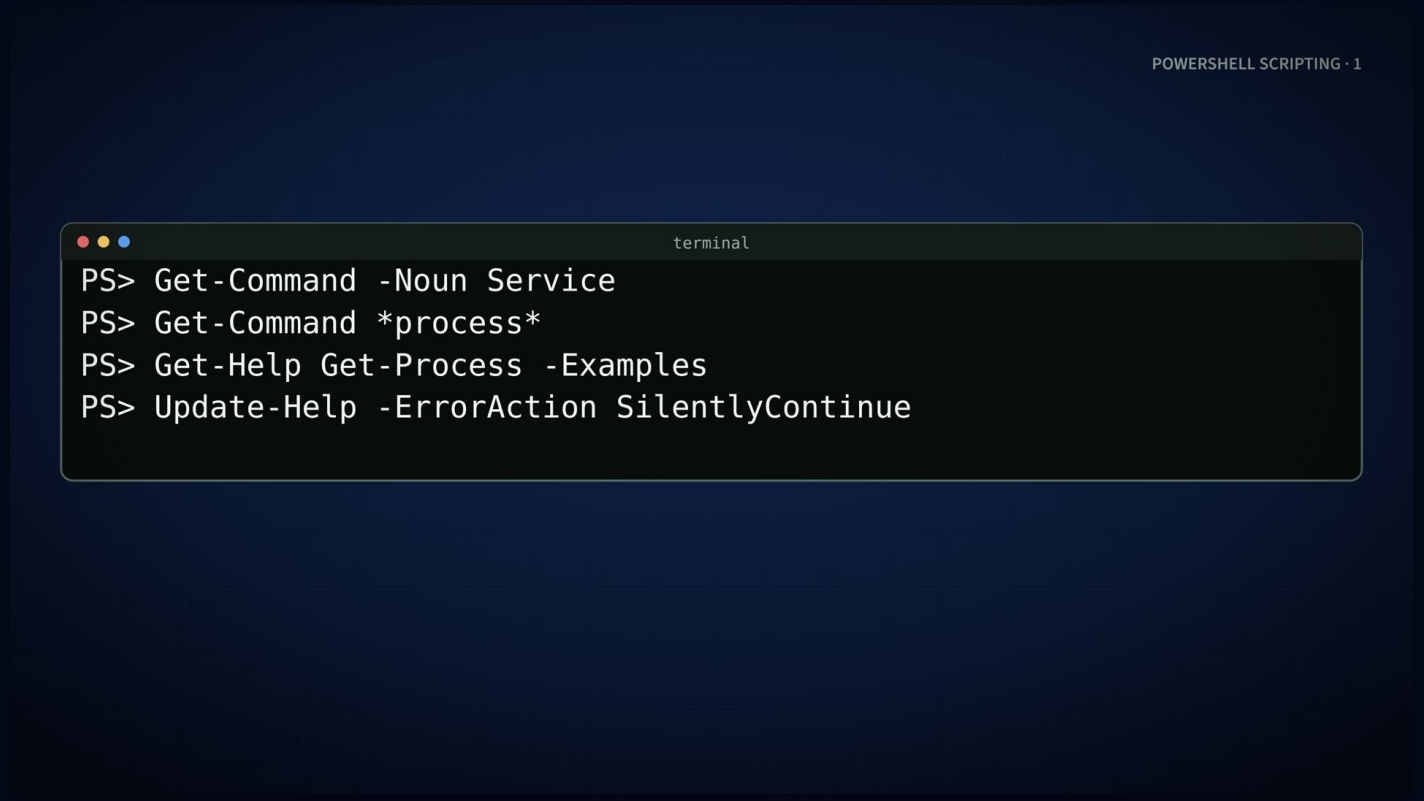
Verb-Noun Commands

Verb	Examples
Get	Get-Process · Get-Service · Get-ChildItem
Set	Set-Location · Set-Content
New	New-Item · New-Object
Remove	Remove-Item · Remove-Variable
Start / Stop	Start-Service · Stop-Process

- PowerShell commands are called cmdlets and are named verb dash noun: Get-Process, …
- Because names are predictable, you can often guess a command. Get-Verb lists the approved …

1.03

Finding Your Way: Get-Help and Get-Command



- Get-Command finds commands, for example everything with the noun Service. Get-Help shows …
- Run Update-Help once to download the full help files. These two commands are how …

1.04

Aliases and Familiar Names

Alias	Full cmdlet
ls · dir · gci	Get-ChildItem
cd · sl	Set-Location
cat · gc	Get-Content
ps · gps	Get-Process
? · where	Where-Object

- PowerShell has aliases so you feel at home: ls, dir and gci all run Get-ChildItem, and cd …
- Use aliases when typing, but write full cmdlet names in scripts so others can read them.

1.05



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 2

The Object Pipeline

Get-Member, Select-Object, Where-Object, Sort-Object and Format-Table

Get-Member: What Is in an Object?



- Pipe any command into Get-Member to see the object type and all its properties and methods.
- This is the most important habit in PowerShell: when you are not sure what you can ask for, …

2.01

Select-Object

select.ps1

```
Get-Process | Select-Object Name, Id, CPU -First 5

Get-Process | Select-Object Name,
    @{ Name = 'MemoryMB'; Expression = { [math]::Round($_.WorkingSet / 1MB, 1) } }
```

- Select-Object picks the properties you want, and dash First or dash Last picks a number of …
- You can even add calculated properties, like memory in megabytes instead of bytes.

2.02

Where-Object

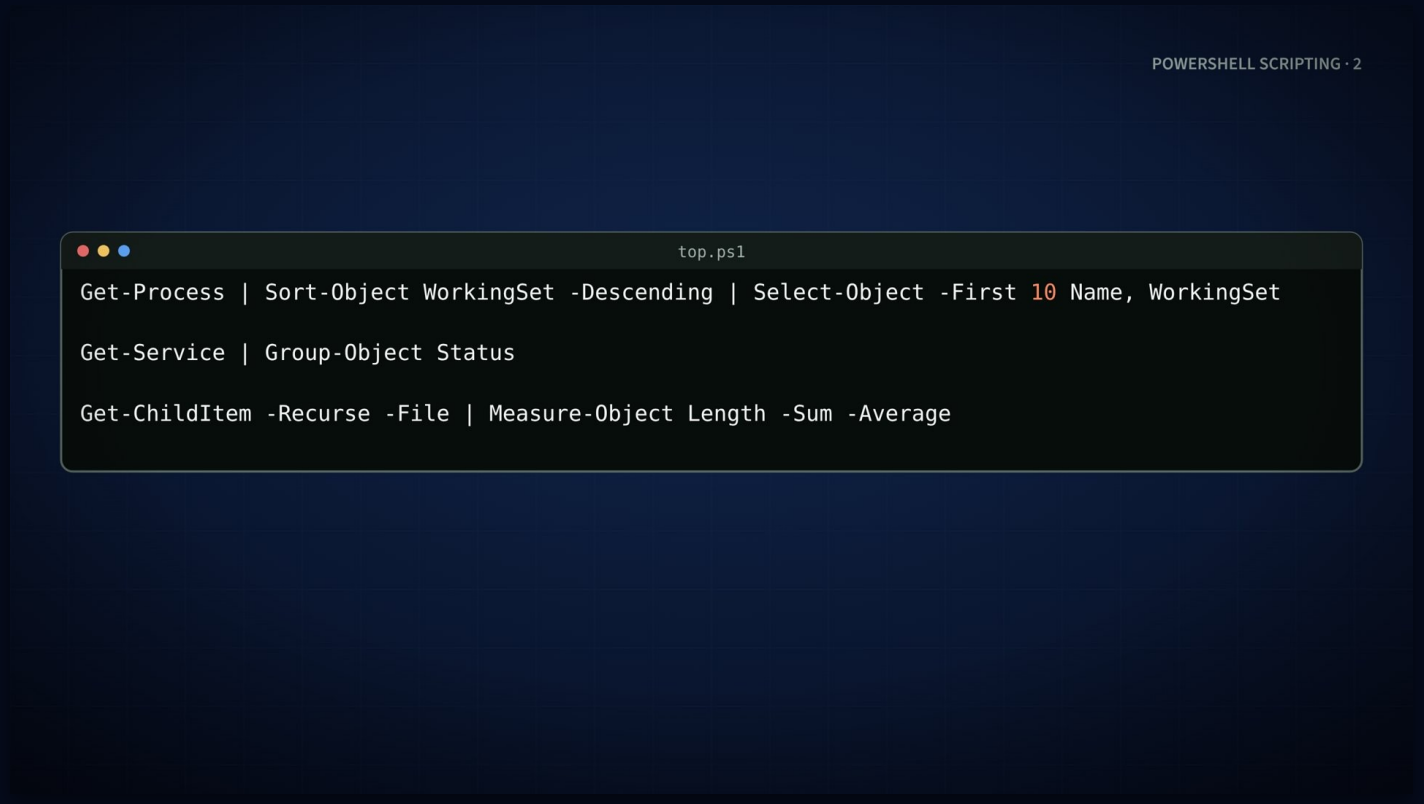
where.ps1

```
Get-Process | Where-Object { $_.CPU -gt 100 }
Get-Service | Where-Object Status -eq 'Running'
Get-ChildItem | Where-Object { $_.Length -gt 10MB }
Get-ChildItem *.log | Where-Object Name -like 'app*'
```

- Where-Object filters objects with a condition. Dollar underscore, or dollar PSItem, is the …
- Comparison operators are words with a dash: dash gt for greater than, dash eq for equals, …

2.03

Sort, Group and Measure



- Sort-Object orders objects by a property, Group-Object groups them, and Measure-Object …
- Together they answer real questions, like which ten processes use the most memory.

2.04

Formatting Output



- Format-Table and Format-List control how objects look on screen. Dash AutoSize makes tables …
- Important: format at the very end of the pipeline only. After Format-Table, you have …

2.05



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 3

Variables, Types and Strings

$variables, .NET types, casting, string interpolation and here-strings

Variables

vars.ps1

```
$name   = 'Mariam'
$age    = 21
$procs  = Get-Process                # store objects
$count  = $procs.Count
"$name has $count processes running"
```

- Every variable starts with a dollar sign, and you can store anything in it: a number, a …
- Variables are not case-sensitive, and dollar null means nothing. Get-Variable lists all …

3.01

Types and Casting

types.ps1

```
$age.GetType().Name              # Int32
[int]$n = '42'                   # string -> int, locked as int
$n + 8                           # 50
[datetime]'2026-10-24'           # Saturday, October 24, 2026
[double]::Parse('3.14')
```

- Under the hood every value is a .NET object with a type. dollar age dot GetType shows it. …
- Common types are int, double, string, bool, datetime and array. Casting a string to int …

3.02

Single vs Double Quotes

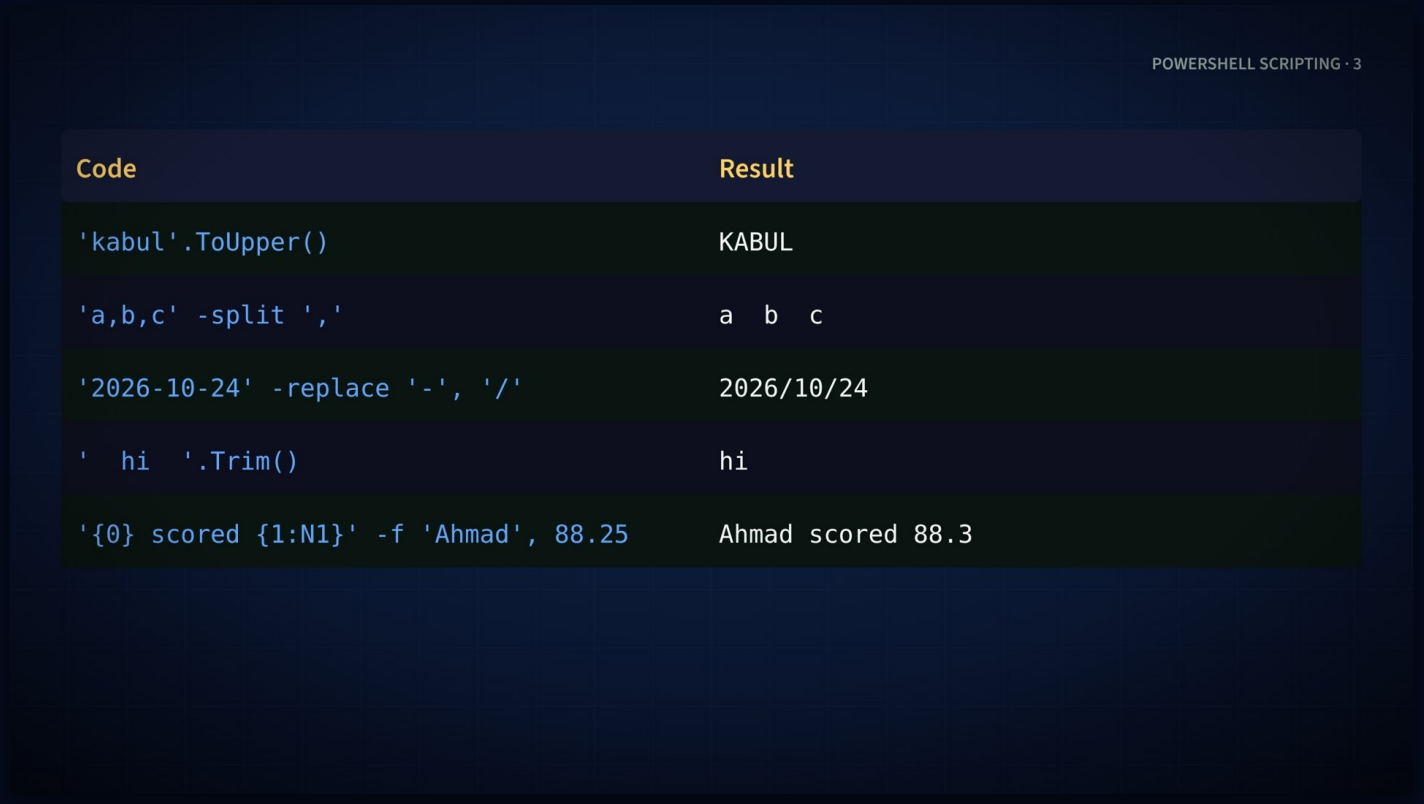
strings.ps1

```
$user = Get-Process -Id $PID
"Hello $name"                    # Hello Mariam
'Hello $name'                    # Hello $name
"Process: $($user.Name)"         # expression inside a string
"Total: $(2 + 3)`nDone"          # `n = new line
```

- Double quotes expand variables, and dollar parenthesis expands whole expressions, like a …
- Single quotes are literal. The backtick is the escape character, so backtick n is a new line.

3.03

String Methods and Operators



- Strings have .NET methods: ToUpper, Replace, Split, Trim and Contains. PowerShell also has …
- The dash f format operator builds neat text with placeholders in curly braces.

3.04

Here-Strings

report.ps1

```
$body = @"
Server : $env:COMPUTERNAME
Date   : $(Get-Date -Format 'yyyy-MM-dd')
Status : All services running
"@
$body | Set-Content report.txt
```

- A here-string holds many lines. Start with at-sign double quote, end with double quote …
- They are perfect for email bodies, config files and SQL queries inside scripts.

3.05



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 4

Arrays, Hashtables and Custom Objects

@(), @{}, [PSCustomObject] and building your own data

Arrays

arrays.ps1

```
$cities = 'Kabul', 'Herat', 'Mazar'
$cities[0]            # Kabul
$cities[-1]           # Mazar
$cities += 'Kandahar'
$cities.Count         # 4
$list = [System.Collections.Generic.List[string]]::new(); $list.Add('Balkh')
```

- An array is a list. Create one with commas, or with at-sign parentheses. Indexes start at …
- Plus equals appends, and dot Count gives the length. For big lists, a generic List is much …

4.01

Hashtables

hash.ps1

```
$port = @{ web = 80; ssh = 22; db = 5432 }
$port.ssh                    # 22
$port['db']                  # 5432
$port.mail = 25              # add
$port.Keys; $port.ContainsKey('web')
```

- A hashtable stores key-value pairs. Create it with at-sign curly braces, and read a value …
- Hashtables are perfect for settings, lookups and splatting parameters into a command.

4.02

Splatting

splat.ps1

```
$params = @{
    Path        = 'C:\Logs'
    Filter      = '*.log'
    Recurse     = $true
    ErrorAction = 'SilentlyContinue'
}
Get-ChildItem @params
```

- Splatting passes a hashtable of parameters to a command with an at sign instead of a dollar …
- Long commands become readable, and you can build parameters step by step before calling.

4.03

[PSCustomObject]

objects.ps1

```powershell
$students = @(
    [PSCustomObject]@{ Name = 'Mariam'; Score = 95; City = 'Kabul' }
    [PSCustomObject]@{ Name = 'Ahmad';  Score = 88; City = 'Herat' }
)
$students | Sort-Object Score -Descending | Format-Table
$students | Export-Csv students.csv -NoTypeInformation
```

- Put PSCustomObject in front of a hashtable and you get a real object with properties, just …
- Your objects work with every cmdlet: Sort-Object, Where-Object, Export-Csv and ConvertTo-Json.

4.04

Building Reports from Objects

disks.ps1

```powershell
$report = foreach ($d in Get-PSDrive -PSProvider FileSystem) {
    [PSCustomObject]@{
        Drive  = $d.Name
        UsedGB = [math]::Round($d.Used / 1GB, 1)
        FreeGB = [math]::Round($d.Free / 1GB, 1)
    }
}
$report | Sort-Object FreeGB | Format-Table -AutoSize
```

- A common pattern: loop over computers or files, build one custom object per item, and …
- This turns any information into a clean table that Excel, a web page or another script can …

4.05



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 5

Operators and Conditions

-eq, -gt, -like, -match, -contains, if/elseif and switch

Comparison Operators

Operator	Example	Meaning
`-eq · -ne`	`5 -eq 5`	equal · not equal
`-gt · -lt`	`10 -gt 3`	greater · less
`-ge · -le`	`60 -ge 60`	or equal
`-ceq`	`'A' -ceq 'a'`	case-sensitive: False

- PowerShell compares with words, not symbols, because the greater-than sign already means …
- dash eq, dash ne, dash gt, dash lt, dash ge and dash le. They are case-insensitive by …

5.01

-like, -match and -contains

match.ps1

```
'app.log' -like '*.log'                    # True
'user42' -match '^user(\d+)$'              # True
$Matches[1]                                # 42
'Kabul', 'Herat' -contains 'Herat'         # True
'Mazar' -in 'Kabul', 'Herat'               # False
```

- dash like uses simple wildcards like star dot log. dash match uses regular expressions and …
- dash contains and dash in check whether a list holds a value, which reads almost like English.

5.02

Logical Operators

logic.ps1

```
$score = 85; $city = 'Herat'
($score -ge 80) -and ($city -eq 'Herat')      # True
($score -lt 50) -or  ($city -like 'K*')       # False
-not (Test-Path 'C:\nope')                    # True
```

- Combine conditions with dash and, dash or and dash not, or the exclamation mark for not.
- Use parentheses to make the order clear, exactly as in mathematics.

5.03

if, elseif and else

service.ps1

```powershell
$svc = Get-Service -Name 'Spooler'
if ($svc.Status -eq 'Running') {
    Write-Host "$($svc.Name) is running" -ForegroundColor Green
} elseif ($svc.StartType -eq 'Disabled') {
    Write-Warning "$($svc.Name) is disabled"
} else {
    Start-Service $svc.Name
}
```

- The if statement looks like C: a condition in parentheses and code in curly braces, with …
- Here we check a service and start it if it is stopped, which is real administration work.

5.04

switch

switch.ps1

```
switch ($env:OS) {
    'Windows_NT' { 'Windows' }
    default      { 'Linux or macOS' }
}

switch -Regex -File app.log {
    'ERROR'   { $errors++ }
    'WARN'    { $warnings++ }
}
```

- switch compares one value against many choices. It can use wildcards with dash Wildcard and …
- It can even loop over a whole file with dash File, testing every line.

5.05



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 6

Loops

foreach, ForEach-Object, for, while, do and loop control

The foreach Statement

foreach.ps1

```powershell
$files = Get-ChildItem -Path .\logs -Filter *.log
foreach ($f in $files) {
    '{0,-20} {1,8:N1} KB' -f $f.Name, ($f.Length / 1KB)
}
```

- foreach loops over a collection already in memory. It is fast and easy to read, and the …
- Here we loop over log files and print each name with its size in kilobytes.

6.01

ForEach-Object in the Pipeline

foreach_object.ps1

```
Get-ChildItem *.txt | ForEach-Object { $_.Name.ToUpper() }

1..5 | ForEach-Object { $_ * $_ }            # 1 4 9 16 25

'web1', 'web2', 'db1' | ForEach-Object -Parallel {
    Test-Connection $_ -Count 1 -Quiet
} -ThrottleLimit 3
```

- ForEach-Object works inside the pipeline, one object at a time, with dollar underscore as …
- Because it streams, it uses little memory even with millions of objects. PowerShell 7 adds …

6.02

for, while and do

loops.ps1

```
for ($i = 1; $i -le 3; $i++) { "server$i" }

$tries = 0
do {
    $ok = Test-Connection 'web1' -Count 1 -Quiet
    $tries++; Start-Sleep -Seconds 2
} until ($ok -or $tries -ge 5)
```

- The for loop counts with a start, a condition and a step, like C. while repeats as long as …
- do-while and do-until always run at least once, which is useful for retry loops.

6.03

break, continue and Ranges

find_big.ps1

```
foreach ($f in Get-ChildItem -Recurse -File) {
    if ($f.Attributes -match 'System') { continue }
    if ($f.Length -gt 1GB) {
        "Found: $($f.FullName)"
        break
    }
}
```

- continue skips to the next item and break leaves the loop. The range operator, two dots, …
- Here we skip system files and stop after finding the first file larger than one gigabyte.

6.04

Which Loop Should I Use?

Loop	Best for
`foreach ($x in $list)`	data in memory, fast
`ForEach-Object`	pipelines, huge data, -Parallel
`for ($i...)`	you need the index
`do { } until ()`	retry until success

- Use foreach when the data is already in a variable and speed matters. Use ForEach-Object in …
- Use for when you need an index, and do-until for retrying something until it works.

6.05



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 7

Functions and Parameters

param blocks, types, defaults, [CmdletBinding()], validation and pipeline input

Your First Function

functions.ps1

```powershell
function Get-Square {
    param([int]$Number)
    $Number * $Number
}
Get-Square -Number 7        # 49
Get-Square 9                # 81 (positional)
```

- Define a function with the word function and a Verb-Noun name. Put parameters in a param …
- Whatever the function outputs is returned. You do not need the word return, though you can …

7.01

Defaults, Switches and Mandatory

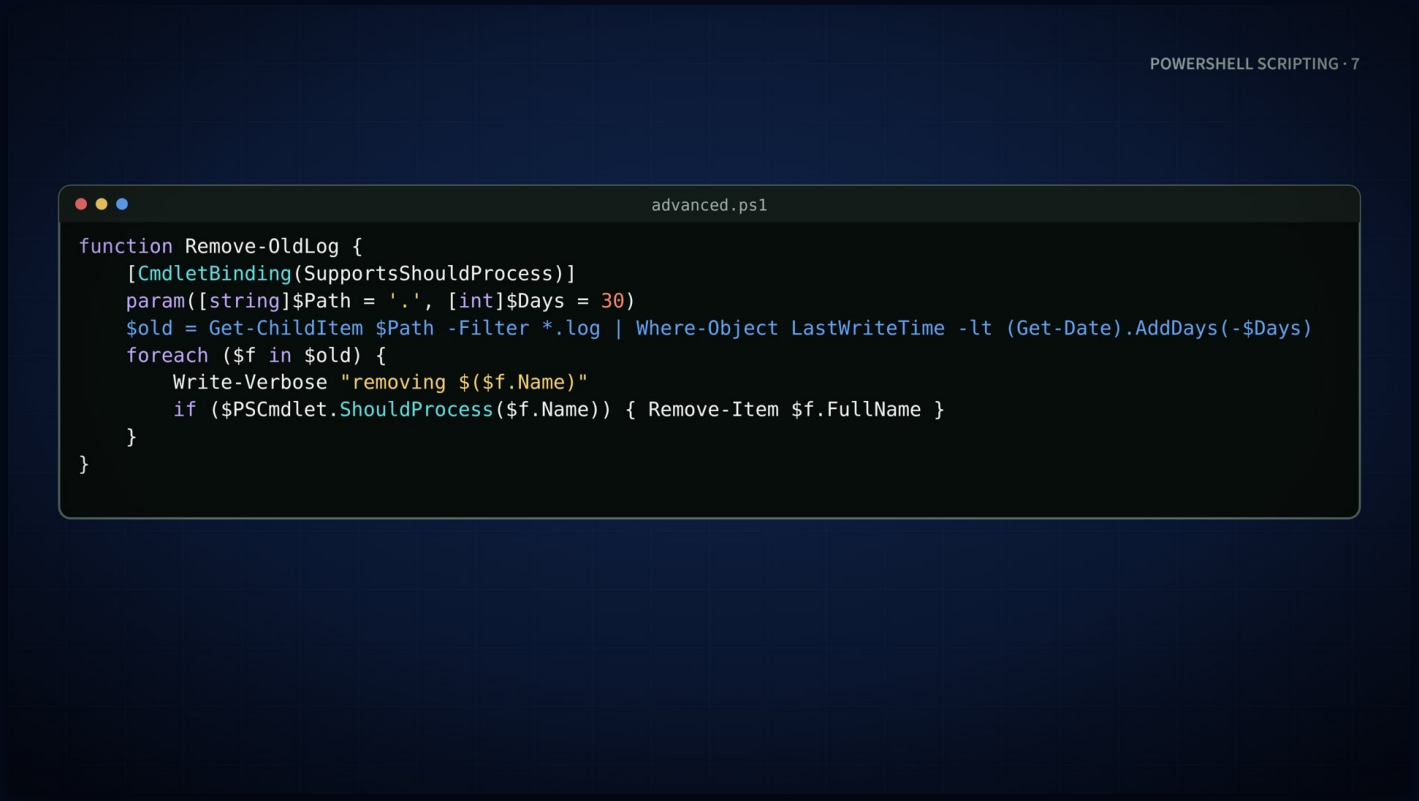
params.ps1

```powershell
function Backup-Folder {
    param(
        [Parameter(Mandatory)] [string]$Path,
        [string]$Destination = 'D:\Backup',
        [switch]$Compress
    )
    if ($Compress) { "zip $Path -> $Destination" } else { "copy $Path -> $Destination" }
}
```

- Give parameters default values, make important ones mandatory, and use the switch type for …
- PowerShell asks for mandatory values if they are missing, and tab completion works for your …

7.02

Advanced Functions: [CmdletBinding()]



- Add CmdletBinding and your function gets the common parameters for free: dash Verbose, dash …
- Write-Verbose messages appear only when the user asks for dash Verbose, which keeps output …

7.03

Parameter Validation

validate.ps1

```powershell
function Set-Environment {
    param(
        [ValidateSet('dev', 'test', 'prod')] [string]$Name,
        [ValidateRange(1, 65535)]           [int]$Port = 8080,
        [ValidateScript({ Test-Path $_ })]  [string]$ConfigFile
    )
    "deploying to $Name on port $Port"
}
```

- Validation attributes reject bad input before your code runs. ValidateSet allows only …
- Your function becomes safer and gives clear error messages, with almost no extra code.

7.04

Accepting Pipeline Input

pipeline.ps1

```
function Get-FileSizeMB {
    param([Parameter(ValueFromPipeline)] [System.IO.FileInfo]$File)
    begin   { $total = 0 }
    process { $total += $File.Length; '{0} {1:N1} MB' -f $File.Name, ($File.Length / 1MB) }
    end     { 'TOTAL {0:N1} MB' -f ($total / 1MB) }
}
Get-ChildItem *.iso | Get-FileSizeMB
```

- Mark a parameter with ValueFromPipeline and put your code in a process block. Now your …
- The begin block runs once at the start and the end block once at the finish, just like …

7.05



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 8

Files, CSV and JSON

Get-Content, Set-Content, Test-Path, Import-Csv, Export-Csv and ConvertTo-Json

Text Files

text.ps1

```
$lines = Get-Content .\servers.txt
Get-Content .\app.log -Tail 20
Get-Content .\app.log -Wait          # like tail -f
'started' | Add-Content .\run.log
Get-Process | Out-File .\procs.txt
```

- Get-Content reads a file as an array of lines, and Set-Content and Add-Content write and …
- Use dash Raw to read the whole file as one string, and dash Tail to read just the last …

8.01

Paths and Items

paths.ps1

```
$dir = Join-Path $HOME 'reports'
if (-not (Test-Path $dir)) { New-Item -ItemType Directory -Path $dir | Out-Null }
Copy-Item .\report.csv (Join-Path $dir 'report-old.csv')
Remove-Item .\temp\* -Recurse -WhatIf      # preview first
```

- Test-Path checks whether a file or folder exists. New-Item creates, Copy-Item and Move-Item …
- Join-Path builds paths correctly on Windows and Linux, so your scripts work on both.

8.02

CSV as Objects

csv.ps1

```
$students = Import-Csv .\students.csv
$students | Where-Object { [int]$_.Score -ge 80 } |
    Sort-Object { [int]$_.Score } -Descending |
    Export-Csv .\honors.csv -NoTypeInformation
($students | Measure-Object Score -Average).Average
```

- Import-Csv turns each row into an object with a property for each column. You can filter, …
- Export-Csv does the reverse, so a few lines turn one spreadsheet into another.

8.03

JSON

json.ps1

```
$cfg = Get-Content .\config.json -Raw | ConvertFrom-Json
$cfg.database.port                  # 5432
$cfg.database.port = 5433
$cfg | ConvertTo-Json -Depth 5 | Set-Content .\config.json
```

- ConvertTo-Json turns objects into JSON text, and ConvertFrom-Json turns JSON back into …
- That makes PowerShell excellent for config files and web APIs. Use dash Depth for nested data.

8.04

Other Formats

Format	Read · Write	Best for
CSV	`Import-Csv · Export-Csv`	spreadsheets
JSON	`ConvertFrom-Json · ConvertTo-Json`	programs, APIs
XML	`[xml](Get-Content f.xml)`	older systems
HTML	`ConvertTo-Html`	reports for people

- PowerShell also reads XML with the xml type accelerator, and HTML reports come from …
- Choose the format your audience needs: CSV for spreadsheets, JSON for programs, HTML for …

8.05



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 9

Error Handling

Terminating vs non-terminating errors, try/catch/finally, -ErrorAction and $Error

Two Kinds of Errors



- Non-terminating errors print a red message but the command continues, like Get-ChildItem on …
- Terminating errors stop the command. try and catch only see terminating errors, which is …

9.01

try, catch and finally

trycatch.ps1

```
try {
    $data = Get-Content .\config.json -ErrorAction Stop | ConvertFrom-Json
    "loaded $($data.name)"
}
catch [System.Management.Automation.ItemNotFoundException] { Write-Warning 'config file missing' }
catch { Write-Error "failed: $($_.Exception.Message)" }
finally { 'done' }
```

- Put risky code in try. Add dash ErrorAction Stop so normal errors become terminating and …
- Inside catch, dollar underscore holds the error. finally always runs, so close connections …

9.02

-ErrorAction and Preferences

Value	Effect
`Stop`	make it terminating (catchable)
`Continue`	show error, keep going (default)
`SilentlyContinue`	hide error, keep going
`Ignore`	hide and do not record

- dash ErrorAction controls one command: Stop, Continue, SilentlyContinue or Ignore. ...
- Many professionals put ErrorActionPreference equals Stop at the top, like set dash e in Bash.

9.03

throw and Your Own Errors

throw.ps1

```
function Get-Config {
    param([string]$Path)
    if (-not (Test-Path $Path)) { throw "Config not found: $Path" }
    Get-Content $Path -Raw | ConvertFrom-Json
}
try { Get-Config .\missing.json } catch { "Error: $_" }
```

- throw creates a terminating error with your message. Use it when input is wrong or a …
- Write-Warning and Write-Error are softer: they report a problem but let the caller decide …

9.04

$Error and Logging

log.ps1

```
Start-Transcript -Path .\run.log -Append
function Write-Log($msg) { "$(Get-Date -Format s) $msg" | Add-Content .\app.log }
try { Restart-Service 'Spooler' -ErrorAction Stop; Write-Log 'restarted' }
catch { Write-Log "FAILED: $($_.Exception.Message)" }
$Error[0].InvocationInfo.ScriptLineNumber
Stop-Transcript
```

- The automatic variable dollar Error keeps a list of recent errors, newest first. dollar …
- Combine catch with a log function, and Start-Transcript to record a whole session, so you …

9.05



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 10

Scripts, Modules and Profiles

Execution policy, .ps1 files, script parameters, modules and $PROFILE

Execution Policy

terminal

```
PS> Get-ExecutionPolicy -List
PS> Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
PS> Unblock-File .\downloaded.ps1
```

- On Windows, the execution policy decides which scripts may run. RemoteSigned is a sensible …
- It is a safety belt, not a security wall. Set it once for your user with Set-ExecutionPolicy.

10.01

Script Files and Parameters

Get-DiskReport.ps1

```
<#
.SYNOPSIS
  Shows free space on every drive.
.EXAMPLE
  .\Get-DiskReport.ps1 -MinFreeGB 20
#>
param([int]$MinFreeGB = 10)
Get-PSDrive -PSProvider FileSystem | Where-Object { $_.Free / 1GB -lt $MinFreeGB }
```

- A script is a dot ps1 file. Put a param block at the top, exactly like a function, and run …
- Comment-based help with synopsis and example lines makes Get-Help work on your own scripts.

10.02

Modules

MyTools.psm1

```
# Documents\PowerShell\Modules\MyTools\MyTools.psm1
function Get-Uptime { (Get-Date) - (Get-CimInstance Win32_OperatingSystem).LastBootUpTime }
function Get-TopProcess { Get-Process | Sort-Object CPU -Descending | Select-Object -First 5 }
Export-ModuleMember -Function Get-Uptime, Get-TopProcess

# PS> Import-Module MyTools; Get-Uptime
```

- A module is a package of functions. PowerShell finds thousands in the PowerShell Gallery: …
- Your own module is just a dot psm1 file of functions in a folder with the same name. …

10.03

The PowerShell Gallery

terminal

```
PS> Find-Module -Name *excel*
PS> Install-Module ImportExcel -Scope CurrentUser
PS> Get-Process | Export-Excel .\procs.xlsx
PS> Get-Module -ListAvailable
```

- The PowerShell Gallery is the public library of modules, for cloud, databases, Active …
- Install for your user with dash Scope CurrentUser, and always check who publishes a module …

10.04

Your Profile

profile.ps1

```
# PS> if (!(Test-Path $PROFILE)) { New-Item $PROFILE -Force }
Set-Alias np notepad
Import-Module MyTools
function prompt { "PS $($PWD.Path.Split('\')[-1])> " }
# PS> . $PROFILE     # reload
```

- dollar PROFILE is a script that runs every time PowerShell starts, like dot zshrc in Zsh. …
- Create it with New-Item if it does not exist, and reload with dot-sourcing: a dot, a space, …

10.05



A FREE MYWEBUNIVERSITY COURSE

POWERSHELL SCRIPTING

LESSON 11

System Administration with PowerShell

Processes, services, event logs, CIM, remoting and REST APIs

Processes and Services

services.ps1

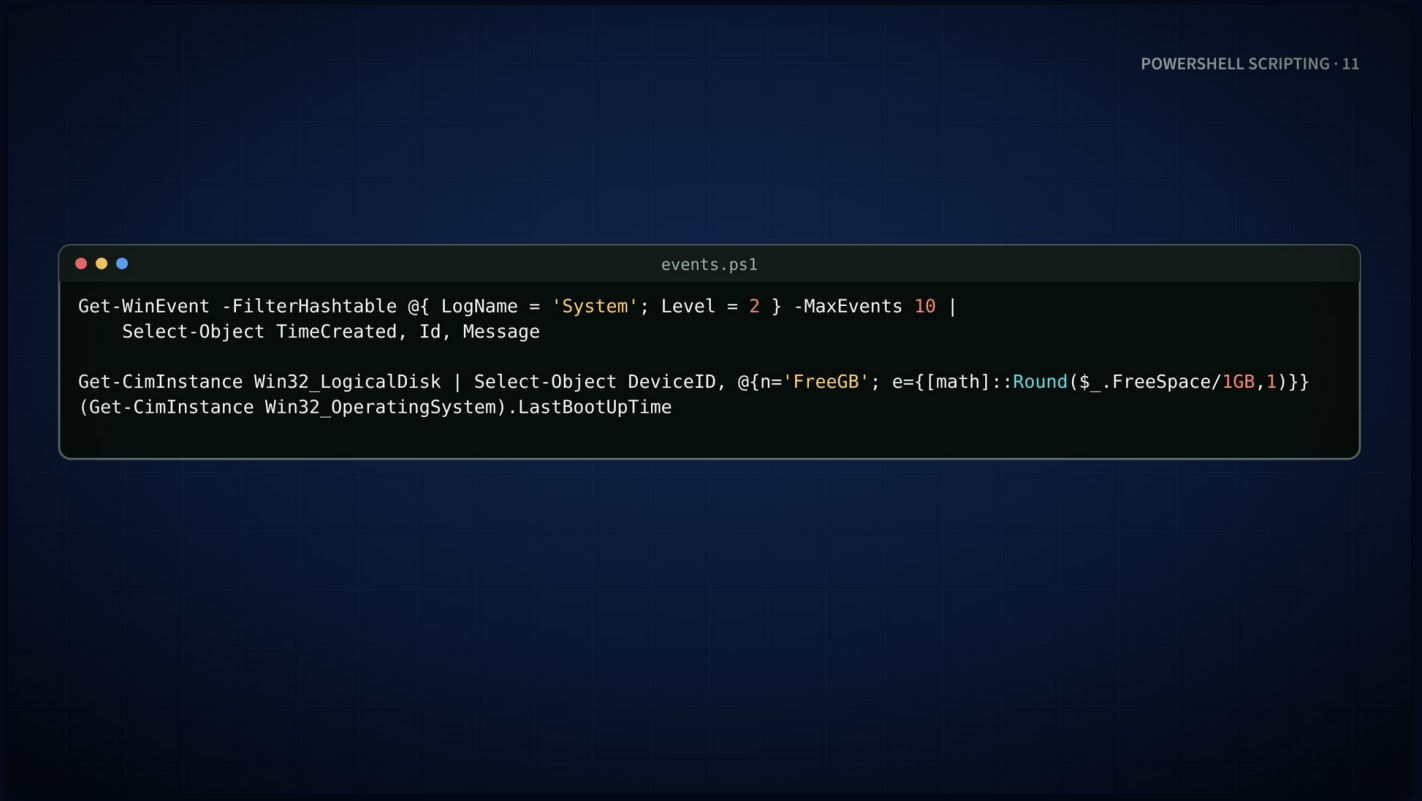
```
Get-Process chrome -ErrorAction SilentlyContinue | Stop-Process -WhatIf

Get-Service | Where-Object { $_.StartType -eq 'Automatic' -and $_.Status -ne 'Running' } |
    ForEach-Object { Start-Service $_.Name -Verbose }
```

- Get-Process and Stop-Process manage running programs. Get-Service, Start-Service, …
- Combine them with Where-Object to find and fix problems, such as automatic services that …

11.01

Event Logs and CIM



- Get-WinEvent reads Windows event logs, for example the last errors in the System log. …
- Get-CimInstance asks the system about hardware and the operating system: disks, memory, …

11.02

Remoting

remote.ps1

```
$servers = 'web1', 'web2', 'db1'
Invoke-Command -ComputerName $servers -ScriptBlock {
    Get-Service W3SVC | Select-Object Status
}

Enter-PSSession -HostName linux01 -UserName mariam      # over SSH
```

- Invoke-Command runs a script block on many computers at once and returns objects, with the …
- Enter-PSSession opens an interactive session on one remote machine. PowerShell 7 can also …

11.03

Talking to Web APIs

api.ps1

```powershell
$r = Invoke-RestMethod 'https://api.github.com/repos/PowerShell/PowerShell'
"$($r.full_name) has $($r.stargazers_count) stars"

$body = @{ text = 'Backup finished on ' + $env:COMPUTERNAME } | ConvertTo-Json
Invoke-RestMethod -Uri $webhookUrl -Method Post -Body $body -ContentType 'application/json'
```

- Invoke-RestMethod calls a web API and turns the JSON answer straight into objects. Send …
- This is how PowerShell automates cloud services, ticket systems and chat notifications.

11.04

Users and Scheduled Tasks

task.ps1

```
$action  = New-ScheduledTaskAction -Execute 'pwsh.exe' -Argument '-File C:\Scripts\Backup.ps1'
$trigger = New-ScheduledTaskTrigger -Daily -At 2am
Register-ScheduledTask -TaskName 'NightlyBackup' -Action $action -Trigger $trigger

Get-LocalUser | Select-Object Name, Enabled, LastLogon
```

- Local users are managed with Get-LocalUser and New-LocalUser, and in companies the …
- Register-ScheduledTask runs your script on a schedule, the Windows version of cron.

11.05



A FREE MYWEBUNIVERSITY COURSE

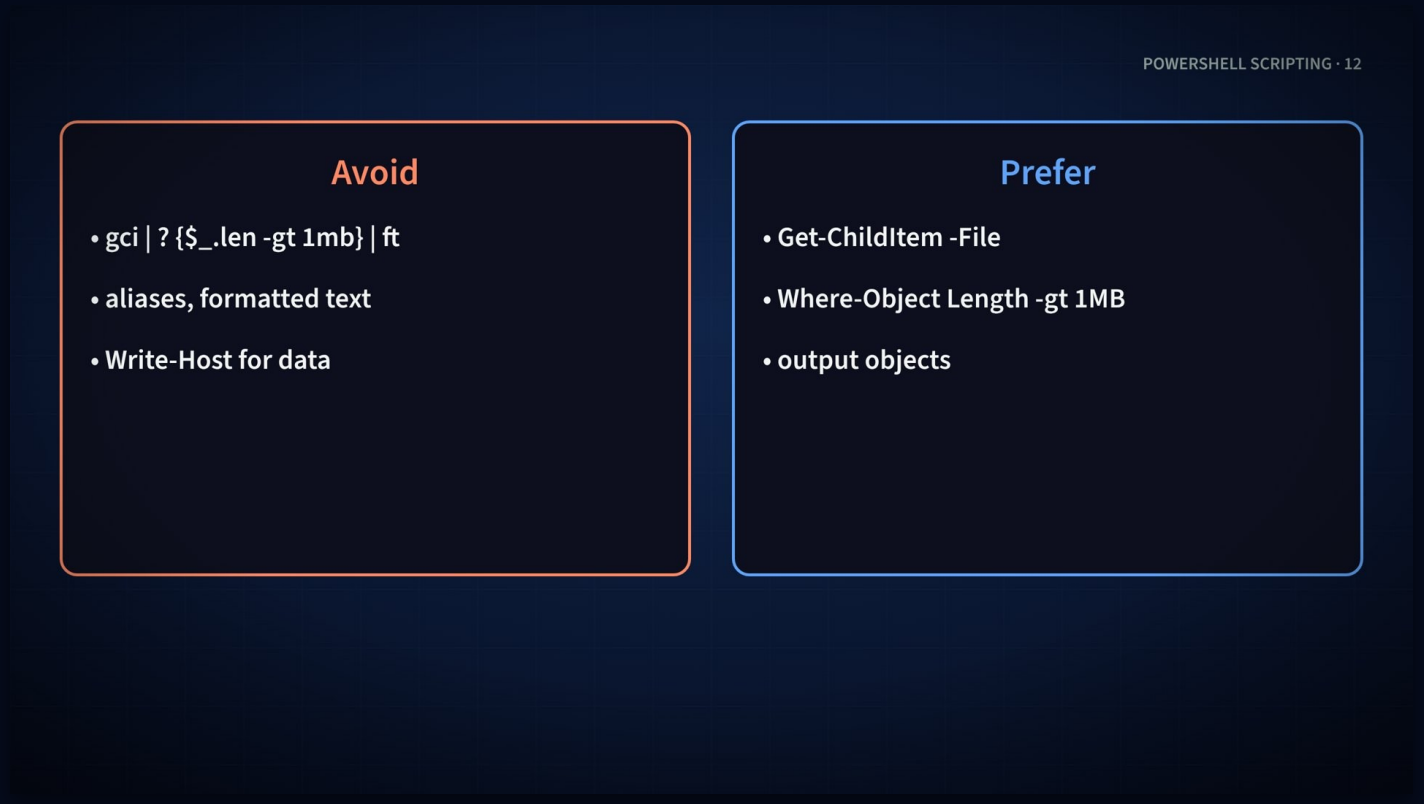
POWERSHELL SCRIPTING

LESSON 12

Best Practices and Next Steps

Style, testing with Pester, security, PowerShell vs Bash vs Python, projects

Style Rules



- Use full cmdlet names and named parameters in scripts, approved verbs for function names, …
- Output objects, not formatted text, so the next person can sort, filter or export your …

12.01

Testing with Pester

Get-Square.Tests.ps1

```
BeforeAll { . $PSScriptRoot\Get-Square.ps1 }
Describe 'Get-Square' {
    It 'squares 7' { Get-Square 7 | Should -Be 49 }
    It 'squares 0' { Get-Square 0 | Should -Be 0 }
}
# PS> Invoke-Pester
```

- Pester is the testing framework for PowerShell. Describe groups tests, It defines one test, …
- Run Invoke-Pester before every change, and your scripts stay correct as they grow.

12.02

Security



- No passwords — Get-Credential, vaults
- -WhatIf first — preview big changes
- Least privilege — admin only when needed

12.03

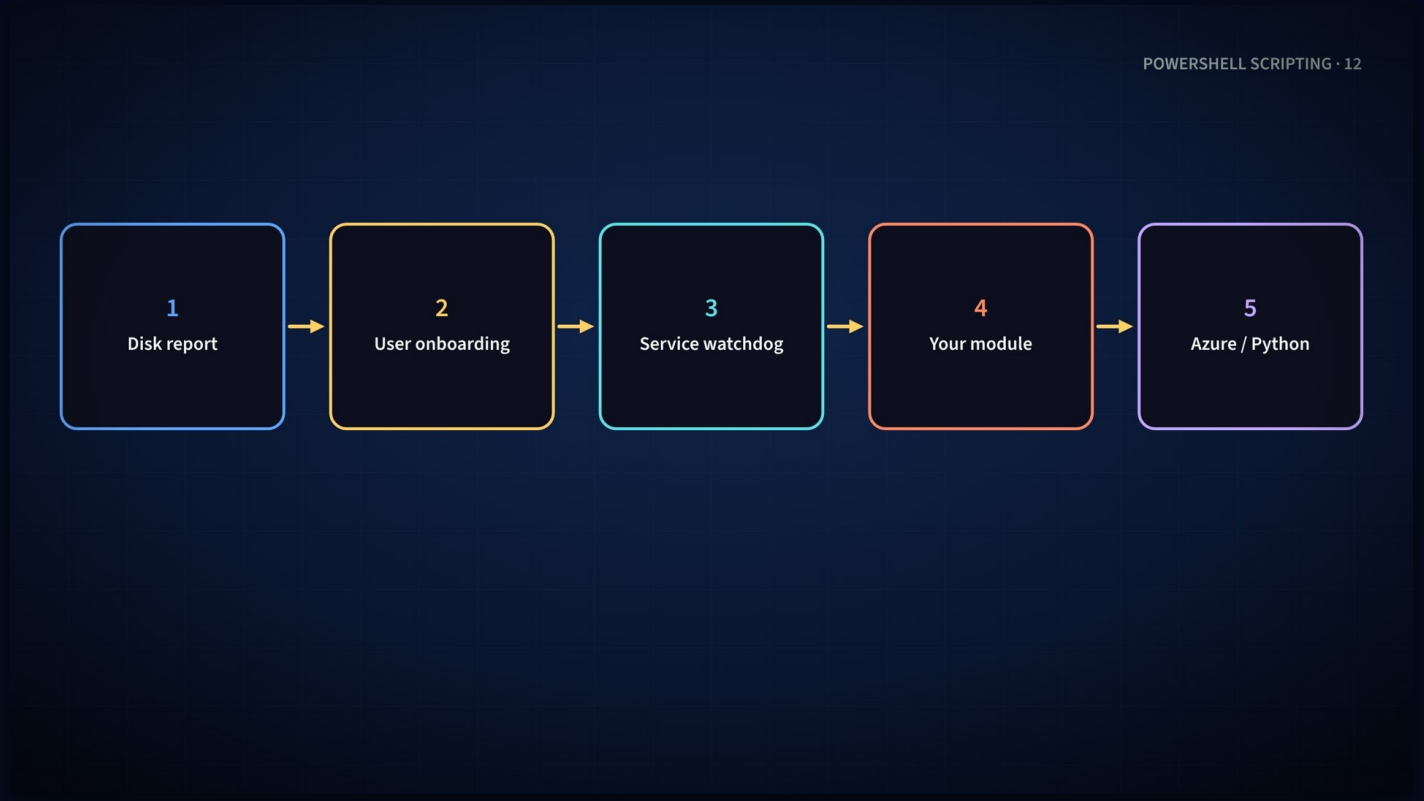
PowerShell, Bash or Python?

Tool	Best for
PowerShell	Windows, Microsoft 365, Azure, objects
Bash	Linux servers, containers, glue
Python	data, web, large programs

- PowerShell is the best choice for Windows, Microsoft 365 and Azure administration, and …
- Python wins for data, web and large programs. Knowing all three makes you valuable in any …

12.04

Projects and Next Steps



- 1 — Disk report
- 2 — User onboarding
- 3 — Service watchdog

12.05

Thank you for learning with us

MyWebUniversity.com

Learn it all @MyWebUniversity

Free technical education since 2000 — English and Dari