Einführung in die Programmierung

Prof. Dr. Peter Thiemann
Marius Weidner
Simon Dorer, Moritz Steffl

Universität Freiburg
Institut für Informatik
Wintersemester 2026

Übungsblatt 2

Abgabe: Dienstag, 03.11.2026, 10:00 Uhr

Bitte beachten Sie:

- Falls Sie sich noch nicht für ein Tutorat angemeldet haben, melden Sie sich bitte **jetzt über HISinOne** für ein Tutorat an.
- Für die Bearbeitung und die Abgabe gelten weiterhin die Abgaberegeln.
- Bei Fragen oder Problemen können Sie sich gerne im Forum melden. Bitte posten Sie Fragen zu den Übungsblättern im jeweils passenden Unterforum, beispielsweise `exercise02` für dieses Blatt. So können Ihre Mitstudierenden die Fragen und Antworten zum jeweiligen Übungsblatt leichter finden und verfolgen.

Aufgabe 2.1 (Typen; 4 Punkte; Datei: `types.txt` oder `types.md`)

Entscheiden Sie für jede der folgenden Wertzuweisungen, ob sie ohne Fehler ausgeführt werden kann. Falls ja, bestimmen Sie den Typ von `res` nach der Zuweisung. Begründen Sie Ihre Antwort jeweils kurz.

(a) `res = float(str(2 // float("1" + "e" + "3"))) * int("""10""")`

(b) `res = int(float("3.0")) * "2" + "1" * 2`

(c) `res = str(int("1" * 4) + 3333) + int("4" * 4 * 4.0) * "1"`

(d) `res = int("3" * (3 ** 3)) // int(3.3) // 3 ** 3.3`

Hinweis zu Aufgabe 2.2, 2.3 und 2.4: `input`-Funktion

Wie in der Vorlesung gezeigt, können Sie mit der `input`-Funktion Benutzereingaben in Ihr Python-Programm einlesen. Im Folgenden wird die Benutzereingabe in türkiser Schrift dargestellt und die Ausgabe des Programms in schwarzer Schrift.

Ein Beispiel für die Verwendung der Funktion `input` ist das Folgende:

```
>>> num = input("Bitte geben Sie hier eine Zahl ein: ")
Bitte geben Sie hier eine Zahl ein: 42
>>> num
'42'
```

Die Funktion `input` gibt immer einen String zurück, auch wenn die Benutzerin eine Zahl eingibt. Sie können einen String in eine Zahl umwandeln, indem Sie die `int`

oder `float` Funktion verwenden:

```
>>> int('1337')
1337
>>> float('3.1415')
3.1415
```

Versucht man einen String, der keiner Zahl entspricht, zu konvertieren, wird die Ausführung des Programms durch eine Ausnahme zum Absturz gebracht:

```
>>> int('not a number')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'not a number'
```

In Ihrem Python-Skript dürfen Sie dies ignorieren. Sie können also annehmen, dass die Benutzerin korrekt formatierte Eingaben macht.

Aufgabe 2.2 (Leiter zeichnen; 2 Punkte; Datei: `ladder.py`)

Erstellen Sie ein Python-Skript `ladder.py`, das die Benutzerin dazu auffordert, die **Breite der Leiter** und die **Anzahl Sprossen** einzugeben.

Im Anschluss soll das Programm eine Leiter mit der angegebenen Breite und Anzahl an Sprossen ausgeben. Die Sprossen bestehen aus Bindestrichen (`-`) und werden an beiden Enden durch ein Pluszeichen (`+`) mit den seitlichen Stäben (`|`) verbunden.

Sie dürfen davon ausgehen, dass die Benutzerin stets eine ganze Zahl größer oder gleich 3 für die Breite und eine ganze Zahl größer oder gleich 1 für die Anzahl der Sprossen eingibt.

Beispielsweise sollen Aufrufe des Skripts mit den Eingaben `"3"` und `"1"` beziehungsweise `"9"` und `"3"` *exakt* folgende Ausgaben erzeugen:

```
Breite der Leiter: 3
Anzahl Sprossen: 1

| |
+-+
| |
```

```
Breite der Leiter: 9
Anzahl Sprossen: 3

|       |
+-------+
|       |
+-------+
|       |
+-------+
|       |
```

Aufgabe 2.3 (PQ-Formel; 2 Punkte; Datei: `pq.py`)

Erstellen Sie ein Python-Skript `pq.py`, das die pq-Formel für zwei beliebige Gleitkommazahlen p und q berechnet:

$$x_{1,2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}$$

Ihr Skript soll dabei die Benutzerin zunächst auffordern die Werte für p und q einzugeben. Anschließend soll es die beiden möglichen Lösungen x_1 und x_2 der pq-Formel berechnen und ausgeben.

Sie dürfen dabei annehmen, dass die Benutzerin stets Werte für p und q eingibt, die für nichtnegative Werte unter der Wurzel sorgen. Wenn es eine eindeutige Lösung gibt, soll diese trotzdem doppelt ausgegeben werden.

Verwenden Sie in Ihrem Programm Variablen, um mehrfach vorkommende Berechnungen zwischenzuspeichern, statt diese neu zu berechnen.

Ein Aufruf des Skripts mit den Eingaben `4.0` und `-5.0` soll *exakt* diese Ausgabe liefern:

```
p = 4.0
q = -5.0
x1 = 1.0
x2 = -5.0
```

Aufgabe 2.4 (Ausdrücke optimieren; 2 Punkte; Datei: `optimize.py`)

Betrachten Sie das folgende Python-Programm:

```
x = input()
n = float(x)
a = n * 2
b = a + 1337
c = b ** 2
d = c - n ** 3
e = d / 42
print(e)
```

Erstellen Sie ein Python-Skript `optimize.py`, das *exakt* das gleiche Verhalten wie das obige Programm hat, aber so wenige Variablenzuweisungen wie möglich verwendet!

Aufgabe 2.5 (Erfahrungen; Datei: `NOTES.md`)

Notieren Sie Ihre Erfahrungen mit diesem Übungsblatt (benötigter Zeitaufwand, Probleme, Bezug zur Vorlesung, Interessantes, etc.).

Editieren Sie hierzu die Datei `NOTES.md` im Abgabeordner dieses Übungsblattes auf unserer Webplattform. Halten Sie sich an das dort vorgegebene Format, da wir den Zeitbedarf mit einem Python-Skript automatisch statistisch auswerten. Die Zeitangabe `4.5 h` steht dabei für 4 Stunden 30 Minuten.

Der Build-Server überprüft ebenfalls, ob Sie das Format korrekt angegeben haben. Prüfen Sie, ob der Build-Server mit Ihrer Abgabe zufrieden ist, so wie es im Skript zum Abgabesystem gezeigt wurde.