Aufgabe 1

Was ist ein Datentyp?

Aufgabe 2

Zähle drei einfache (primitive) Datentypen der Programmiersprache Python auf.

Aufgabe 3

Welche Laufzeitkomplexität haben die folgenden Operationen für den Python-Datentyp `list`? Gehe davon aus, dass die gegebene Liste `L` die Länge n hat. Der Index `i` ist jeweils ein gültiger Index ($0 \leq i < n$).

(a) `L[i]`

(b) `L.pop(i)`

(c) `L.pop()`

(d) `L.append(77)`

Aufgabe 4

Welche Laufzeitkomplexität haben die folgenden Operationen für den Python-Datentyp `dict`? Gehe davon aus, dass das gegebene Dictionary `D` die Länge n hat. Die Variable `k` bezeichnet jeweils einen gültigen Schlüssel.

(a) `D[k]`

(b) `D.del(k)`

(c) `for k in D:`

Aufgabe 5

Welche Ausgaben machen die folgenden Python-Anweisungen, wenn die Mengen `A` und `B` wie folgt definiert sind.

`A = {1, 2, 3, 5}`

`B = {1, 3, 5, 7}`

Die folgenden Aufgaben sind jeweils unabhängig voneinander.

(a) `print(A.union(B))`

(b) `print(A.intersection(B))`

(c) `print(B.difference(A))`

(d) `print(A.issubset(B))`

Aufgabe 6

Nenne vier Anwendungsgebiete für Stacks in der Informatik.

Aufgabe 7

Es sei `s` eine Instanz der Klasse `Stack`. Welches ist der Unterschied zwischen den Ausdrücken `s.peek()` und `s.pop()`?

Aufgabe 8

Eine Python-Implementation des Datentyps `Stack` hat folgenden Konstruktor

```
class Stack:

    def __init__():
        self.items = []
```

Implementiere die folgenden Methoden

(a) `pop()`

(b) `push(item)`

(c) `isEmpty()`

für Objekte des Datentyps `Stack`.

Aufgabe 9

Stelle die folgenden Infix-Ausdrücke jeweils als gleichwertige, klammerlose Postfix-Terme dar. *Hinweise:* `a`, `b`, `c`, `d` stehen für Zahlen, `**` ist der Potenzoperator.

(a) `a * b - (c + d)`

(b) `c / b ** (d - a)`

Aufgabe 10

Gib die Ausgabe des folgenden Programmfragments als Liste an, wenn der Stack von links nach rechts wächst.

```
s = Stack()
s.push('m')
s.push('a')
s.push('z')
s.pop()
s.push('k')
s.push('c')
s.pop()

print(s)
```

Aufgabe 11

Beschreibe drei verschiedene Anwendungen für die Datenstruktur *Queue*.

Aufgabe 12

Notiere die Kurzform für die Art und Weise, wie Daten zu Queues hinzugefügt bzw. aus ihnen entfernt werden.

Aufgabe 13

Eine Python-Implementation des Datentyps Queue hat folgenden Konstruktor

```
class Queue:

    def __init__():
        self.items = []
```

Implementiere die folgenden Methoden für Objekte dieses Datentyps:

(a) `enqueue(item)`

(b) `size()`

Aufgabe 14

Gib die Ausgabe des folgenden Programmfragments als Liste an, wenn die Elemente von links in die der Queue zugrunde liegende Liste eingefügt werden.

```
from queue import Queue

q = Queue()
q.enqueue('m')
q.enqueue('z')
q.dequeue()
q.enqueue('a')
q.enqueue('f')
q.enqueue(q.dequeue())
q.enqueue('s')

print(q)
```

Hinweis: Da es für Python 3.x bereits ein Modul mit dem Namen `queue` gibt, muss ein Modul mit gleichem Namen im gleichen Verzeichnis wie das es verwendende Programm liegen, damit es keine Konflikte gibt. Besser wäre es jedoch, das eigene Modul anders zu nennen (z. B. `myqueue`).