Python Objekt Lebensdauer (Lifetime) (C) 2017-2022 T.Birnthaler OSTC GmbH
==========

--> python-scope.txt Scope/Sichtbarkeitsbereich
--> python-namespace.txt Namespace/Namensraum
--> python-lifetime.txt Lebensdauer/Lifetime/Existenz/Gültigkeitsbereich

Der Begriff "Lebensdauer/Lifetime/Existenz/Gültigkeitsbereich" (wie lange ein Python-Objekt = Wert/Objekt existiert und benutzbar ist) hat mit den Begriffen "Scope/Sichtbarkeitsbereich" und "Namespace/Namensraum" nichts zu tun!

In Python ist ein WERT/OBJEKT existent und benutzbar ("am Leben"), solange MINDESTENS EINE REFERENZ darauf zeigt und diese REFERENZ von Python aus direkt per NAME oder indirekt über andere Wert/Objekte erreichbar ist.

Sobald die LETZTE REFERENZ auf ein Wert/Objekt verschwindet, ist es nicht mehr erreichbar und sein Speicherplatz kann von Python für andere Zwecke verwendet werden. Dazu zählt Python die ANZAHL der Referenzen auf ein Wert/Objekt ständig in einem "Reference Counter" mit, der Teil des Wert/Objektes ist (abfragbar per sys.getrefcount(OBJ)).

Jede weitere Referenz erhöht den Referenz-Zähler um 1, jede aufgelöste Referenz verringert den Referenz-Zähler um 1. Erreicht der Referenzzähler den Wert 0, ist sichergestellt, dass ein Wert/Objekt nicht mehr erreichbar ("referenzierbar") und damit benutzbar ist und sein Speicherplatz wieder für andere Aufgaben zur Verfügung steht. --> TODO: GIL (Global Interpreter Lock).

Sich gegenseitig ZYKLISCH referenzierende aber insgesamt nicht mehr erreichbare Wert/Objekte werden von einer periodisch stattfindenden GARBAGE COLLECTION aufgeräumt. --> TODO: Effekt!

```
  text = "hallo welt"      # 1. Referenz auf str-Objekt "hallo welt"
  ref1 = text              # 2. Referenz auf str-Objekt "hallo welt"
  ref2 = [ 1, 2, text ]    # 3. Referenz auf str-Objekt "hallo welt"

  assert id(text) == id(ref1) == id(ref2[2])   # OK

  ref2[2] = 3              # 3. Referenz auf str-Objekt zerstört
  ref1 = 123               # 2. Referenz auf str-Objekt zerstört
  del text                 # 1. Referenz auf str-Objekt zerstört
```

Am Programmende löscht der Python-Interpreter immer alle Referenzen und daher werden auch alle Wert/Objekte aufgeräumt (d.h. ihr Speicherplatz wird freigegeben).

ACHTUNG: Dieses Verhalten gilt im Rahmen einer IDE (Integrated Development Environment) nicht, da der Python Interpreter am Programmende von der IDE nicht automatisch beendet wird. Üblicherweise wird er erst direkt vor dem nächsten Programmlauf beendet.

Datenübergabe (d.h. Übergabe von Werten/Objekten) an Funktionen und Datenrückgabe aus Funktionen erfolgt in Python grundsätzlich in Form von REFERENZEN (CALL BY/RETURN BY REFERENCE).

```
  def f(a, b, c):          # Referenzen auf Objekte in a, b, c übergeben;
      t = (a + b, b + c)   # Neues Tupel-Objekt mit Name t erzeugt
      return t             # Referenz auf Tupel-Objekt zurückgeben

  erg = f(1, 2, 3)         # Referenzen auf Objekte 1, 2 und 3 übergeben;
                           # erg enthält zurückgegebene Referenz auf
                           # in Funktion erzeugtes Tupel-Objekt zugewiesen
```

Parameter-Variablen zeigen also auf Objekte, die von außen stammen und stellen eine weitere Referenz auf diese da.

In einer Funktion NEU ERZEUGTE OBJEKTE können problemlos als return-Wert zurückgegeben werden (auch wenn der lokale Variablenname außerhalb der Funktion nicht mehr existiert), da bereits eine einzige Referenz auf sie außerhalb der Funktion dafür sorgt, dass sie nicht freigegeben werden.