Einführung in die Programmierung

Prof. Dr. Peter Thiemann
Marius Weidner
Simon Dorer, Timpe Hörig

Universität Freiburg
Institut für Informatik
Wintersemester 2025

Übungsblatt 2

Abgabe: Montag, 27.10.2025, 09:00 Uhr

Vorab beachten Sie bitte folgendes:

- Falls Sie sich noch nicht für ein Tutorat in HisInOne angemeldet haben, tun Sie dies bitte **umgehend** hier. Schreiben Sie aufgrund der verspäteten Anmeldung zudem eine E-Mail an Marius Weidner (weidner@cs.uni-freiburg.de) mit der **Übungsgruppe**, der Sie beigetreten sind, und Ihrem **RZ-Kürzel**.
- Beachten Sie, dass weiterhin die Abgaberegeln (hier) gelten.
- Sollten Sie Fragen oder Probleme haben, dann können Sie gerne im Chat oder in den Tutorien fragen. Wir sind immer noch ganz lieb und beißen auch nicht :)

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

Bestimmen Sie nach jeder der folgenden Wertzuweisungen an die Variable `res` den Typ von `res`. Geben Sie jeweils eine kurze Erläuterung, warum das so ist.

(a) `res = 3 + float(str(6 - float("." + "32")) + """4""")`

(b) `res = int(float("3.5")) * "2" + "1" * 2`

(c) `res = int(2 ** 4.5 // 3) * "X" + "Y" * 3`

(d) `res = int("3" * (3 ** 3)) // int(3.3) // 2 ** 2.0`

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 einer blauen 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: could not convert string to int: '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 (Rechteck zeichnen; 2 Punkte; Datei: `rectangle.py`)

Erstellen Sie ein Python-Skript `rectangle.py`, das die Benutzerin nacheinander dazu auffordert, die **Breite** und **Höhe** eines Rechtecks sowie jeweils ein einzelnes Zeichen für den **Rand** und die **Innenfläche** des Rechtecks einzugeben.

Im Anschluss soll das Programm ein Rechteck mit der angegebenen Breite und Höhe ausgeben, wobei die Ränder und die Innenfläche aus den jeweils eingegebenen Zeichen bestehen.

Sie dürfen davon ausgehen, dass die Benutzerin stets ganze Zahlen größer oder gleich 2 für die Breite und die Höhe sowie genau ein Zeichen für den Rand und die Innenfläche eingibt.

Beispielsweise soll ein Aufruf des Skripts mit den Eingaben `"12"`, `"4"`, `"$"` und `"-"` *exakt* folgende Ausgabe erzeugen:

```
Breite: 12
Hoehe: 4
Rand: $
Innenflaeche: -
$$$$$$$$$$$$
$----------$
$----------$
$$$$$$$$$$$$
```

Aufgabe 2.3 (Mitternachtsformel; 2 Punkte; Datei: `abc.py`)

In dieser Aufgabe schreiben Sie ein Python-Skript, das die Mitternachtsformel für drei beliebige Gleitkommazahlen a, b und c berechnet:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Ihr Skript soll dabei die Benutzerin zunächst auffordern die Werte für a, b und

c einzugeben. Anschließend soll es die beiden möglichen Lösungen x_1 und x_2 der Mitternachtsformel berechnen und ausgeben.

Sie dürfen dabei annehmen, dass die Benutzerin stets Werte für a, b und c eingibt, die für positive 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 `-2.0`, `-8.0` und `10.0` soll *exakt* diese Ausgabe liefern:

```
a = -2.0
b = -8.0
c = 10.0
x1 = -5.0
x2 = 1.0
```

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

Betrachten Sie das folgende Python-Programm:

```
x = input()
n = float(x)
a = n // 1.337
b = a ** 3
c = b * a
d = c - n
e = d * 10
print(e)
```

Schreiben 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 Video zur Lehrplattform gezeigt wurde.