07b_Miniuebungen_Loesungen

0.1 Mini-Aufgaben zur Überprüfung des Verständnis: Funktionsargumente

Erklären Sie in den folgenden Codeschnipseln die erwarteten Ausgaben, ohne sie auszuführen. Wenn Ihre Erwartung nicht zur Realität passt, dann untersuchen Sie das Verhalten im Detail, bspw. mit passenden id()-Ausgaben.

[4]:
```
var1 = 5
var2 = ["a"]

def test():
    var1 = 50
    var2[0] = "b"

print(var1, var2)
test()
print(var1, var2)
```

```
5 ['a']
5 ['b']
```

Hier gibt es keine Funktionsargumente, wir müssen also nur zwischen lokalen und globalen Variablen unterscheiden. Beide Variablen sind global. var1 is immutable, also erzeugt innerhalb der Funktion die Zuweisung eine neue lokale Variable. Ein Lesezugriff wäre nebenbei kein Problem. var2 zeigt auf ein mutable Objekt, deshalb wird die Änderung der Liste in der Funktion übernommen, eben gerade weil die Variable global ist und nicht Teil der Argumentliste. Listenmodifikationen durch [index] ändern das Originalobjekt qua Definition von mutable Modifikationen.

Zum Vergleich, wir sehen wie erwartet keinen Unterschied:

[1]:
```
var1 = 5
var2 = ["a"]

def test(var1,var2):
    var1 = 50
    var2[0] = "b"

print(var1, var2)
test(var1,var2)
print(var1, var2)
```

```
5 ['a']
5 ['b']
```

Erklären Sie das Verhalten auch hier:

[3]:
```
def foo(x, y):
    a = 42
    x,y = y,x
    b = 33
    b = 17
    c = 100
    print(a, b, x, y)

a,b,x,y = 1,15,3,4
print(a, b, x, y)
foo(17,4)
print(a, b, x, y)
```

```
1 15 3 4
42 17 4 17
1 15 3 4
```

Auch hier geht es wieder nur um lokale und globale Variablen. Keine der Änderungen innerhalb der Funktion ist außerhalb der Funktion sichtbar.

0.1.1 Berechnung der Fläche eines beliebigen Dreiecks

Ein beliebiges Dreieck kann durch Angabe der drei Eckpunkte $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ angegeben werden. Die Fläche des Dreiecks kann dann über die Formel

$$A = \tfrac{1}{2}|x_2y_3 - x_3y_2 - x_1y_3 + x_3y_1 + x_1y_2 - x_2y_1|$$

berechnet werden. Schreiben Sie eine Funktion area(vertices) welche die Fläche eines Dreiecks berechnet, das durch die verschachtelte Liste vertices definiert wird. Validieren Sie Ihre Implementierung anhand verschiedener Beispiele wie dem Dreieck, das durch die Eckpunkte $(0,0)$, $(1,0)$ und $(0,2)$ definiert ist.

Stellen Sie sicher, dass die verschachtelte Liste vertices nie kopiert wird.

[5]:
```
def area(v):
    # als Liste ist das Funktionsargument mutable, wird also nicht kopiert,
    # insbesondere, weil hier nur Lesezugriffe passieren
    return 0.5 * ( v[1][0]*v[2][1] - v[2][0]*v[1][1] - v[0][0]*v[2][1]
                 + v[2][0]*v[0][1] + v[0][0]*v[1][1] - v[1][0]*v[0][1] )

vertices = [(0,0), (1,0), (0,2)]
A = area(vertices) # so soll der Funktionsaufruf funktionieren
print(A)
```

```
1.0
```

0.2 Impressum

0.2.1 Programmierkurs Python, Dominik Göddeke https://www.ians.uni-stuttgart.de, Universität Stuttgart

Version vom April 2023

Veröffentlicht auf https://zoerr.de, (alle Rechte am Logo vorbehalten)

Gefördert durch die Stiftung Innovation in der Hochschullehre. (alle Rechte am Logo vorbehalten)

Gefördert mit Mitteln der Deutschen Forschungsgemeinschaft (EXC 2075 - 390740016) im Rahmen der Exzellenzstrategie.

[ ]: