```
# einrichtFktn.ps1  (Version_0)
# 19.01.2026
# Funktionen für das Einrichten des Netzes
#
#   * schreibeMatKonfig    matrizen.kfg anlegen
#   * holeNetzKonfig       netzKonfig.txt auswerten
#   * erzeugeGewichte      gew0.mat.start, ... anlegen
# ================================================================
function schreibeMatKonfig {
  param([int[,]]$mat)

  # matrizen als Stringwerte schreiben
  #   matrizen[Rows, Columns] --> matKonfig.kfg
  # -----------------------------------------
  $content  = @()
  foreach($i in 0..($anzS-2)) {
    $content += "$($matrizen[$i,0]) $($matrizen[$i,1])"
  }
  Set-Content Daten/matKonfig.kfg $content
} # schreibeMatKonfig

# ----------------------------------------------------------------
function holeNetzKonfig {
  $anzS = $anzEN = $anzHN = $anzAN = -1     # noch unbelegt

  # netzKonfig.txt lesen
  # --------------------
  $datei = Get-Content ./Daten/netzKonfig.txt -ea Stop

  # Inhaltszeilen ausblenden
  # ------------------------
  $inhalt = @()
  foreach($zeile in $datei) {
    $tok = $zeile.split('=')
    if($tok.count -ne 2) { continue }

    # Es sind 2 Token
    # ---------------
    $tok[0] = $tok[0].trim()
    $tok[1] = $tok[1].trim()

    # Zeilenanfang prüfen
    # -------------------
    if( ($tok[0] -ne 'anzS' ) -and
        ($tok[0] -ne 'anzEN') -and
        ($tok[0] -ne 'anzHN') -and
        ($tok[0] -ne 'anzAN') )    { continue }

    # Wert abhängig vom Zeilenanfang prüfen
    # -------------------------------------
    if($tok[0] -eq 'anzS') {
      $val = 0
      if([int]::tryParse($tok[1],[ref]$val)) { $anzS = $val }
      else { throw 'Typfehler bei anzS' }
    }
    if($tok[0] -eq 'anzEN') {
      $val = 0
      if([int]::tryParse($tok[1],[ref]$val)) { $anzEN = $val }
      else { throw 'Typfehler bei anzEN' }
    }
```

```
    if($tok[0] -eq 'anzHN') {
      $val = 0
      if([int]::tryParse($tok[1],[ref]$val)) { $anzHN = $val }
      else { throw 'Typfehler bei anzHN' }
    }
    if($tok[0] -eq 'anzAN') {
      $val = 0
      if([int]::tryParse($tok[1],[ref]$val)) { $anzAN = $val }
      else { throw 'Typfehler bei anzAN' }
    }
  } # foreach(zeile)

  # Konsistenzprüfung
  # -----------------
  if($anzS -eq -1) { throw 'Schichtenangabe anzS fehlt' }
  if($anzS -lt  2) { throw 'Zu wenige Schichten' }
  if($anzS -eq  2) { $anzHN = 0 }
  else {
    if($anzHN -lt 1) { throw 'Zu wenige versteckte Neuronen' }
  }
  if($anzEN -lt 1) { throw 'Zu wenige Eingangsneuronen' }
  if($anzAN -lt 1) { throw 'Zu wenige Ausgangsneuronen' }

  return ($anzS, $anzEN, $anzHN, $anzAN)
} # holeNetzKonfig

# ------------------------------------------------------------------
function erzeugeGewichte {
  # Anfangsgewichte der Gewichtsmatrizen aus dem Bereich
  # +- 1/sqrt(Anzahl der Eingänge (Spalten)) ohne Null
  # Erzeugt werden die Dateien gew0.mat.start, ...
  # Parameter: Dateipfad, Anzahl Zeilen, Anzahl Spalten
  # ---------------------------------------------------
  param( [string]$p, [int]$z, [int]$s )

  $anz = $z*$s
  $grenze = 1/[math]::sqrt($s)
  $dVals = [Double[]]::new($anz)
  $i  = -1
  while($True) {
    $i++
    if($i -ge $anz) { break }
    $dVals[$i] = Get-Random -Min -$grenze -Max $grenze
    if($dVals[$i] -eq 0) { $i--; continue }
  }
  $bytes = foreach($val in $dVals) {
    [System.BitConverter]::GetBytes([double]$val)
  }
  [System.IO.File]::WriteAllBytes($p, $bytes)
} # erzeugeGewichte
```