POWERSHELL 빠른 참조

Cmdlet, 파이프라인, 객체, 스크립팅, 모듈

기본

명령어 및 도움말

```
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
```

일반 별칭

`ls → Get-ChildItem`	파일 및 디렉터리 나열
`cd → Set-Location`	디렉터리 변경
`cp → Copy-Item`	파일 또는 디렉터리 복사
`mv → Move-Item`	이동 또는 이름 변경
`rm → Remove-Item`	파일 또는 디렉터리 삭제
`cat → Get-Content`	파일 내용 읽기
`echo → Write-Output`	파이프라인에 출력
`cls → Clear-Host`	콘솔 지우기

변수

변수 기본

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

자동 변수

`$_`	현재 파이프라인 객체
`$PSVersionTable`	PowerShell 버전 정보
`$HOME`	사용자 홈 디렉터리
`$PWD`	현재 디렉터리
`$null`	null 값
`$true / $false`	불리언 상수
`$Error`	최근 오류 배열
`$LASTEXITCODE`	마지막 네이티브 명령의 종료 코드

환경 변수

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

연산자

비교 연산자

`-eq  -ne`	같음 / 같지 않음
`-gt  -lt`	크다 / 작다
`-ge  -le`	크거나 같다 / 작거나 같다
`-like  -notlike`	와일드카드 매칭 (`*`, `?`)
`-match  -notmatch`	정규식 매칭
`-contains`	컬렉션에 값 포함 여부
`-in  -notin`	값이 컬렉션 내에 있는지

논리 및 기타 연산자

`-and  -or  -not`	논리 연산자
`!`	논리 NOT (별칭)
`-replace`	정규식 치환: `'hi' -replace 'h','b'`
`-split  -join`	문자열 분리 / 합치기
`..`	범위: `1..5` → 1,2,3,4,5
`?:`	삼항 (v7+): `$x ? 'yes' : 'no'`

제어 흐름

if / elseif / else

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

switch

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

반복문

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

함수

함수 정의

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

고급 파라미터

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

객체 및 파이프라인

파이프라인 기본

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

주요 파이프라인 Cmdlet

`Where-Object`	객체 필터링: `| Where {$_.CPU -gt 10}`
`Select-Object`	속성 선택: `| Select Name, CPU`
`Sort-Object`	정렬: `| Sort CPU -Desc`
`ForEach-Object`	각 변환: `| ForEach {$_.Name}`
`Measure-Object`	개수, 합계, 평균, 최솟값, 최댓값
`Group-Object`	속성 값으로 그룹화
`Format-Table`	테이블로 표시: `| ft -Auto`
`Export-Csv`	CSV로 내보내기: `| Export-Csv out.csv`

파일

파일 작업

```
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
```

경로 및 파일 Cmdlet

`Test-Path $path`	파일/디렉터리 존재 여부 확인
`New-Item -Type File`	파일 생성
`New-Item -Type Directory`	디렉터리 생성
`Resolve-Path`	절대 경로 가져오기
`Join-Path`	경로 세그먼트 합치기
`Split-Path`	부모 또는 리프 가져오기
`Get-ItemProperty`	파일 속성 및 메타데이터
`Remove-Item -Recurse`	디렉터리 재귀 삭제

문자열

문자열 기본

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

문자열 메서드

`.ToUpper() / .ToLower()`	대소문자 변환
`.Trim()`	앞뒤 공백 제거
`.Split(',')`	배열로 분리
`.Replace('a','b')`	부분 문자열 치환
`.StartsWith() / .EndsWith()`	접두사 / 접미사 확인
`.Substring(0,5)`	부분 문자열 추출
`.Contains('text')`	포함 여부 확인
`-f operator`	`'{0} is {1}' -f 'sky','blue'`

모듈

모듈 관리

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

모듈 명령어

`Get-Module`	로드된 모듈 나열
`Import-Module`	세션에 모듈 로드
`Remove-Module`	세션에서 모듈 언로드
`Update-Module`	설치된 모듈 업데이트
`Get-Command -Module X`	모듈의 명령어 나열
`$env:PSModulePath`	모듈 검색 경로

일반 패턴

오류 처리

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

실행 정책 및 원격

`Get-ExecutionPolicy`	현재 스크립트 정책 표시
`Set-ExecutionPolicy RemoteSigned`	로컬 스크립트 허용
`Enter-PSSession -Computer SRV1`	대화형 원격 세션
`Invoke-Command -Computer SRV1 -Script {}`	원격으로 스크립트 블록 실행
`Start-Job { long-task }`	백그라운드 작업 실행
`Receive-Job -Id 1`	백그라운드 작업 출력 가져오기