# POWERSHELL RIFERIMENTO RAPIDO

Cmdlet, pipeline, oggetti, scripting, moduli

## Basi

### Comandi e Aiuto

```
Get-Help Get-Process          # mostra aiuto per il cmdlet
Get-Help Get-Process -Online # apre documentazione online
Get-Command *service*        # trova comandi per nome
Get-Alias ls                 # mostra destinazione alias
```

### Alias Comuni

| | |
|---|---|
| `ls → Get-ChildItem` | Elenca file e directory |
| `cd → Set-Location` | Cambia directory |
| `cp → Copy-Item` | Copia file o directory |
| `mv → Move-Item` | Sposta o rinomina |
| `rm → Remove-Item` | Elimina file o directory |
| `cat → Get-Content` | Legge il contenuto del file |
| `echo → Write-Output` | Stampa nella pipeline |
| `cls → Clear-Host` | Pulisce la console |

## Variabili

### Basi delle Variabili

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

### Variabili Automatiche

| | |
|---|---|
| `$_` | Oggetto pipeline corrente |
| `$PSVersionTable` | Info versione PowerShell |
| `$HOME` | Directory home dell'utente |
| `$PWD` | Directory corrente |
| `$null` | Valore null |
| `$true / $false` | Costanti booleane |
| `$Error` | Array degli errori recenti |
| `$LASTEXITCODE` | Codice di uscita dell'ultimo comando nativo |

### Variabili d'Ambiente

```
$env:PATH                # leggi variabile d'ambiente
$env:MY_VAR = "value"    # imposta variabile d'ambiente
Get-ChildItem Env:       # elenca tutte le variabili d'ambiente
```

## Operatori

### Operatori di Confronto

| | |
|---|---|
| `-eq  -ne` | Uguale / diverso |
| `-gt  -lt` | Maggiore / minore di |
| `-ge  -le` | Maggiore/minore o uguale |
| `-like  -notlike` | Corrispondenza wildcard (`*`, `?`) |
| `-match  -notmatch` | Corrispondenza regex |
| `-contains` | La collezione contiene il valore |
| `-in  -notin` | Valore nella collezione |

### Operatori Logici e Altri

| | |
|---|---|
| `-and  -or  -not` | Operatori logici |
| `!` | NOT logico (alias) |
| `-replace` | Sostituzione regex: `'hi' -replace 'h','b'` |
| `-split  -join` | Divide / unisce stringhe |
| `..` | Range: `1..5` → 1,2,3,4,5 |
| `?:` | Ternario (v7+): `$x ? 'yes' : 'no'` |

## Flusso di Controllo

### if / elseif / else

```
if ($age -ge 18) {
    "Adulto"
} elseif ($age -ge 13) {
    "Adolescente"
} else {
    "Bambino"
}
```

### switch

```
switch ($color) {
    "red"   { "Stop" }
    "green" { "Vai" }
    default { "Sconosciuto" }
}
```

### Cicli

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

## Funzioni

### Definire Funzioni

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

### Parametri Avanzati

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

## Oggetti e Pipeline

### Basi della Pipeline

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

### Cmdlet Pipeline Principali

| | |
|---|---|
| `Where-Object` | Filtra oggetti: `\| Where {$_.CPU -gt 10}` |
| `Select-Object` | Seleziona proprietà: `\| Select Name, CPU` |
| `Sort-Object` | Ordina: `\| Sort CPU -Desc` |
| `ForEach-Object` | Trasforma ogni elemento: `\| ForEach {$_.Name}` |
| `Measure-Object` | Conta, somma, media, min, max |
| `Group-Object` | Raggruppa per valore di proprietà |
| `Format-Table` | Visualizza come tabella: `\| ft -Auto` |
| `Export-Csv` | Esporta in CSV: `\| Export-Csv out.csv` |

## File

### Operazioni sui File

```
Get-Content log.txt                    # legge file
Set-Content out.txt "hello"            # scrive (sovrascrive)
Add-Content out.txt "more"             # aggiunge in fondo
Get-Content log.txt | Select-String "error"  # grep
```

### Cmdlet Percorso e File

| | |
|---|---|
| `Test-Path $path` | Verifica se file/dir esiste |
| `New-Item -Type File` | Crea file |
| `New-Item -Type Directory` | Crea directory |
| `Resolve-Path` | Ottieni percorso assoluto |
| `Join-Path` | Combina segmenti di percorso |
| `Split-Path` | Ottieni genitore o foglia |
| `Get-ItemProperty` | Attributi e metadati del file |
| `Remove-Item -Recurse` | Elimina directory in modo ricorsivo |

## Stringhe

### Basi delle Stringhe

```
"Hello, $name"               # interpolazione (doppi apici)
'Hello, $name'               # letterale (singoli apici)
"Length: $($list.Count)"      # espressione nella stringa
@"
Multi-riga
here-string con $name
"@
```

### Metodi sulle Stringhe

| | |
|---|---|
| `.ToUpper() / .ToLower()` | Cambia maiuscolo/minuscolo |
| `.Trim()` | Rimuovi spazi iniziali/finali |
| `.Split(',')` | Divide in array |
| `.Replace('a','b')` | Sostituisce sottostringa |
| `.StartsWith() / .EndsWith()` | Verifica prefisso / suffisso |
| `.Substring(0,5)` | Estrae sottostringa |
| `.Contains('text')` | Verifica se contiene |
| `-f operator` | `'{0} is {1}' -f 'sky','blue'` |

## Moduli

### Gestione dei Moduli

```
Get-Module -ListAvailable          # moduli installati
Find-Module -Name Az*              # cerca nella gallery
Install-Module -Name Pester        # installa dalla gallery
Import-Module ActiveDirectory      # carica modulo
```

### Comandi Modulo

| | |
|---|---|
| `Get-Module` | Elenca i moduli caricati |
| `Import-Module` | Carica il modulo nella sessione |
| `Remove-Module` | Scarica il modulo dalla sessione |
| `Update-Module` | Aggiorna il modulo installato |
| `Get-Command -Module X` | Elenca i comandi nel modulo |
| `$env:PSModulePath` | Percorsi di ricerca dei moduli |

## Pattern Comuni

### Gestione degli Errori

```
try {
    Get-Item "C:\missing" -ErrorAction Stop
} catch {
    "Errore: $_"
} finally {
    "Pulizia qui"
}
```

### Criteri di Esecuzione e Remoting

| | |
|---|---|
| `Get-ExecutionPolicy` | Mostra la policy degli script corrente |
| `Set-ExecutionPolicy RemoteSigned` | Consenti script locali |
| `Enter-PSSession -Computer SRV1` | Sessione remota interattiva |
| `Invoke-Command -Computer SRV1 -Script {}` | Esegui blocco script in remoto |
| `Start-Job { long-task }` | Esegui job in background |
| `Receive-Job -Id 1` | Ottieni output del job in background |