PowerShell Quick Reference

Cmdlets, pipeline, objects, scripting, modules

Basics

Commands & Help

```
Get-Help Get-Process         # show help for cmdlet
Get-Help Get-Process -Online # open online docs
Get-Command *service*        # find commands by name
Get-Alias ls                 # show alias target
```

Common Aliases

ls → Get-ChildItem	List files and directories
cd → Set-Location	Change directory
cp → Copy-Item	Copy file or directory
mv → Move-Item	Move or rename
rm → Remove-Item	Delete file or directory
cat → Get-Content	Read file contents
echo → Write-Output	Print to pipeline
cls → Clear-Host	Clear console

Variables

Variable Basics

```
$name = "Alice"          # string
$count = 42              # integer
$pi = 3.14               # double
$list = @(1, 2, 3)       # array
$hash = @{a=1; b=2}      # hashtable
```

Automatic Variables

$_	Current pipeline object
$PSVersionTable	PowerShell version info
$HOME	User's home directory
$PWD	Current directory
$null	Null value
$true / $false	Boolean constants
$Error	Array of recent errors
$LASTEXITCODE	Exit code of last native command

Environment Variables

```
$env:PATH                    # read env var
$env:MY_VAR = "value"        # set env var
Get-ChildItem Env:           # list all env vars
```

Operators

Comparison Operators

-eq -ne	Equal / not equal
-gt -lt	Greater / less than
-ge -le	Greater/less or equal
-like -notlike	Wildcard match (*****, **?**)
-match -notmatch	Regex match
-contains	Collection contains value
-in -notin	Value in collection

Logical & Other Operators

-and -or -not	Logical operators
!	Logical NOT (alias)
-replace	Regex replace: **'hi' -replace 'h','b'**
-split -join	Split / join strings
..	Range: **1..5** → 1,2,3,4,5
?:	Ternary (v7+): **$x ? 'yes' : 'no'**

Control Flow

if / elseif / else

```
if ($age -ge 18) {
    "Adult"
} elseif ($age -ge 13) {
    "Teen"
} else {
    "Child"
}
```

switch

```
switch ($color) {
    "red"   { "Stop" }
    "green" { "Go" }
    default { "Unknown" }
}
```

Loops

```
foreach ($item in $list) { $item }
for ($i=0; $i -lt 5; $i++) { $i }
while ($x -lt 10) { $x++ }
1..5 | ForEach-Object { $_ * 2 }
```

Functions

Defining Functions

```
function Get-Greeting {
    param([string]$Name = "World")
    "Hello, $Name!"
}
Get-Greeting -Name "Alice"
```

Advanced Parameters

```
function Copy-SafeFile {
    [CmdletBinding()]
    param(
        [Parameter(Mandatory)][string]$Path,
        [Parameter(Mandatory)][string]$Dest
    )
    Copy-Item $Path $Dest -WhatIf:$WhatIfPreference
}
```

Objects & Pipeline

Pipeline Basics

```
Get-Process | Sort-Object CPU -Desc | Select-Object -First 5
Get-Service | Where-Object Status -eq "Running"
Get-ChildItem | Measure-Object -Property Length -Sum
```

Key Pipeline Cmdlets

Where-Object	Filter objects: **| Where {$_.CPU -gt 10}**
Select-Object	Pick properties: **| Select Name, CPU**
Sort-Object	Sort: **| Sort CPU -Desc**
ForEach-Object	Transform each: **| ForEach {$_.Name}**
Measure-Object	Count, sum, average, min, max
Group-Object	Group by property value
Format-Table	Display as table: **| ft -Auto**
Export-Csv	Export to CSV: **| Export-Csv out.csv**

Files

File Operations

```
Get-Content log.txt                  # read file
Set-Content out.txt "hello"          # write (overwrite)
Add-Content out.txt "more"           # append
Get-Content log.txt | Select-String "error"  # grep
```

Path & File Cmdlets

Test-Path $path	Check if file/dir exists
New-Item -Type File	Create file
New-Item -Type Directory	Create directory
Resolve-Path	Get absolute path
Join-Path	Combine path segments
Split-Path	Get parent or leaf
Get-ItemProperty	File attributes and metadata
Remove-Item -Recurse	Delete directory recursively

Strings

String Basics

```
"Hello, $name"               # interpolation (double quotes)
'Hello, $name'               # literal (single quotes)
"Length: $($list.Count)"      # expression in string
@"
Multi-line
here-string with $name
"@
```

String Methods

.ToUpper() / .ToLower()	Change case
.Trim()	Remove leading/trailing whitespace
.Split(',')	Split into array
.Replace('a','b')	Replace substring
.StartsWith() / .EndsWith()	Check prefix / suffix
.Substring(0,5)	Extract substring
.Contains('text')	Check if contains
-f operator	**'{0} is {1}' -f 'sky','blue'**

Modules

Module Management

```
Get-Module -ListAvailable          # installed modules
Find-Module -Name Az*              # search gallery
Install-Module -Name Pester        # install from gallery
Import-Module ActiveDirectory      # load module
```

Module Commands

Get-Module	List loaded modules
Import-Module	Load module into session
Remove-Module	Unload module from session
Update-Module	Update installed module
Get-Command -Module X	List commands in module
$env:PSModulePath	Module search paths

Common Patterns

Error Handling

```
try {
    Get-Item "C:\missing" -ErrorAction Stop
} catch {
    "Error: $_"
} finally {
    "Cleanup here"
}
```

PowerShell Quick Reference

Execution Policy & Remoting

`Get-ExecutionPolicy`	Show current script policy
`Set-ExecutionPolicy RemoteSigned`	Allow local scripts
`Enter-PSSession -Computer SRV1`	Interactive remote session
`Invoke-Command -Computer SRV1 -Script {}`	Run script block remotely
`Start-Job { long-task }`	Run background job
`Receive-Job -Id 1`	Get background job output