Aufgaben zu Methoden in Python

Aufgabe 1

Erkläre dieses gesamte Programm mit „idiotensicheren“ englischen Kommentaren:

addcomments.py

```
def sum(a, b):
    return (a + b)

a = int(input('Enter 1st number: '))
b = int(input('Enter 2nd number: '))

print(f'Sum of {a} and {b} is {sum(a, b)}')
```

Aufgabe 2

Schreibe eine Methode, die eine Prozentangabe (Parameter 1) und einen Basiswert (Parameter 2) als Benutzer:inneneingabe entgegennimmt und daraus den Prozentanteil berechnet. Die Ausgabe soll in der Methode selbst erfolgen.

Beispielablauf:

```
Geben Sie den Basiswert ein: 100
Geben Sie den Prozentanteil ein: 30
30% von 100 sind 30.
```

Aufgabe 3

Modifiziere deine Methode aus Aufgabe 2 so, dass die Ausgabe NICHT innerhalb der Methode, sondern im Hauptprogramm erfolgt (return prozentanteil).

Aufgabe 4

Man kann auch Arrays übergeben (und zurückgeben), z.B. dieses hier.

array_bsp.py

```
noten = [1,2,3,4,5,6]
```

Schreibe eine Methode, die ermittelt, ob eine eingebene Zahl im Array „noten“ enthalten ist (z.B.

Rückgabewert return True). Achte darauf, dass die Zählung der Elemente eines Arrays bei Null beginnt.

Die Länge eines Arrays (Anzahl der Elemente), kannst du mit len() bestimmen.

array_len.py

```
noten = [1,2,3,4,5,6]
print(len(noten))
```

Aufgabe 5

Schreibe eine Methode FillArray(laenge), die ein array_bsp Array der Länge x (Parameter) mit zufälligen Ganzzahlen füllt und dieses Array zurückgibt. Gibt dann die Elemente dieses Arrays wiederum mit einer weiteren Methode PrintArray(feld) aus, die du INNERHALB der Methode FillArray(laenge) aufrufst.

Hinweis zum Anhängen von Elementen an ein Array

array_append.py

```
feld = []
element0 = 1
feld.append(element0)
```

Aufgabe 6

Beschreibe, was dieser Code macht.

CallByReference.py

```
# Call by value oder call by reference?

def SetField(field):
    field[0] = 5

test = [1,2,3,4,5,6]

SetField(test)

# Call by reference!
print(test[0])
```

Es ergibt sich ein Widerspruch zu den Aussage zur Gültigkeit von Variablen. Welcher Widerspruch ist das? Suche im Netz nach „call by value“ und „call by reference“, um dieses Verhalten zu erklären.

Lösungen

Klicke hier für Beispiellösungen

Es handelt sich um Musterlösungen. Beim Programmieren gibt es stets unterschiedliche Ansätze. Wenn dein Programm auch mit einem anderen Ansatz gut funktioniert, gilt die Aufgabe auch als gelöst.

Aufgabe 1

commented.py

```
# first defining a method requiring two parameters a,b to sum up
def sum(a, b):
   # returning result of sum from parameters
   # that might be an integer in most cases
   return (a + b)

# requesting two integers - may be type "float" could be more suitable
for higher precision
a = int(input('Enter 1st number: '))
b = int(input('Enter 2nd number: '))

# generating a formatted output.
# as shown, method call represents an integer value
print(f'Sum of {a} and {b} is {sum(a, b)}')
```

Aufgabe 2

percentage01.py

```
def calc_percentage(base_value, percentage_value):
   result = percentage_value/base_value*100
   print(f'{percentage_value}% von {base_value} sind {result}')

base = float(input('Geben Sie den Basiswert ein: '))
percentage = float(input('Geben Sie den Prozentanteil ein: '))

calc_percentage(base, percentage)
```

Aufgabe 3

percentage02.py

```
def calc_percentage(base_value, percentage_value):
   result = percentage_value/base_value*100
   return result

base = float(input('Geben Sie den Basiswert ein: '))
percentage = float(input('Geben Sie den Prozentanteil ein: '))

result = calc_percentage(base, percentage)

print(f'{percentage}% von {base} sind {result}')
```

Aufgabe 4

array01.py

```
# methods
def search_in_array(array, key):

   # hier etwas Magie ...
   # wir laufen von i=0 bis zur Länge des Arrays vermindert um 1 -
range(len(array)-1)
   # Sonst bekommen wir einen Indexfehler, weil Elemente von Arrays
beginnend bei 0 gezählt werden

   #for i in range(len(array)-1):
   #   if array[i] == key:
   #      # return wirkt hier wie "break"
   #      return True

   # der for-Teil lässt aber (noch) einfacher schreiben
   for i in array:
      if i == key:
         # return wirkt hier wie "break"
         return True

   # Wenn wir durchgelaufen sind und nichts gefunden haben
   return False

# values
noten = [1,2,3,4,5,6]
key = int(input('Nach welcher Zahl soll gesucht werden? '))

# main
if search_in_array(noten, key):
   print(f'Zahl {key} befindet sich im Array.')
else:
   print(f'Zahl {key} befindet sich nicht im Array.')
```

Aufgabe 5

import random

array02.py

```
import random

# methods
def fillArray(size):
   i = 0
   array_random = []
   while i < size:
      array_random.append(random.randint(1,100))
      i+=1
   printArray(array_random)

def printArray(array):
   for i in array:
      print(i)

# main
fillArray(8)
```

Aufgabe 6

Innerhalb der Methode wird der Inhalt eines Elements geändert. Diese Änderung wirkt sich **auch** im Hauptprogramm aus. Eigentlich sollten Variablen in Methoden streng von Variablen im Hauptprogramm getrennt sein und mit Rückgabewerten gearbeitet werden.

Arrays sind in Python oder aber auch in Java Referenzdatentypen. Sie enthalten eine Adresse zu einem Speicherbereich, in dem die einzelnen Elemente liegen. Diese Adresse wird für die Methode kopiert, zeigt aber auf die gleichen Elemente. Man spricht auch von einer **Referenz** auf die Elemente.

Wenn man das sauber trennen möchte, müsste man das gesamte Array vor dem Aufruf der Methode kopieren, um dann dort Änderungen zu machen. Da das Ressourcen erfordert und auch der Umgang mit Arrays dadurch komplexer wird, macht man das nicht, sondern arbeitet mit Referenzen (Call-by-Reference).