ETH zürich
Departement Informatik

Programmieren und Problemlösen

Dr. Dennis Komm
https://lec.inf.ethz.ch/ppl/
3. Mai 2021

Übungsblatt 4
Lösungsvorschläge

Aufgabe 1

Lesen Sie das folgende Programm und vollziehen Sie es nach. Das Programm fragt nach der nächsten Aktion, die auf einem Stack ausgeführt werden soll. Anschliessend wird die entsprechende Funktion aufgerufen. Falls dabei `add_to_stack(stack)` ausgewählt wird, muss zusätzlich eine Zahl eingeben werden. Danach wiederholt sich der Prozess.

```
1  def add_to_stack(stack):
2      i = int(input("Welche Zahl soll hinzugefuegt werden? "))
3      stack.append(i)
4
5  def print_stack(stack):
6      print("Stack:")
7      for i in range(len(stack)-1, -1, -1):
8          print(" ", stack[i])
9
10 def pop_from_stack(stack):
11     i = stack.pop()
12     print("Element", i , "vom Stack entfernt.")
13
14 def stack_manipulation():
15     stack = []
16     while True:
17         c = input("Add, Pop, Print oder Stop? ")
18         if c == "Add":
19             add_to_stack(stack)
20         elif c == "Pop":
21             pop_from_stack(stack)
22         elif c == "Print":
23             print_stack(stack)
24         elif c == "Stop":
25             break
26         else:
27             print("Eingabe nicht verstanden.")
28
29 stack_manipulation()
```

Sie erhalten im Folgenden alle `Add`-Eingaben, die eine Benutzerin oder ein Benutzer ausgeführt hat und welche Werte der Stack danach enthält. Finden Sie heraus, wann zwischen den `Add`-Eingaben jeweils `Pop` eingegeben wurde.

```
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 5
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 7
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 21
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 8
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 7
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 8
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 7
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 15
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 48
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 45
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 48
Add, Pop, Print oder Stop? Print
Stack:
   48
   45
   15
   7
   8
   8
   21
   5
Add, Pop, Print oder Stop? Stop
```

Lösung

```
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 5
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 7
Add, Pop, Print oder Stop? Pop
Element 7 vom Stack entfernt.
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 21
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 8
```

```
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 7
Add, Pop, Print oder Stop? Pop
Element vom 7 Stack entfernt.
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 8
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 7
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 15
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 48
Add, Pop, Print oder Stop? Pop
Element 48 vom Stack entfernt.
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 45
Add, Pop, Print oder Stop? Add
Welche Zahl soll hinzugefuegt werden? 48
Add, Pop, Print oder Stop? Print
Stack:
   48
   45
   15
   7
   8
   8
   21
   5
Add, Pop, Print oder Stop? Stop
```

Aufgabe 2

Betrachten Sie die folgende, leicht angepasste Version von Mergesort und bestimmen Sie die Ausgabe des Programms.

```python
def merge(left, right):
    result = []
    while len(left) > 0 and len(right) > 0:
        if left[0] > right[0]:
            result.append(right.pop(0))
        else:
            result.append(left.pop(0))
    return result + left + right

def mergesort(data):
    while len(data) > 1:
        left = data.pop(0)
        right = data.pop(0)
        data.append(merge(left, right))
```

```
        print(data)
    return data

data = [29, 17, 3, 4, 10, 2, 100, 47, 16, 5]
single_list = []
for d in data:
    single_list.append([d])

print(single_list)
mergesort(single_list)
```

Lösung

Die folgenden zweidimensionalen Listen werden ausgegeben.

```
[[29], [17], [3], [4], [10], [2], [100], [47], [16], [5]]
[[3], [4], [10], [2], [100], [47], [16], [5], [17, 29]]
[[10], [2], [100], [47], [16], [5], [17, 29], [3, 4]]
[[100], [47], [16], [5], [17, 29], [3, 4], [2, 10]]
[[16], [5], [17, 29], [3, 4], [2, 10], [47, 100]]
[[17, 29], [3, 4], [2, 10], [47, 100], [5, 16]]
[[2, 10], [47, 100], [5, 16], [3, 4, 17, 29]]
[[5, 16], [3, 4, 17, 29], [2, 10, 47, 100]]
[[2, 10, 47, 100], [3, 4, 5, 16, 17, 29]]
[[2, 3, 4, 5, 10, 16, 17, 29, 47, 100]]
```

Aufgabe 3

Kopieren Sie die untenstehenden Funktionen in Code-Expert und führen Sie sie aus. Fügen Sie an jeweils einer der angegeben Stellen ein `break`- oder ein `return`-Statement ein. Formulieren Sie, was `break` und `return` machen und versuchen Sie, die unterschiedlichen Ausgaben abzuleiten.

```
A = [ [ 1,  2,  3,  4],
      [ 5,  6,  7,  8],
      [ 9, 10, 11, 12],
      [13, 14, 15, 16] ]

def print_matrix(A):
    # break
    for i in range(0,4):
        # break
        for j in range(0,4):
            print(A[i][j])
            # break
        # break
    # break

def print_matrix_2(A):
```

```
17      # return
18      for i in range(0,4):
19          # return
20          for j in range(0,4):
21              print(A[i][j])
22              # return
23          # return
24      # return
25
26  print_matrix(A)
27  print_matrix_2(A)
```

Lösung

`break` beendet die Ausführung einer Schleife, aber nicht die aktuell ausgeführte Funktion. Es kann sich dabei um eine `while`- oder eine `for`-Schleife handeln.

`return` beendet wiederum die Funktion. Falls das Statement innerhalb einer Schleife befindet, so wird diese ebenfalls beendet.

Die beiden Schlüsselwörter haben also, je nach Ausführungszeitpunkt (Zeile) einen anderen Effekt, was die unterschiedlichen Ausgaben erklärt.

Aufgabe 4

Vollziehen Sie nach, was das folgende Programm bewirkt, welches eine List-Comprehension verwendet, um eine Liste von sogenannten „Tupeln“ zu erstellen. Fügen Sie Bedingungen hinzu, so dass in der Liste nur gerade Zahlen verwendet werden.

```
1  data = [ (a, b) for a in range(0, 10) for b in range(0, 10) ]
2  print(data)
```

Lösung

```
1  data = [ (a, b) for a in range(0, 10) for b in range(0, 10) \
2           if b % 2 == 0 and a % 2 == 0 ]
3  print(data)
```

Das Programm kreiert eine Liste mit Tupeln, die alle Kombinationen der Zahlen 0 bis 9 enthalten. Mit der zusätzlichen Bedingung werden nur noch gerade Zahlen verwendet.

Hinweis: „\“ erlaubt Ihnen, etwas in 2 Zeilen zu schreiben. Dies wird hier gemacht, um die Lösung leserbarer zu gestalten.

Aufgabe 5

Initialisieren Sie eine (10×10)-Matrix, welche mit Nullen gefüllt ist. Verwenden Sie hierzu zwei List-Comprehensions.

```
1  matrix =   # Fuegen Sie Ihren Code hier ein
2  for m in matrix:
3      print(m)
```

Lösung

```
matrix = [ [ 0 for y in range(0, 10) ] for x in range(0, 10) ]
for m in matrix:
    print(m)
```

`[ ... for x in range(0, 10) ]` beschreibt, was für jedes `x` in `range(0, 10)` eingesetzt werden soll. `[ 0 for y in range(0, 10) ]` ist eine Liste, die für jedes `y` in `range(0, 10)` eine 0 enthält.

Aufgabe 6

Betrachten Sie das folgende Programm zunächst und überlegen Sie sich, was die Ausgabe sein könnte. Führen Sie es danach aus und erklären Sie die Ausgabe.

```
A = [ [ 1, 2, 3, 4],
      [ 5, 6, 7, 8],
      [ 9, 10, 11, 12],
      [13, 14, 15, 16] ]

print(A[0][-1])
print(A[0:2][-1])
```

Lösung

Um das Programm besser zu verstehen, können Sie Folgendes ausführen:

```
A = [ [ 1, 2, 3, 4],
      [ 5, 6, 7, 8],
      [ 9, 10, 11, 12],
      [13, 14, 15, 16] ]

print(A[0][-1])
print(A[0:2][-1])
B = A[0:2]
print(B)
```

Die Ausgabe ist dann:

```
4
[5, 6, 7, 8]
[ [1, 2, 3, 4], [5, 6, 7, 8] ]
```

`A[0:2]` gibt eine Liste von Listen zurück. In `B` sind das nullte und das erste Element von `A` enthalten, also die beiden ersten Listen von `A`. `A[0:2][-1]` ist äquivalent zu `B[-1]`, also wird das letzte Element von `B` ausgegeben. Da `B` wiederum eine Liste von Listen ist, ist auch dieses letzte Element eine Liste und zwar `[5, 6, 7, 8]`.