Clevere Algorithmen programmieren: Berechnung mit Listen

08.01.2020

Code-Beispiele mit Listen:

```
a = [1, 3, 5, 1]          eine Liste definieren
a.append(8)               ein Element am ende der Liste hinzufügen
for x in a:               alle Element der Liste ausdrucken
  print x
print a[1]                das zweite Element der Liste ausdrucken (man zählt von null!)
print len(a)              die Länge der Liste (die Anzahl der Elemente) ausdrucken

b = [2*x for x in range(10)]      eine Liste mit einer Regel definieren
print b                           die ganze Liste ausdrucken
c = [[1], [2,2], [3,3,3]]         eine Liste von Listen ist auch möglich
print len(c)                      ☆ zur Aufgabe 1
d = []
for i in range(4):
  d.append([0, 0, 0, 0])
print d                           ☆ zur Aufgabe 1
```

1) Was drucken die Befehle mit ☆ im Code oben aus? Versuche die Antwort zu finden, ohne den Code im Computer auszuführen. Überprüfe es danach mit Computer.

2) Erstelle eine Liste aller Quadraten von 1 bis 10000. (`[1, 4, 9, 16, …, 10000]`)

3) Schreibe eine Funktion `quadratListe(N)`, die eine Liste mit N Listen, jede mit N Nulls, zurückgibt. Beispiel: `quadratListe(3) = [[0,0,0], [0,0,0], [0,0,0]]`.

4) Schreibe eine Funktion `dreieckListe(N)`, die eine Liste mit N Listen zurückgibt: die erste Liste mit einer 1, die nächste mit zwei 2s, dann drei 3s, …, bis zur letzten Liste mit N mal N. (`dreieckListe(4) = [[1], [2,2], [3,3,3], [4,4,4,4]]`)

5) Schreibe eine Funktion `fibonacci(N)`, die eine Liste mit den ersten N Zahlen der Fibonacci-Folge zurückgibt. Die Fibonacci-Folge ist eine Folge von Zahlen die mit zwei 1s beginnt, und jede nächste Zahl ist die Summe der zwei vorherigen. Also, die nächste ist 1 + 1 = 2, dann 1 + 2 = 3, dann 2 + 3 = 5, usw. (`fibonacci(7) = [1, 1, 2, 3, 5, 8, 13]`)

6) Schreibe eine Funktion `pascal(N)`, die die ersten N Reihen des Pascalschen Dreiecks zurückgibt. Das Pascalsche Dreieck ist ein unendliches dreieck von Zahlen, in dem jede Zahl die Summe der zwei darüberstehenden Zahlen ist:

```
          1
        1 1
      1 2 1
    1 3 3 1
  1 4 6 4 1
1 5 10 10 5 1
    . . .
```

Beispiel: `pascal(5) = [[1], [1,1], [1,2,1], [1,3,3,1], [1,4,6,4,1]]`