Einführung in die Programmierung

Prof. Dr. Peter Thiemann
Marius Weidner, Hannes Saffrich
Lukas Kleinert, Timpe Hörig

Universität Freiburg
Institut für Informatik
Wintersemester 2023

Übungsblatt 2

Abgabe: Montag, 30.10.2023, 9:00 Uhr morgens

Falls sie noch nicht für ein Tutorat/Übungsgruppe in HisInOne angemeldet haben, tun Sie dies umgehend. Schreiben Sie aufgrund der verspäteten Anmeldung zudem eine Email[1] mit der Übungsgruppe, der Sie begetreten sind und ihrem RZ Accountname. Falls Sie diese Woche zum ersten Mal eine Abgabe machen, schreiben Sie ebenfalls eine Mail an Marius Weidner, damit Sie einer Tutorin zugewiesen werden. Sollten Sie Probleme haben, dann können Sie gerne im Chat oder in den Tutoraten fragen. Wir sind auch immer noch ganz lieb und beißen auch nicht :)

Dateiformate und sonstige Regeln für Ihre Abgabe

- Achten Sie darauf, dass Ihre Ausgaben exakt so aussehen wie in den vorgegebenen Beispielen! Achten Sie dabei z.B. auch auf Leerzeichen.
- Verwenden Sie nur Befehle und Programmiertechniken, die Inhalt der *bisherigen* Vorlesungen (bis zum Abgabetermin) und Übungsblättern waren.
- Nach dem Hochladen Ihrer Lösung führt der Buildserver style checks ihres Codes mit `Flake8` aus. Diese style checks müssen mit *success* beendet werden.
- Die einzelnen Aufgaben auf den Übungsblättern sind stets mit vollständigen Dateinamen annotiert. Bitte halten Sie sich an die Namen und die dazugehörigen Dateiformate. Beispielsweise soll Aufgabe 2.1 als Datei `arithmetik.txt` abgegeben werden.
- Die Dateiformate sind immer in *plaintext* (UTF-8) und haben die Endungen `.txt` bzw. `.md` für Text und `.py` für Python-Code. Das heißt es sind insbesondere keine PDFs, keine Word-Dokumente und auch keine Bildschirmfotos erlaubt!
- Eine simple Möglichkeit diese Kriterien zu erfüllen ist es einfach alle Dateien mit Visual Studio Code zu erstellen bzw. zu bearbeiten. Wenn Sie zum Beispiel die Endung `.txt` verwenden, dann erkennt Visual Studio Code, dass es sich nicht um ein Python-Skript handelt und verhält sich wie ein normaler Texteditor. Beim Bearbeiten von `.py` Dateien werden `Flake8`-Warnungen gelb markiert und mit einem Mouseover erklärt.
- Abgaben per Mail können nicht berücksichtigt werden. Geben Sie Ihre Aufgaben über unser git-System [2] ab.

[1] `mailto:weidner@cs.uni-freiburg.de`
[2] `https://git.laurel.informatik.uni-freiburg.de/`

Aufgabe 2.1 (Arithmetische Ausdrücke; 4 Punkte; Datei: `arithmetik.txt`)

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 = "X" * (1 + 2) ** 2 + str(2)
```

(b)
```
>>> res = int(float(str(2 + 52) + ".3")) + 5
```

(c)
```
>>> from math import log, e
>>> res = int(log(42, e)) * abs(e ** 1)
```

(d)
```
>>> res = (0x1b ^ 0b1110) // int(3.1415) + 2.0
```

Hinweise zu input

Blaue Schrift stellt eine Eingabe der Benutzerin dar. Beispiel:

```
>>> print(input("Bitte geben Sie etwas ein: "))
Bitte geben Sie etwas ein: Kekse!!!
Kekse!!!
```

Strings können wie folgt zu Ganzzahlen konvertiert werden:

```
>>> int('2')
2
```

Versucht man einen String, der keiner Ganzzahl 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. Das gleiche gilt z.B. auch für Gleitkommazahlen (float). Sie können also annehmen, dass die Benutzerin stets eine valide Eingabe tätigt.

Aufgabe 2.2 (Fläche drucken; 2 Punkte; Datei: `area.py`)

Schreiben Sie ein Python-Skript `area.py`, welches die Benutzerin dazu auffordert, eine Breite und eine Höhe (jeweils ganzzahlig) einzugeben. Dann soll die Benutzerin dazu aufgefordert werden, ein Zeichen [3] einzugeben. Anschließend soll dieses Zeichen der Breite und Höhe entsprechend oft ausgegeben werden, und es soll die Anzahl der Zeichen ausgegeben werden.

[3] Sie dürfen davon ausgehen, dass die Eingabe die Länge 1 hat

Ein Aufruf des Skripts mit der Eingabe `5`, `3` und `X` soll *exakt* so aussehen:

```
Breite: 5
Höhe: 3
Zeichen: X
XXXXX
XXXXX
XXXXX

Anzahl: 15
```

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

Schreiben Sie ein Python-Skript `abc.py`, welches die Benutzerin dazu auffordert, die Gleitkommazahlen `a`, `b` und `c` für die Mitternachtsformel

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

einzugeben, beide mögliche Lösungen x_1 und x_2 ausrechnet, und diese ausgibt. Verwenden Sie in Ihrem Programm Variablen um mehrfach vorkommende Berechnungen zwischenzuspeichern, statt diese neu zu berechnen.

Ein Aufruf des Skripts mit den Eingaben 1, -2 und -3 soll *exakt* so aussehen:

```
a = 1
b = -2
c = -3
x1 = 3.0
x2 = -1.0
```

Eingaben, bei denen der Wert unter der Wurzel negativ ist, dürfen Sie ignorieren. Wenn es eine eindeutige Lösung gibt, soll diese trotzdem doppelt ausgegeben werden.

Aufgabe 2.4 (Ausdrücke zusammenfassen; 2 Punkte; Datei: `shorter.py`)

Betrachten Sie das folgende Python-Programm:

```
i = input()
n = float(i)
a = n
b = 2 * n
c = b + 0.33
d = a / 2.72
e = d ** n
f = (3.14 * c)
result = f ** 0.5 * e
print(result)
```

Schreiben Sie ein Python-Skript `shorter.py`, das *exakt* das gleiche Verhalten wie das obige Programm hat. Jedoch soll Ihr Programm so wenige Zeilen wie möglich haben!

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 Webplatform. Halten Sie sich an das dort vorgegebene Format, da wir den Zeitbedarf mit einem Python-Skript automatisch statistisch auswerten. Die Zeitangabe `3.5 h` steht dabei für 3 Stunden 30 Minuten.

Der Buildserver überprüft ebenfalls, ob Sie das Format korrekt angegeben haben. Prüfen Sie, ob der Buildserver mit Ihrer Abgabe zufrieden ist, so wie es im Video zur Lehrplatform gezeigt wurde.