**BATCH-Programmierung (keine powerShell)**

**1. Variablen**

* **Setzen:**

set "NAME=Wert" :: Sicher bei Leerzeichen

* **Lesen:**

echo %NAME%

* **Löschen:**

set NAME=

* **String-Operationen:**

%var:alt=neu% :: Ersetzen

%var:~start,len% :: Substring

**2. Parameter (%~)**

* %1 = erstes Argument
* %~1 = ohne Quotes
* %~dpnx1 = Drive, Path, Name, Extension
* **Modifikatoren:**

| **Code** | **Bedeutung** |
| --- | --- |
| d | Laufwerk |
| p | Pfad |
| n | Name |
| x | Extension |
| f | Vollständiger Pfad |

**3. IF-Konstrukte**

* **String-Vergleich:**

if "%var%"=="Text" echo OK

* **Existenz:**

if exist datei.txt echo Datei gefunden

* **Errorlevel:**

if errorlevel 1 echo Fehler

* **Block:**

if "%x%"=="1" (

echo Zeile 1

echo Zeile 2

)

**4. FOR-Schleifen**

* **Über Dateien:**

for %%F in (\*.txt) do echo %%F

* **Über Befehlsausgabe:**

for /f "delims=" %%D in ('dir /b') do echo %%D

* **Tokens & Delims:**

for /f "tokens=1,2 delims=," %%a in (datei.csv) do echo %%a %%b

**5. Steuerung**

* **Sprungmarken:**

:label

goto label

* **Beenden:**

exit /b 0 :: Erfolgreich

exit /b 1 :: Fehlercode

**6. Sonderzeichen escapen**

* ^ = Escape für &, |, >, (, )
* Beispiel:

echo ^(Test^)

**7. Delayed Expansion**

* Aktivieren:

setlocal enabledelayedexpansion

* Zugriff:

!var! :: statt %var%, wenn Wert sich in Schleife ändert

**8. Best Practices**

✔ Immer set "VAR=Wert" verwenden.  
✔ Quotes um Vergleiche: if "%var%"=="".  
✔ delims= bei for /f, um Leerzeichen zu behalten.  
✔ exit /b <code> für sauberes Beenden.

**✅ Delayed Expansion in Batch**

**Warum?**

* %VAR% wird **beim Parsen** des Skripts ersetzt, nicht zur Laufzeit.
* In Schleifen oder IF-Blöcken ändert sich der Wert oft **nach** dem Parsen → %VAR% zeigt alten Wert.
* Lösung: **Delayed Expansion** → !VAR! wird **zur Laufzeit** ausgewertet.

**Aktivieren**

setlocal enabledelayedexpansion

**Syntax**

* **Normal:** %VAR% (statisch, beim Parsen)
* **Delayed:** !VAR! (dynamisch, zur Laufzeit)

**Beispielproblem**

set COUNT=0

for %%i in (a b c) do (

set /a COUNT+=1

echo %COUNT% :: Gibt immer 0 aus!

)

**Lösung**

setlocal enabledelayedexpansion

set COUNT=0

for %%i in (a b c) do (

set /a COUNT+=1

echo !COUNT! :: Gibt 1, 2, 3 aus

)

**Best Practices**

✔ Immer setlocal enabledelayedexpansion am Anfang.  
✔ Nutze !VAR! nur, wenn sich der Wert in Schleifen/IF-Blöcken ändert.  
✔ Vorsicht: ! in Texten muss ggf. mit ^! escaped werden.

* Normalerweise speichert set alles als Text.
* Mit /a interpretiert set den Wert als **arithmetischen Ausdruck** und rechnet ihn aus.

**Beispiele**

set /a x=1+2

echo %x% :: Ausgabe: 3

set /a y=10/3

echo %y% :: Ausgabe: 3 (Ganzzahl-Division)

set /a z=(5\*2)+7

echo %z% :: Ausgabe: 17

**✅ Besonderheiten**

* Variablen können ohne % im Ausdruck stehen:

set /a count+=1

* Unterstützt Operatoren wie + - \* / % << >> & | ^ ~.
* Rückgabewert des Ausdrucks wird auch als **Errorlevel** gesetzt.

💡 **Merksatz:**  
set /a = „Rechne und speichere das Ergebnis“.