NamensrÃ¤ume in Python (Scopes) (C) 2017-2020 T.Birnthaler OSTC GmbH
==========

Folgende NamensrÃ¤ume (Scopes) gibt es in Python:

Scope	Bedeutung
Global	Im gestarteten Skript (Modul "__main__") auÃM-^_erhalb von Funktion, Klasse, Objekt ohne Qualifizierung OBJEKT.NAME (nicht eingerÃ¼ckt)
Funktionslokal	Innerhalb FUNKTION per NAME = WERT oder Funktions-Parameter (Name + Wert gehen beim Verlassen der Funktion verloren)
Klassenlokal	Innerhalb KLASSE auÃM-^_erhalb METHODE per NAME = WERT (Zugriff per KLASSE.NAME)
Objektlokal	Innerhalb von METHODE per self.NAME = WERT
Modullokal	AuÃM-^_erhalb von MODUL per MODUL.NAME = WERT (innerhalb von MODUL analog globale Variable)
Objektattribut	Per OBJEKT.NAME

Folgende LEGB-Regel gilt bezÃ¼gliche der Reihenfolge der durchsuchten NamensrÃ¤ume bei einem Name Lookup von oben nach unten. Beim ersten Treffer ist die Suche zu Ende:

L)ocal	Function Scope / Lambda
E)nclosing	Nested Functions
G)lobal	Module Scope
B)uilt-in	Main Program

Es gibt KEINE Block-lokalen Namen in Python (KEINE Block-Scopes, BlÃ¶cke werden durch EinrÃ¼cken nach einem ":" gebildet):

```
  var = 0                   # var ist globale Variable
  for i in range(1,10+1):   # i ist globale Variable
      var += i              # var, i sind globale Variable
  print(var)                # var, i sind globale Variable
  print(i)                  # var, i sind globale Variable
```

Auf eine globale Variable kann in Funktionen LESEND zugegriffen werden, falls keine gleichnamige lokale Variable durch Zuweisung gebildet wird:

```
  gvar = "global"   # gvar ist globale Variable
  def f(...):       # Funktion f
      print(gvar)   # Globale Variable lesen
```

Per SchlÃ¼sselwort "global" kann eine Funktion auf einen globalen Namen des Hauptprogramms lesend + schreibend zugreifen:

```
  gvar = "global"       # gvar ist globale Variable
  def f(...):           # Funktion f
      global gvar       # Globale Variable lesen + schreiben erlauben
      print(gvar)       # Globale Variable lesen
      gvar = "lokal"    # Globale Variable schreiben
      print(gvar)       # Globale Variable lesen
```

Per SchlÃ¼sselwort "nonlocal" kann eine Funktion auf einen Namen der direkt umgebenden "Wrapper"-Funktion zugreifen (lesen + schreiben):

```
  def f(...):               # Funktion f definieren
      lvar = "lokal"        # Lokale Variable erzeugen
      def g(...):           # Eingeschachtelte (lokale) Funktion g definieren
          nonlocal lvar     # Nichtlokale Variable au0erhalb
          print(lvar)       # Nichtlokale Variable lesen
          lvar = "lokal"    # Nichtlokale Variable schreiben
          print(lvar)       # Nichtlokale Variable lesen
```

Das Modul "__main__" bietet einen 2. Zugriffsweg auf globale Namen des Hauptmoduls (Funktionen, Exceptions, Konstanten, ...):

```
  gvar = "hallo"             # gvar ist globale Variable
  print(gvar)                # Globale Variable lesen
  import __main__            # Namensraum von Hauptmodul
  print(gvar)                # Globale Variable lesen
  print(__main__.gvar)       # Globale Variable lesen
  def f(...):                # Funktion f definieren
      gvar = "lokal"         # Lokale Variable erzeugen
      print(gvar)            # Lokale Variable lesen
      print(__main__.gvar)   # Globale Variable lesen
```

```
      __main__.gvar          # Globale Variable schreiben
  f()                        # Funktion f aufrufen
  print(gvar)                # Globale Variable lesen
  print(__main__.gvar)       # Globale Variable lesen

Das Modul "builtins" bietet einen 2. Zugriffsweg auf den vordefinierten
globalen "Built-in" Namensraum (Funktionen, Exceptions, Konstanten, ...):

  map(...)             # Eingebaute Funktion aufrufen
  map = "hallo"        # Mit eigener Variable Ã¼berdefiniert
  print(map)           # (eingebaute Funktion nicht mehr erreichbar)
  import builtins      # "Built-in" Namensraum importieren
  builtins.map(...)    # Eingebaute Funktion aufrufen
  print(map)           # Eigene Variable lesen
```