Grundlagen Zusammenfassung

Variablen & Strings

- Variablen starten mit `$` und werden mit `=` gesetzt → `$x = 5` (Typ automatisch)
- Zugriff immer mit `$`
- `"..."` → Variable wird **ersetzt** (ausgewertet) → `"Wert: $x"` → Wert: 5
- `'...'` → Variable bleibt **Text** → `'Wert: $x'` → Wert: $x
- `=` → Zuweisung | Vergleich z. B. mit `-eq`

□□ Merksatz:
Doppelte Anführungszeichen = Wert wird eingesetzt
Einfache Anführungszeichen = bleibt nur Text

Datentypen

- PowerShell ist **dynamisch typisiert** → Typ wird automatisch erkannt (`$x = 5`, `$y = "Text"`)
- Häufige Typen: String (`"Text"`), Int (`5`), Double (`3.14`), Boolean (`$true/$false`), Array (`1,2,3`)
- Typ kann **explizit gesetzt** werden → `[int]$x = 5`, `[string]$name = "Derick"`
- Typ prüfen → `$x.GetType()`

□□ Wichtig:

- Rechnen nur mit Zahlen sinnvoll (`"5" + 5` → kann unerwartet sein)
- Arrays werden mit Komma erstellt → `$arr = 1,2,3`

□□ Merksatz:
Typ kommt automatisch – kann aber festgelegt werden

- Ausgabe →
  - `Write-Host "Text"` (nur Anzeige, nicht weiter nutzbar)
  - `Write-Output "Text"` oder `"Text"` (**empfohlen, weiterverarbeitbar**)
- Eingabe → `$x = Read-Host "Frage"`
- Datum → `Get-Date`
- Konsole löschen → `Clear-Host` / `cls` (beides leert die Konsole)

□□ Beispiele anzeigen

```
# Ausgabe 'Write-Output'

# Variante 1 (offiziell)
Write-Output "Hallo"   # → Ausgabe: Hallo

# Variante 2 (Kurzform)
"Hallo"                # → Ausgabe: Hallo

# Konsole löschen 'cls'

# Variante 1 (voller Befehl)
Clear-Host             # → Konsole wird geleert

# Variante 2 (Kurzform / Alias)
cls                    # → Konsole wird geleert
```

Merksatz: **Write-Output besser als Write-Host | Read-Host Input einlesen | Cls entspricht Clear-Host**

Rechnen

- Grundrechenarten → `+` `-` `*` `/`
- Beispiele → `$x = 5 + 3` | `$y = 10 - 2` | `$z = 4 * 2` | `$a = 8 / 2`
- Punkt vor Strich gilt → `$r = 2 + 3 * 4` → Ergebnis: 14
- Klammern steuern Reihenfolge → `$r = (2 + 3) * 4` → Ergebnis: 20
- Zuweisung kombiniert → `$x += 5` (gleich `$x = $x + 5`)

Beispiel anzeigen

```
# Variante 1 (lang)
$x = $x + 5   # → erhöht $x um 5

# Variante 2 (kurz)
$x += 5       # → erhöht $x um 5
```

Merksatz: **Operatoren wie Mathe | Reihenfolge beachten | Klammern nutzen**

Operatoren

- Vergleich → `-eq` `-ne` `-gt` `-lt` → vergleichen Werte (True/False)
- Rechnen → `+` `-` `*` `/`
- Zuweisung → `=` `+=`
- String → `-like` (Wildcard) | `-match` (Regex)
- Listen → `-contains` (Liste enthält Wert) | `-in` (Wert in Liste)
- Logisch → `-and` `-or` `-not`
- Zusatz → `-not` (verneint) | `-c` (case-sensitive, z. B. `-ceq`)

Beispiele anzeigen

```
# Vergleich
5 -eq 5        # → True
5 -gt 3        # → True

# Rechnen
2 + 3          # → 5

# Zuweisung
$x = 5
$x += 2        # → 7

# String
"Test" -like "T*"    # → True
"Test" -match "es"   # → True

# Listen
1,2,3 -contains 2    # → True
2 -in 1,2,3          # → True

# Logisch
($true -and $false)  # → False

# Zusatz
5 -ne 3              # → True
```

```
"Test" -ceq "test"   # → False (Groß/Klein beachten)
```

PowerShell Verzweigungen (kurz & knapp)

- `if / else` → führt Code je nach Bedingung aus
- `switch` → prüft eine Variable gegen mehrere Fälle (übersichtlicher als viele ifs)

Beispiele anzeigen

```
# if / else
$x = 10

if ($x -gt 5) {
   "größer als 5"
} else {
   "kleiner oder gleich 5"
}

# switch
$farbe = "rot"

switch ($farbe) {
   "rot"   { "Stop" }
   "grün"  { "Go" }
   default { "Unbekannt" }
}
```

Merksatz: **if = einfache Entscheidung | switch = viele Fälle übersichtlich**

Revision #17
Created 20 April 2026 23:57:17 by Admin
Updated 21 April 2026 01:25:32 by Admin