MÖHL TRAINING
PROFESSIONAL TRAINING SUPPORT

PowerShell-Glossar

Begriffe, Befehle und Konzepte – Handout mit 212 Einträgen

Dieses Handout fasst die wichtigsten Begriffe rund um PowerShell zusammen, von den Grundlagen bis zu Remoting, Microsoft 365 und PowerShell 7. Befehle und Code sind in `Konsolenschrift` hervorgehoben.

Drei Befehle helfen fast immer weiter: `Get-Help` erklärt einen Befehl, `Get-Command` findet ihn, und `Get-Member` zeigt, was in einem Ergebnis steckt.

Inhalt

1. Grundbegriffe (10)
2. Hilfe und Erkundung (8)
3. Befehle und Module (15)
4. Objekte und Pipeline (8)
5. Objekte verarbeiten (11)
6. Variablen und Datentypen (14)
7. Operatoren (10)
8. Zeichenketten und reguläre Ausdrücke (10)
9. Ablaufsteuerung (8)
10. Fehlerbehandlung und Debugging (10)
11. Dateisystem und Navigation (15)
12. Ein- und Ausgabe, Formatierung und Export (18)
13. Skripte und Funktionen (13)
14. Netzwerk und Web (8)
15. System und Verwaltung (19)
16. Benutzer und Cloud-Verwaltung (8)
17. Sicherheit (11)
18. Neuerungen in PowerShell 7 (8)
19. Werkzeuge rund um PowerShell (8)

→ Schnellreferenz: Aliase und bekannte Befehle

1. Grundbegriffe

Begriff	Erklärung
PowerShell	Shell und Skriptsprache von Microsoft, objektorientiert und auf .NET aufgebaut. Es gibt `Windows PowerShell 5.1` (in Windows enthalten) und das plattformübergreifende `PowerShell 7.x` (Programmname `pwsh`).
Shell	Ein Programm, das Befehle entgegennimmt und ausführt. Weitere Beispiele sind cmd (Eingabeaufforderung) und Bash unter Linux.
.NET	Die Microsoft-Programmierplattform, auf der PowerShell aufbaut. Dadurch stehen in PowerShell auch alle .NET-Klassen zur Verfügung, z. B. `[System.Math]::Round(3.7)`.
Konsole / Host	Das Programm, in dem PowerShell läuft, z. B. Windows Terminal, die klassische Konsole oder VS Code.
Windows Terminal	Moderne Terminal-Anwendung von Microsoft mit Tabs, Profilen und guter Schriftdarstellung. Ab Windows 11 Standard.
Sitzung (Session)	Eine laufende PowerShell-Instanz. Variablen, geladene Module und Funktionen gelten nur bis zum Schließen der Sitzung.
ISE	Integrated Scripting Environment: der ältere Editor für Windows PowerShell. Er wird nicht mehr weiterentwickelt; als Ersatz dient VS Code mit der PowerShell-Erweiterung.
Prompt	Die Eingabezeile, z. B. `PS C:\>`. Sie lässt sich über die Funktion `prompt` anpassen.
Skript	Eine Textdatei mit PowerShell-Befehlen und der Endung `.ps1`. Aufruf mit `.\skript.ps1`.
Groß-/Kleinschreibung	PowerShell unterscheidet bei Befehlen, Parametern und Variablen nicht zwischen Groß- und Kleinschreibung. `get-process` funktioniert genauso wie `Get-Process`.

2. Hilfe und Erkundung

Begriff	Erklärung
`Get-Help`	Zeigt die Hilfe zu einem Befehl. Mit `-Examples` nur Beispiele, mit `-Full` alles, mit `-Online` im Browser.
`Update-Help`	Lädt die aktuellen Hilfetexte herunter. Einmalig als Administrator ausführen, sonst ist die Hilfe nur rudimentär.
`Get-Command`	Findet Befehle nach Stichwort, z. B. `Get-Command *service*` oder `Get-Command -Module NetTCPIP`.
`Get-Member`	Zeigt alle Eigenschaften und Methoden eines Objekts (Alias `gm`). Das wichtigste Werkzeug, um herauszufinden, „was drinsteckt“.
`Get-Alias`	Zeigt, für welchen Befehl ein Alias steht, z. B. `Get-Alias ls`.
`Show-Command`	Öffnet ein Fenster, in dem man die Parameter eines Befehls per Formular ausfüllen kann (nur Windows). Gut für Einsteiger.
about-Hilfethemen	Hintergrundartikel zu Sprachkonzepten, z. B. `Get-Help about_Operators` oder `Get-Help about_*` für die Übersicht.
`$PSVersionTable`	Zeigt die installierte PowerShell-Version und weitere Umgebungsinformationen.

3. Befehle und Module

Begriff	Erklärung
Cmdlet	Ein eingebauter PowerShell-Befehl im Schema Verb-Nomen, z. B. `Get-Process`. Ausgesprochen „Command-let“.
Verb-Nomen-Syntax	Die Namenskonvention für Befehle: Das Verb sagt, was passiert (`Get`, `Set`, `New`, `Remove`), das Nomen, worauf es wirkt. `Get-Verb` listet die zulässigen Verben.
Parameter	Eine Option eines Befehls, die mit Bindestrich beginnt, z. B. `-Path` oder `-Name`. Parameternamen dürfen abgekürzt werden, solange sie eindeutig bleiben.
Switch-Parameter	Ein Parameter ohne Wert, der nur ein- oder ausgeschaltet wird, z. B. `-Recurse` oder `-Force`.
Positionaler Parameter	Ein Parameter, dessen Name weggelassen werden darf, weil seine Position feststeht: `Get-ChildItem C:\Temp` entspricht `Get-ChildItem -Path C:\Temp`.
Parametersatz (Parameter Set)	Eine zulässige Kombination von Parametern. Manche Parameter schließen sich gegenseitig aus; `Get-Help` zeigt die Sätze im Abschnitt SYNTAX.
Allgemeine Parameter (Common Parameters)	Parameter, die jedes Cmdlet kennt, z. B. `-Verbose`, `-ErrorAction`, `-WarningAction`, `-OutVariable`.
Alias	Ein Kurzname für einen Befehl, z. B. `ls` oder `dir` für `Get-ChildItem`. Eigene Aliase mit `Set-Alias`. In Skripten besser die vollen Namen verwenden.
Funktion	Ein selbst definierter, wiederverwendbarer Befehlsblock, angelegt mit `function Name { … }`.
Modul	Ein Paket aus Cmdlets und Funktionen, z. B. `Az` oder `Microsoft.Graph`. Laden mit `Import-Module`; PowerShell lädt Module beim ersten Befehlsaufruf meist automatisch.
`Get-Module`	Zeigt geladene Module; mit `-ListAvailable` alle installierten.
`Find-Module` / `Install-Module`	Sucht bzw. installiert Module aus der PowerShell Gallery. Mit `-Scope CurrentUser` ohne Administratorrechte.
PSResourceGet	Der modernere Nachfolger von PowerShellGet mit Befehlen wie `Find-PSResource` und `Install-PSResource`. In aktuellen PowerShell-7-Versionen enthalten.
PowerShell Gallery	Das zentrale Online-Repository für Module und Skripte (powershellgallery.com).
`$env:PSModulePath`	Liste der Ordner, in denen PowerShell nach Modulen sucht.

4. Objekte und Pipeline

Begriff	Erklärung
Objekt	PowerShell gibt Objekte mit Eigenschaften und Methoden zurück, nicht reinen Text. Das ist der Hauptunterschied zu cmd oder Bash.
Eigenschaft (Property)	Ein Datenfeld eines Objekts, z. B. `.Name` oder `.Length`. Zugriff mit Punkt: `(Get-Item datei.txt).Length`.
Methode	Eine Aktion, die ein Objekt ausführen kann, z. B. `"text".ToUpper()`.

Begriff	Erklärung
Statische Methode	Eine Methode einer .NET-Klasse, aufgerufen mit `::`, z. B. `[DateTime]::Now` oder `[Math]::Sqrt(16)`.
Pipeline (|)	Gibt die Objekte eines Befehls an den nächsten weiter: `Get-Process | Sort-Object CPU`.
`$_` / `$PSItem`	Das aktuelle Objekt in der Pipeline, z. B. in `Where-Object { $_.CPU -gt 100 }`.
Pipeline-Bindung	Wie PowerShell Pipeline-Objekte an Parameter übergibt: ByValue (das ganze Objekt passt zum Parametertyp) oder ByPropertyName (eine Eigenschaft heißt so wie der Parameter).
„Filter left, format right“	Faustregel: so früh wie möglich filtern (am besten mit `-Filter` des ersten Befehls), Formatierung erst ganz am Ende.

5. Objekte verarbeiten

Begriff	Erklärung
`Where-Object`	Filtert Objekte nach einer Bedingung (Aliase `where`, `?`). Kurzform: `Where-Object CPU -gt 100`.
`Select-Object`	Wählt Eigenschaften aus oder begrenzt die Anzahl: `Select-Object Name, CPU -First 5`.
`ForEach-Object`	Führt einen Skriptblock für jedes Pipeline-Objekt aus (Aliase `foreach`, `%`).
`Sort-Object`	Sortiert Objekte, mit `-Descending` absteigend, mit `-Unique` ohne Duplikate.
`Group-Object`	Gruppiert Objekte nach einer Eigenschaft, z. B. Dateien nach Endung.
`Measure-Object`	Zählt Objekte oder berechnet Summe, Durchschnitt, Minimum und Maximum.
`Compare-Object`	Vergleicht zwei Sammlungen und zeigt die Unterschiede.
`Tee-Object`	Speichert die Pipeline-Daten zusätzlich in einer Datei oder Variable und reicht sie trotzdem weiter.
Berechnete Eigenschaft	Eine selbst definierte Spalte in `Select-Object`: `@{Name='MB'; Expression={$_.Length/1MB}}`.
`-ExpandProperty`	Gibt bei `Select-Object` nur den Wert einer Eigenschaft zurück statt eines Objekts.
`[PSCustomObject]`	Erstellt ein eigenes Objekt aus einer Hashtable: `[PSCustomObject]@{ Name='Kurs'; Teilnehmer=12 }`. Ideal für Berichte.

6. Variablen und Datentypen

Begriff	Erklärung
Variable	Ein Speicher für Werte, beginnt immer mit `$`, z. B. `$name = "Kurt"`.
Automatische Variable	Von PowerShell vorbelegte Variablen wie `$true`, `$false`, `$null`, `$HOME` oder `$PSVersionTable`.
`$PSScriptRoot`	Der Ordner, in dem das laufende Skript liegt. Nützlich für Pfade zu Dateien neben dem Skript.
`$args`	Enthält die an ein Skript oder eine Funktion übergebenen Argumente, wenn kein `param()`-Block definiert ist.

Begriff	Erklärung
`$?` / `$LASTEXITCODE`	`$?` zeigt, ob der letzte Befehl erfolgreich war (`$true`/`$false`); `$LASTEXITCODE` enthält den Rückgabecode des letzten externen Programms.
Umgebungsvariable	Wird mit `$env:` angesprochen, z. B. `$env:USERNAME`, `$env:COMPUTERNAME` oder `$env:PATH`.
Array	Eine Liste von Werten: `@(1, 2, 3)`. Zugriff per Index ab 0: `$liste[0]`, letztes Element `$liste[-1]`.
Hashtable	Schlüssel-Wert-Paare: `@{ Name = "Kurt"; Ort = "Wien" }`. Zugriff mit `$h.Name` oder `$h["Name"]`.
Geordnete Hashtable	`[ordered]@{ … }` behält die Reihenfolge der Schlüssel bei, was bei Ausgaben und Exporten wichtig ist.
Datentyp-Cast	Erzwingt einen Typ, z. B. `[int]"42"` oder `[datetime]"2026-11-08"`. Typisierte Variable: `[int]$zahl = 5`.
Typbeschleuniger	Kurznamen für .NET-Typen wie `[string]`, `[int]`, `[bool]`, `[array]`, `[hashtable]`, `[xml]`, `[ipaddress]`.
Enum	Ein Aufzählungstyp mit festen benannten Werten: `enum Ampel { Rot; Gelb; Gruen }`.
Scope	Der Gültigkeitsbereich einer Variablen: `global`, `script`, `local`, z. B. `$script:zaehler`.
`Get-Variable`	Listet alle Variablen der Sitzung auf; `Remove-Variable` löscht eine.

7. Operatoren

Begriff	Erklärung
Vergleichsoperatoren	`-eq` (gleich), `-ne` (ungleich), `-gt`/`-lt` (größer/kleiner), `-ge`/`-le` (größer/kleiner gleich). Statt `=` oder `>` werden diese verwendet.
Groß-/Kleinschreibung bei Vergleichen	Vergleiche ignorieren standardmäßig die Schreibweise. Mit vorangestelltem c wird unterschieden: `-ceq`, `-clike`, `-cmatch`.
`-like` / `-match`	Mustervergleich: `-like` mit Platzhaltern (`*`, `?`), `-match` mit regulären Ausdrücken.
`-in` / `-contains`	Prüft, ob ein Wert in einer Liste vorkommt: `"Wien" -in $orte` bzw. `$orte -contains "Wien"`.
`-is` / `-as`	Prüft den Typ eines Werts (`5 -is [int]`) bzw. wandelt ihn um, ohne bei Fehlern abzubrechen.
Logische Operatoren	`-and`, `-or`, `-xor`, `-not` (bzw. `!`).
Arithmetische Operatoren	`+`, `-`, `*`, `/` und `%` (Rest der Division). Größeneinheiten werden direkt verstanden: `1KB`, `1MB`, `1GB`.
Zuweisungsoperatoren	`=`, `+=`, `-=`, `++` usw., z. B. `$summe += 5`.
Bereichsoperator	Erzeugt eine Zahlenfolge: `1..10`.
Umleitung	`>` schreibt in eine Datei, `>>` hängt an, `2>` leitet Fehler um.

8. Zeichenketten und reguläre Ausdrücke

Begriff	Erklärung
Einfache vs. doppelte Anführungszeichen	In `"Hallo $name"` wird die Variable eingesetzt, in `'Hallo $name'` bleibt der Text wörtlich stehen.
Subexpression `$( )`	Wertet einen Ausdruck innerhalb eines Strings aus: `"Heute ist $(Get-Date -Format d)"`.
Here-String	Ein mehrzeiliger Text zwischen `@"` und `"@` (bzw. `@'` und `'@`). Die Endmarke muss am Zeilenanfang stehen.
Format-Operator `-f`	Füllt Platzhalter: `"{0} hat {1} Dateien" -f $name, $anzahl`.
Backtick (`)	Das Escape-Zeichen in PowerShell, z. B. `` `n `` für einen Zeilenumbruch, `` `t `` für einen Tabulator oder am Zeilenende für die Fortsetzung in der nächsten Zeile.
`-replace` / `-split` / `-join`	Ersetzt Text (mit Regex), teilt einen String in ein Array bzw. fügt ein Array zu einem String zusammen.
String-Methoden	Nützliche .NET-Methoden wie `.Trim()`, `.ToUpper()`, `.Substring(0,3)`, `.StartsWith("A")`, `.Contains("x")`.
Regulärer Ausdruck (Regex)	Ein Suchmuster für Text, z. B. `'^\d{4}$'` für genau vier Ziffern. Wird von `-match`, `-replace`, `-split` und `Select-String` genutzt.
`$Matches`	Enthält nach einem erfolgreichen `-match` die gefundenen Treffer und Gruppen.
`Select-String`	Durchsucht Text oder Dateien nach Mustern, ähnlich wie grep oder findstr: `Select-String -Path *.log -Pattern "Fehler"`.

9. Ablaufsteuerung

Begriff	Erklärung
Skriptblock	Ein Codeblock in geschweiften Klammern `{ … }`, der an Befehle übergeben oder gespeichert werden kann.
`if` / `elseif` / `else`	Bedingte Ausführung.
`switch`	Mehrfachauswahl anhand eines Werts; unterstützt auch `-Wildcard` und `-Regex`.
`foreach`	Schleife über eine Sammlung: `foreach ($datei in $dateien) { … }`. Nicht zu verwechseln mit `ForEach-Object` in der Pipeline.
`for`	Zählschleife: `for ($i = 1; $i -le 10; $i++) { … }`.
`while` / `do … until`	Wiederholt, solange eine Bedingung gilt bzw. bis sie erfüllt ist.
`break` / `continue`	Bricht eine Schleife ab bzw. springt zum nächsten Durchlauf.
`return` / `exit`	Verlässt eine Funktion bzw. beendet ein Skript, optional mit Rückgabecode (`exit 1`).

10. Fehlerbehandlung und Debugging

Begriff	Erklärung
`try` / `catch` / `finally`	Fehlerbehandlung: `try` enthält den riskanten Code, `catch` reagiert auf Fehler, `finally` läuft immer.

Begriff	Erklärung
Terminierender Fehler	Bricht die Ausführung ab und kann mit `catch` abgefangen werden. Nicht-terminierende Fehler laufen weiter, außer man setzt `-ErrorAction Stop`.
`-ErrorAction`	Legt pro Befehl fest, was bei Fehlern passiert: `Stop`, `Continue`, `SilentlyContinue`, `Inquire`.
`$ErrorActionPreference`	Legt das Fehlerverhalten für alle folgenden Befehle fest, z. B. `'Stop'` am Skriptanfang.
`$Error`	Liste der zuletzt aufgetretenen Fehler; `$Error[0]` ist der neueste.
`throw`	Löst selbst einen Fehler aus: `throw "Datei nicht gefunden"`.
`-WhatIf` / `-Confirm`	Zeigt, was passieren würde, bzw. fragt vor der Ausführung nach. Ideal zum gefahrlosen Testen.
`Set-StrictMode`	Macht PowerShell strenger, z. B. Fehler bei nicht definierten Variablen: `Set-StrictMode -Version Latest`.
`Write-Verbose` / `Write-Debug`	Zusätzliche Meldungen, die nur mit `-Verbose` bzw. `-Debug` erscheinen.
Haltepunkt (Breakpoint)	Hält ein Skript an einer Stelle an, um Variablen zu untersuchen. In VS Code per Klick neben der Zeilennummer, in der Konsole mit `Set-PSBreakpoint`.

11. Dateisystem und Navigation

Begriff	Erklärung
`Get-ChildItem`	Listet Dateien und Ordner auf (Aliase `ls`, `dir`, `gci`). Mit `-Recurse` inklusive Unterordner, mit `-Filter *.txt` gefiltert, mit `-File`/`-Directory` nur Dateien bzw. Ordner.
`Set-Location` / `Get-Location`	Wechselt das Verzeichnis (`cd`) bzw. zeigt das aktuelle an (`pwd`).
`Push-Location` / `Pop-Location`	Merkt sich das aktuelle Verzeichnis und kehrt später dorthin zurück.
`Copy-Item` / `Move-Item` / `Remove-Item`	Kopieren, Verschieben und Löschen von Dateien und Ordnern.
`Rename-Item`	Benennt eine Datei oder einen Ordner um.
`New-Item`	Legt Dateien oder Ordner an: `New-Item -ItemType Directory -Name Test`.
`Get-Item` / `Get-ItemProperty`	Ruft ein einzelnes Element bzw. dessen Eigenschaften ab (auch in der Registry).
`Get-Content` / `Set-Content` / `Add-Content`	Liest den Inhalt einer Textdatei, schreibt ihn neu oder hängt etwas an. `-Tail 20` zeigt die letzten Zeilen.
`Test-Path`	Prüft, ob ein Pfad existiert, und gibt `$true` oder `$false` zurück.
`Join-Path` / `Split-Path`	Setzt Pfade sicher zusammen bzw. zerlegt sie in Ordner und Dateiname.
`Compress-Archive` / `Expand-Archive`	Erstellt ZIP-Archive bzw. entpackt sie.
`Get-FileHash`	Berechnet die Prüfsumme einer Datei (Standard SHA256), etwa um Downloads zu verifizieren.
`Get-Acl` / `Set-Acl`	Liest bzw. setzt die Zugriffsrechte (NTFS-Berechtigungen) einer Datei oder eines Ordners.

Begriff	Erklärung
Relativer / absoluter Pfad	`.\datei.txt` bezieht sich auf das aktuelle Verzeichnis, `C:\Ordner\datei.txt` ist der vollständige Pfad. `..` steht für den übergeordneten Ordner, `~` für das Benutzerverzeichnis.
Platzhalter (Wildcards)	`*` steht für beliebig viele Zeichen, `?` für genau eines.

12. Ein- und Ausgabe, Formatierung und Export

Begriff	Erklärung
`Format-Table` / `Format-List`	Zeigt Objekte als Tabelle (`ft`) oder als Liste (`fl`). Gehört immer ans Ende der Pipeline, weil danach nur noch Formatierungsdaten übrig sind.
`Format-Wide`	Zeigt nur eine Eigenschaft in mehreren Spalten, z. B. Dateinamen.
`Out-GridView`	Öffnet die Ausgabe in einem filterbaren Fenster (nur Windows). Mit `-PassThru` lassen sich Zeilen auswählen und weitergeben.
`Out-File`	Schreibt die Ausgabe in eine Datei; `-Encoding utf8` legt die Kodierung fest.
`Out-Null`	Verwirft die Ausgabe, wenn man sie nicht sehen will.
`Export-Csv` / `Import-Csv`	Speichert Objekte als CSV bzw. liest CSV-Dateien als Objekte ein. Für deutschsprachiges Excel `-Delimiter ';'` oder `-UseCulture` verwenden.
`ConvertTo-Json` / `ConvertFrom-Json`	Wandelt Objekte in JSON um und umgekehrt, etwa für APIs und Konfigurationsdateien. Bei verschachtelten Daten `-Depth` erhöhen.
`ConvertTo-Html`	Erzeugt aus Objekten eine HTML-Tabelle, z. B. für einfache Berichte.
`Export-Clixml` / `Import-Clixml`	Speichert Objekte vollständig mit Typinformationen als XML und stellt sie wieder her.
`Write-Host`	Schreibt (farbigen) Text direkt auf den Bildschirm, aber nicht in die Pipeline. Für Ergebnisse besser `Write-Output`.
`Write-Output`	Gibt Objekte in die Pipeline aus. Das ist das Standardverhalten, wenn man einfach einen Wert hinschreibt.
`Write-Warning` / `Write-Error`	Gibt eine Warnung bzw. eine Fehlermeldung über den jeweiligen Ausgabestrom aus.
`Write-Progress`	Zeigt einen Fortschrittsbalken bei länger laufenden Skripten.
Ausgabeströme	PowerShell hat mehrere Kanäle: Output, Error, Warning, Verbose, Debug und Information. Sie lassen sich getrennt umleiten.
`Read-Host`	Fragt eine Eingabe vom Benutzer ab: `$name = Read-Host "Ihr Name"`.
`Get-Clipboard` / `Set-Clipboard`	Liest aus der Zwischenablage bzw. schreibt hinein: `Get-Process | Set-Clipboard`.
`Clear-Host`	Leert den Bildschirm (Aliase `cls`, `clear`).
`Start-Transcript`	Protokolliert die gesamte Sitzung in eine Textdatei, bis `Stop-Transcript` aufgerufen wird. Praktisch für Schulungen.

13. Skripte und Funktionen

Begriff	Erklärung
`param()`-Block	Definiert die Parameter eines Skripts oder einer Funktion, inklusive Typen und Standardwerten.

Begriff	Erklärung
Advanced Function	Eine Funktion mit `[CmdletBinding()]`, die sich wie ein echtes Cmdlet verhält (inklusive `-Verbose`, `-WhatIf` und Pipeline-Unterstützung).
Parametervalidierung	Attribute wie `[Parameter(Mandatory)]`, `[ValidateSet('A','B')]` oder `[ValidateRange(1,10)]` prüfen Eingaben automatisch.
`begin` / `process` / `end`	Blöcke einer Funktion: `begin` läuft einmal am Anfang, `process` für jedes Pipeline-Objekt, `end` einmal am Schluss.
Kommentarbasierte Hilfe	Ein Kommentarblock mit `.SYNOPSIS`, `.DESCRIPTION`, `.EXAMPLE`, damit `Get-Help` auch für eigene Funktionen funktioniert.
Kommentar	`#` für eine Zeile, `<# … #>` für einen Block.
Splatting	Übergibt Parameter gesammelt als Hashtable mit `@`: `Copy-Item @params`. Macht lange Befehle lesbar.
Dot-Sourcing	`. .\skript.ps1` lädt ein Skript in den aktuellen Scope, sodass dessen Funktionen und Variablen danach verfügbar sind.
Aufrufoperator `&`	Führt einen Befehl oder Pfad aus, der als String vorliegt: `& "C:\Program Files\tool.exe"`.
`#Requires`	Legt Voraussetzungen fest, z. B. `#Requires -RunAsAdministrator`, `-Version 7` oder `-Modules ActiveDirectory`.
Klasse (class)	Eigene Objekttypen mit Eigenschaften und Methoden: `class Kurs { [string]$Titel }`. Verfügbar ab PowerShell 5.
Modulmanifest (`.psd1`)	Die Beschreibungsdatei eines Moduls mit Version, Autor und exportierten Funktionen. Der Code steckt in einer `.psm1`-Datei.
Profil	Ein Skript, das beim Start automatisch geladen wird; der Pfad steht in `$PROFILE`. Dort legt man eigene Aliase und Funktionen ab. Öffnen mit `notepad $PROFILE`.

14. Netzwerk und Web

Begriff	Erklärung
`Test-Connection`	Der PowerShell-Ping: prüft, ob ein Rechner erreichbar ist.
`Test-NetConnection`	Prüft Erreichbarkeit inklusive Port: `Test-NetConnection moehl.at -Port 443` (Alias `tnc`, nur Windows).
`Resolve-DnsName`	Fragt DNS-Einträge ab, ähnlich nslookup (nur Windows).
`Get-NetIPAddress` / `Get-NetIPConfiguration`	Zeigt IP-Adressen und Netzwerkkonfiguration, vergleichbar mit ipconfig.
`Get-NetAdapter`	Listet die Netzwerkadapter mit Status und Geschwindigkeit.
`Invoke-WebRequest`	Ruft eine Webseite oder Datei ab (Alias `iwr`), z. B. zum Herunterladen mit `-OutFile`.
`Invoke-RestMethod`	Ruft eine REST-API auf und wandelt JSON-Antworten automatisch in Objekte um (Alias `irm`).
REST-API	Eine Web-Schnittstelle, die Daten meist als JSON über HTTP-Methoden wie GET, POST, PUT und DELETE austauscht.

15. System und Verwaltung

Begriff	Erklärung
`Get-Service` / `Start-Service` / `Stop-Service`	Verwaltet Windows-Dienste; `Restart-Service` startet neu, `Set-Service` ändert den Starttyp.
`Get-Process` / `Stop-Process`	Zeigt laufende Prozesse an bzw. beendet sie.
`Start-Process`	Startet ein Programm, mit `-Verb RunAs` als Administrator, mit `-Wait` wartend.
`Get-ComputerInfo`	Liefert umfangreiche Informationen zu Betriebssystem und Hardware.
CIM / WMI	Schnittstellen für Systeminformationen, z. B. `Get-CimInstance Win32_OperatingSystem`. CIM ist der moderne Nachfolger von WMI; `Get-WmiObject` gibt es in PowerShell 7 nicht mehr.
`Get-WinEvent`	Liest die Windows-Ereignisprotokolle, z. B. `Get-WinEvent -LogName System -MaxEvents 20`.
`Get-HotFix`	Listet installierte Windows-Updates.
`Get-Date`	Liefert Datum und Uhrzeit; mit `-Format "dd.MM.yyyy"` formatiert. `(Get-Date).AddDays(-7)` rechnet mit Daten.
`New-TimeSpan`	Berechnet Zeitspannen zwischen zwei Zeitpunkten.
`Measure-Command`	Misst, wie lange ein Befehl oder Skriptblock braucht.
`Start-Sleep`	Pausiert für eine bestimmte Zeit: `Start-Sleep -Seconds 5`.
`Get-Random`	Erzeugt Zufallszahlen oder wählt zufällige Elemente aus einer Liste, z. B. für Gruppeneinteilungen.
`Restart-Computer` / `Stop-Computer`	Startet den Rechner neu bzw. fährt ihn herunter (auch für entfernte Rechner).
Provider / PSDrive	Macht Datenquellen wie Laufwerke ansprechbar, z. B. die Registry (`HKCU:`, `HKLM:`), Umgebungsvariablen (`Env:`) oder Zertifikate (`Cert:`). `Get-PSDrive` zeigt alle.
Job	Eine Aufgabe, die im Hintergrund läuft: `Start-Job`, `Get-Job`, `Receive-Job`.
Geplante Aufgabe	Ein Skript zeitgesteuert ausführen, mit `Register-ScheduledTask` (Aufgabenplanung).
Remoting	Befehle auf entfernten Rechnern ausführen, interaktiv mit `Enter-PSSession` oder für viele Rechner gleichzeitig mit `Invoke-Command`.
WinRM	Der Windows-Dienst, auf dem klassisches PowerShell-Remoting basiert; aktivieren mit `Enable-PSRemoting`. PowerShell 7 kann alternativ SSH nutzen.
Als Administrator ausführen	Erhöhte Rechte, die für viele Systembefehle nötig sind. Rechtsklick auf PowerShell bzw. Windows Terminal und „Als Administrator ausführen“.

16. Benutzer und Cloud-Verwaltung

Begriff	Erklärung
`Get-LocalUser` / `New-LocalUser`	Verwaltet lokale Benutzerkonten auf einem Windows-Rechner.
`Get-LocalGroupMember`	Zeigt die Mitglieder einer lokalen Gruppe, z. B. `Get-LocalGroupMember Administratoren`.

Begriff	Erklärung
ActiveDirectory-Modul	Befehle für das lokale Active Directory, z. B. `Get-ADUser`, `New-ADUser`, `Get-ADGroupMember`. Teil der RSAT-Tools.
Microsoft Graph PowerShell	Das aktuelle Modul für Microsoft 365 und Entra ID: `Connect-MgGraph`, `Get-MgUser`. Ersetzt die veralteten Module AzureAD und MSOnline.
ExchangeOnlineManagement	Modul zur Verwaltung von Exchange Online: `Connect-ExchangeOnline`, `Get-Mailbox`.
MicrosoftTeams-Modul	Verwaltung von Teams, Kanälen und Richtlinien: `Connect-MicrosoftTeams`.
Az-Modul	Verwaltung von Microsoft Azure: `Connect-AzAccount`, `Get-AzVM`.
Scopes (Graph)	Berechtigungen, die beim Verbinden mit Microsoft Graph angefordert werden: `Connect-MgGraph -Scopes "User.Read.All"`.

17. Sicherheit

Begriff	Erklärung
Execution Policy	Regelt, ob Skripte ausgeführt werden dürfen, z. B. `Restricted`, `RemoteSigned`, `AllSigned`, `Unrestricted`. Abfragen mit `Get-ExecutionPolicy -List`. Sie ist eine Schutzvorrichtung gegen Versehen, keine echte Sicherheitsgrenze.
`-Scope` bei der Execution Policy	Die Richtlinie lässt sich nur für den aktuellen Benutzer (`CurrentUser`) oder nur für die laufende Sitzung (`Process`) ändern, ohne Administratorrechte.
`Unblock-File`	Entfernt die Markierung „aus dem Internet heruntergeladen“, wegen der eine Datei unter `RemoteSigned` blockiert wird.
Codesignatur	Skripte mit einem Zertifikat signieren (`Set-AuthenticodeSignature`), damit sie unter `AllSigned` laufen.
`Get-Credential`	Fragt Benutzername und Passwort sicher ab und liefert ein `PSCredential`-Objekt.
SecureString	Ein geschützt im Speicher gehaltener String für Passwörter, z. B. von `Read-Host -AsSecureString`.
SecretManagement	Module (`Microsoft.PowerShell.SecretManagement` und `SecretStore`), um Passwörter und API-Schlüssel sicher zu speichern statt im Skript.
`Invoke-Expression`	Führt einen String als Code aus. Gilt als Sicherheitsrisiko, weil fremder Text als Befehl laufen kann; möglichst vermeiden.
Constrained Language Mode	Ein eingeschränkter Sprachmodus, der z. B. durch AppLocker oder WDAC erzwungen wird und gefährliche Funktionen sperrt.
Skriptblockprotokollierung	Protokolliert ausgeführten PowerShell-Code im Ereignisprotokoll; per Gruppenrichtlinie aktivierbar und wichtig für die Sicherheitsüberwachung.
JEA (Just Enough Administration)	Gibt Benutzern per Remoting nur genau die Befehle frei, die sie brauchen.

18. Neuerungen in PowerShell 7

Begriff	Erklärung
Parallelinstallation	PowerShell 7 (`pwsh.exe`) wird neben Windows PowerShell 5.1 (`powershell.exe`) installiert und ersetzt es nicht. Installation z. B. mit `winget install Microsoft.PowerShell`.
Ternärer Operator	Kurzform für `if`/`else`: `$bedingung ? 'ja' : 'nein'`.
Null-Koaleszenz `??`	Liefert einen Ersatzwert, wenn etwas `$null` ist: `$name ?? 'Unbekannt'`. Mit `??=` wird nur zugewiesen, wenn die Variable leer ist.
Pipeline-Verkettung `&&` / `||`	Führt den zweiten Befehl nur bei Erfolg (`&&`) bzw. nur bei Fehler (`||`) des ersten aus.
`ForEach-Object -Parallel`	Verarbeitet Pipeline-Objekte parallel und damit deutlich schneller bei vielen unabhängigen Aufgaben.
`Get-Error`	Zeigt den letzten Fehler ausführlich mit allen Details an.
`$PSStyle`	Steuert Farben und Formatierung der Konsolenausgabe.
Windows-Kompatibilität	Ältere Windows-Module, die nur in 5.1 laufen, lassen sich mit `Import-Module -UseWindowsPowerShell` trotzdem nutzen.

19. Werkzeuge rund um PowerShell

Begriff	Erklärung
VS Code mit PowerShell-Erweiterung	Der empfohlene Editor für Skripte, mit IntelliSense, Debugger und integrierter Konsole.
PSReadLine	Das Modul hinter Befehlsverlauf, Syntaxfarben und Vorschlägen in der Konsole. `Strg+R` durchsucht den Verlauf, `F2` wechselt die Ansicht der Vorschläge.
Tab-Vervollständigung	Ergänzt Befehle, Parameter und Pfade mit der Tab-Taste; `Strg+Leertaste` zeigt alle Möglichkeiten.
`Get-History`	Zeigt die bisher eingegebenen Befehle der Sitzung; `Invoke-History` (Alias `r`) führt einen davon erneut aus.
Pester	Ein Framework für automatisierte Tests von PowerShell-Code.
PSScriptAnalyzer	Prüft Skripte auf Stilprobleme und Fehlerquellen (Linter), auch in VS Code integriert.
winget	Der Windows-Paketmanager, oft zusammen mit PowerShell verwendet, z. B. zur Installation von Programmen per Befehl.
Oh My Posh	Ein optionales Werkzeug für einen farbigen, informativen Prompt (z. B. mit Git-Status).

Schnellreferenz: Aliase und bekannte Befehle

Alias	Cmdlet	Entspricht in cmd / Bash
`ls`, `dir`, `gci`	`Get-ChildItem`	dir / ls
`cd`, `sl`	`Set-Location`	cd
`pwd`, `gl`	`Get-Location`	cd (ohne Argument) / pwd

Alias	Cmdlet	Entspricht in cmd / Bash
`cat`, `type`, `gc`	`Get-Content`	type / cat
`cp`, `copy`	`Copy-Item`	copy / cp
`mv`, `move`	`Move-Item`	move / mv
`rm`, `del`	`Remove-Item`	del / rm
`ren`	`Rename-Item`	ren / mv
`mkdir`, `md`	`New-Item -ItemType Directory`	mkdir
`cls`, `clear`	`Clear-Host`	cls / clear
`echo`	`Write-Output`	echo
`ps`, `gps`	`Get-Process`	tasklist / ps
`kill`	`Stop-Process`	taskkill / kill
`?`, `where`	`Where-Object`	–
`%`, `foreach`	`ForEach-Object`	for-Schleife
`select`	`Select-Object`	–
`sort`	`Sort-Object`	sort
`ft`, `fl`	`Format-Table`, `Format-List`	–
`gm`	`Get-Member`	–
`h`, `history`	`Get-History`	doskey /history / history
`iwr`, `irm`	`Invoke-WebRequest`, `Invoke-RestMethod`	curl / wget
`sls`	`Select-String`	findstr / grep

Hinweis: Unter Linux und macOS sind Aliase wie `ls`, `cat`, `cp` oder `ps` in PowerShell 7 nicht vorhanden, dort werden die gleichnamigen Systemprogramme aufgerufen.