

Sidebar

Programming · PowerShell Cheat Sheet

Everyday PowerShell for Windows: finding help, editing the command line, files, the object pipeline, variables and operators, scripts and the execution policy, web requests and system tasks. Written for **Windows PowerShell 5.1**, the version installed on this machine (PowerShell 7 is not installed); features that need PowerShell 7 are marked. Every command and parameter on the sheet was checked to exist in this installation.

HELP & DISCOVERY

`$PSVersionTable.PSVersion` Which PowerShell you are in (this machine: Windows PowerShell 5.1)

`Get-Help` `Get-Process` `-Examples` Show examples; also `-Full`, `-Detailed`, `-Online`, `-ShowWindow`

`Update-Help` Download the full help files (run PowerShell as administrator)

`Get-Command` `*service*` Find commands by part of a name

`Get-Command` `Stop-Service` `-Syntax` Show every way to call a command

`Get-Verb` The approved verbs: commands are named `Verb-Noun`

`Get-Member` Pipe any object to it to list its properties and methods

`Get-Alias` / `Get-Alias` `-Definition` `Get-ChildItem` List aliases / find the aliases of a command

`Show-Command` `Get-Process` A small window for filling in a command's options

`Tab` Completes commands, parameters, paths and even parameter values

COMMAND-LINE KEYS (PSREADLINE 2.0.0 HELP)

`Up` / `Down` Previous / next command in the history

`Ctrl+R` / `Ctrl+S` Search the history backwards / forwards

`F7` A pop-up list of the history, a classic Windows command prompt feature, Windows Terminal

`Tab` / `Shift+Tab` Complete forwards / backwards

`Ctrl+Space` A menu of every possible completion

`Alt+.` Insert the last argument of the previous command

`Ctrl+W` or `Ctrl+Backspace` Delete the word before the cursor

`Ctrl+Left` / `Ctrl+Right` Move one word

`Ctrl+Z` / `Ctrl+Y` Undo / redo while editing the line

`Shift+Enter` Start a new line without running

`Ctrl+L` Clear the screen

`Ctrl+C` Copy if text is selected, otherwise cancel the current operation

`Ctrl+Alt+?` Show every key binding

`Get-PSReadLineOption` See the current editing settings (PowerShell 7)

MOVING AROUND & WORKING WITH FILES

`Set-Location D:\Work` (`cd`) Change folder; `Get-Location` (`pwd`) shows where you are

`Push-Location` / `Pop-Location` Remember the current folder and go back to it later

`Get-ChildItem` `-Recurse` `-File` `-Filter *.md` (`ls`, `dir`) List files; `-Directory`, `-Hidden`, `-Force`, `-Name`, `-Depth 2` help

`New-Item` `notes.txt` `-ItemType File` Create a file; `-ItemType Directory` for a folder

`Copy-Item` `a.txt` `b.txt` `Move-Item` `Rename-Item` Copy, move and rename; `-Recurse` for folders

`Remove-Item` `old` `-Recurse` `-Force` Delete (no recycle bin). Add `-WhatIf` first to see what would happen

`Test-Path` `.\file.txt` `True` or `False`: does it exist?

`Get-Content` `log.txt` `-Tail 20` `-Wait` Show the end of a file and keep following it; `-Raw` reads it as one string

`Set-Content` / `Add-Content` / `Out-File` Write text to a file / append / write command output

`Join-Path` `A` `B` `Split-Path` `Resolve-Path` Build, take apart and resolve paths

`Compress-Archive` `-Path` `src` `-DestinationPath` `out.zip` Make a zip; `Expand-Archive` unzips it

`Get-FileHash` `file.iso` A checksum (SHA256 by default)

`Select-String` `-Path *.md` `-Pattern 'TODO'` Search inside files (`sls`); there is no `-Recurse`, so pipe from `Get-ChildItem -Recurse`

THE PIPELINE

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

`Where-Object` { `$_`.Length -gt 1MB } Keep more than 1MB objects
Short for `Where-Object { $_.Length -gt 1MB }`

`Select-Object` `Name`, `Length` Choose properties
`-ExpandProperties`

`Sort-Object` `Length` `-Descending` Sort (`sort`)

`Group-Object` `Extension` Count items per value

`Measure-Object` `Length` `-Sum` `-Average` `-Maximum`

`ForEach-Object` { `$_`.Name.ToUpper() } Run a script block on each object

`Select-Object` @{Name='MB'; Expression={\$_.Length/1024}}

`Format-Table` `-AutoSize` / `Format-List` `*` Display as table or list

`Out-GridView` A sortable, filterable window of the pipeline

`Tee-Object` `-FilePath` `out.txt` Show the output to a file

`Compare-Object` `$a` `$b` Differences between two objects

`Filter` `early` Use a command's own `-Filter` script block, much faster on big folders

VARIABLES, TYPES & OPERATORS

`$name = 'Philip'` Variables start with `$`; no declaration needed

`[int]$n = '42'` Force a type; others: `[string]`, `[datetime]`, `[bool]`, `[double]`

`$env:PATH $HOME $PWD $PROFILE` Environment variable / home / current folder / profile path

`$? $LASTEXITCODE` Did the last command succeed? / exit code of the last program

`$null $true $false` Built-in values

`-eq -ne -gt -ge -lt -le` Compare. Text comparison ignores case unless you use `-ceq`

`-like 'a*' -match '^a.+z$'` Wildcard / regular expression match; `-notlike`, `-notmatch`

`-contains -in` Is the item in the list? (`$list -contains 3` or `3 -in $list`)

`-and -or -not -xor !` Logic. In Windows PowerShell 5.1 there is no `&&` or `||` between commands

`-replace -split -join -f` Text operators: `'a,b' -split ','`, `$a -join ';'` , `'{0:N2}' -f 3.14159`

`1..5` A range of numbers

`'single $x' "double $x" "${o.Name}"` Single quotes are literal; double quotes expand variables; `$( )` inside them runs an expression

`@' ... '@ '@' ... '@` Here-strings for several lines (the closing mark must start a line)

`` (backtick)` The escape character, and a line continuation at the end of a line

CONTROL FLOW, FUNCTIONS & ERRORS

`if ($x -gt 1) { } elseif ($x -eq 1) { } else { }` Conditions

`switch ($x) { 1 { 'one' } default { 'other' } }` Multi-way choice; `switch -Wildcard` and `-Regex` also work

`foreach ($f in $files) { } for while do { } until ( )` Loops; `break` and `continue` work as usual

`function Get-Thing { param([string]$Path, [switch]$Force) ... }` A function with typed parameters; add `[CmdletBinding()]` before `param` for `-Verbose` and `-WhatIf`

`return $x / Write-Output $x` Anything a function outputs is returned; `Write-Host` only prints to the screen

`try { } catch { $_.Exception.Message } finally { }` Handle errors. `catch` only sees *terminating* errors

`-ErrorAction Stop` Make a command's errors terminating so `try/catch` catches them

`$ErrorActionPreference = 'Stop'` Stop on any error for the whole script (the default is `Continue`)

`$Error[0]` The most recent error

`throw 'message'` Raise your own error

`Write-Verbose Write-Warning Write-Error` Messages of different levels

`Measure-Command { ... }` Time a block of code

`. .\lib.ps1` Dot-sourcing: run a script in the current scope so its functions stay available

ARRAYS, HASHTABLES & OBJECTS

`$a = @(1,2,3); $a[0]; $a[-1]; $a.Count` Array

`$a += 4` Adds an item by building a new array, so

`$h = @{ name='x'; age=5 }` Hashtable; read with `$h.Values`

`[ordered]@{ a=1; b=2 }` Keeps the order you want

`[pscustomobject]@{ Name='x'; Size=3 }` Make an object

`Invoke-Thing @params` Splatting: pass a hashtable

`'a b c'.Split(' ') .Trim() .Replace('a','b')`

`Import-Csv f.csv / Export-Csv f.csv -NoTypeInformation`

`Get-Content f.json -Raw | ConvertFrom-Json`

`$obj | ConvertTo-Json -Depth 5` Turn objects into JSON levels

SCRIPTS, EXECUTION POLICY & YOUR PROFILE

`.\script.ps1` Run a script in the current folder (the file must be executable)

`Get-ExecutionPolicy -List` Show the policy at the current location. `CurrentUser` undefined

`Set-ExecutionPolicy RemoteSigned -Scope CurrentUser`

`Unblock-File .\downloaded.ps1` Remove the 'blocked' flag

`powershell -NoProfile -ExecutionPolicy Bypass`

`$PROFILE` Path of your profile script: here `C:\Users\emuba\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1`

`Test-Path $PROFILE` Does it exist yet?

`New-Item -Path $PROFILE -ItemType File -Force`

`Set-Alias gs Get-Status` A session alias; put it in your profile

`Start-Transcript -Path log.txt` Record everything

WEB, JSON & APIS

`Invoke-RestMethod https://api.example.com/items` GET and parse JSON automatically (`irm`)

`Invoke-WebRequest $url -OutFile f.zip` Download a file; the result also has `.StatusCode` , `.Content` , `.Headers` (`iwr`)

`-Method Post -Body ($o | ConvertTo-Json) -ContentType 'application/json'` Send JSON

`-Headers @{ Authorization = "Bearer $token" }` Add headers

`-UseBasicParsing` Skip the Internet Explorer engine in Windows PowerShell 5.1 (avoids a setup prompt on some machines)

`curl` and `wget` In Windows PowerShell 5.1 these are *aliases* of `Invoke-WebRequest` , not the real programs. Type `curl.exe` to run the real curl

`Test-NetConnection localhost -Port 8000` Can I reach that port? (handy for checking your FastAPI server)

`Resolve-DnsName example.com` DNS lookup

`Test-Connection example.com` Ping

PowerShell 7 only `Invoke-RestMethod -SkipHttpErrorCheck` and some other parameters do not exist in 5.1

MODULES & REMOTING

`Get-Module -ListAvailable` Modules installed

`Import-Module Name` Load one (most load automatically when you use their commands)

`Find-Module name / Install-Module name -Scope CurrentUser` Search the PowerShell Gallery and install without administrator rights

`Get-InstalledModule` Modules installed from the Gallery

`Enter-PSSession -ComputerName pc` Interactive session on another Windows computer (needs remoting set up)

`Invoke-Command -ComputerName pc { ... }` Run code remotely

Linux and WSL Use `ssh` or `wsl` from PowerShell; PowerShell 7 can also remote over SSH

A REASONABLE STARTER PROFILE (MY SUGGESTIONS)

`Set-Alias ll Get-ChildItem` A short name for listing

`function mkcd($p){ New-Item $p -ItemType Directory -Force | Out-Null; Set-Location $p }` Make a folder and go into it

`function which($n){ (Get-Command $n).Source }` Where is this program?

`$PSDefaultParameterValues['Out-File:Encoding'] = 'utf8'` Every Out-File writes UTF-8 (5.1 adds a BOM)

`$env:PATH += 'C:\tools'` Add a folder to PATH for the session

`Set-PSReadLineOption -EditMode Windows` The default editing keys, stated explicitly

WHY DOES MY REDIRECTED FILE LOOK LIKE GARBAGE? ENCODING

In Windows PowerShell 5.1, command `> file.txt` and `Out-File` write **UTF-16** (with a byte-order mark), which many tools show as text with gaps between the letters. Use command `| Out-File file.txt -Encoding utf8` instead (utf8 in 5.1 adds a BOM) or `Set-Content -Encoding UTF8`. PowerShell 7 writes plain UTF-8 by default.

PROCESSES, SERVICES, NETWORK & WINDOW

`Get-Process chrome | Stop-Process` Find and

`Start-Process notepad -Verb RunAs` Start a p

`Get-Service / Start-Service / Stop-Service`

`Get-NetTCPConnection -State Listen` Which p

`Get-NetIPAddress` Your IP addresses

`Get-CimInstance Win32_OperatingSystem` Syst

`Get-WinEvent -LogName System -MaxEvents 20`

`Get-ChildItem Env:` List environment variables

`[Environment]::SetEnvironmentVariable('X',`

`Get-ItemProperty HKCU:\Software\...` Read the

`Get-Clipboard / Set-Clipboard` Read / write t

`Get-Date -Format 'yyyy-MM-dd'` Formatted da

`Start-Sleep -Seconds 5` Wait

`Start-Job { ... } / Get-Job / Receive-Job` R

`Get-ScheduledTask / Register-ScheduledTask`

WINDOWS POWERSHELL 5.1 VS POWERSHELL

`powershell.exe / pwsh.exe` 5.1 ships with Wind
installs beside it. 7

`Get it` `winget install Microsoft.PowerShe`
knowledge)

`&& and ||` Run the next command only if the last

`? : and ?? ?.` Ternary, null-coalescing and null-c

`ForEach-Object -Parallel` 7 only (the paramet

`Encoding` 5.1 writes UTF-16 with `>` and `Out-Fi`

`Predictions and colours` Inline command pred

SAFE HABITS FOR DELETING AND CHANGING THINGS

Put `-WhatIf` after `Remove-Item`, `Move-Item`, `Stop-Process` and similar commands to see what *would* happen, and `-Confirm` to be asked about each one. `Remove-Item` does not use the recycle bin, and `Remove-Item -Recurse -Force` on the wrong folder cannot be undone. Test a `Get-ChildItem` with the same path first.

"RUNNING SCRIPTS IS DISABLED ON THIS SYSTEM"

That is the execution policy, not a bug. The one-line fix for your own account is `Set-ExecutionPolicy RemoteSigned -Scope CurrentUser`, as explained on the Python Environments sheet. Your launchers use `-ExecutionPolicy Bypass` for a single run, which changes nothing permanently.