Informatik I: Einführung in die Programmierung

Prof. Dr. Peter Thiemann
Dr. Daniel Büscher, Hannes Saffrich
Wintersemester 2019

Universität Freiburg
Institut für Informatik

Übungsblatt 4 – Lösungen

Abgabe: Montag, 18.11.2019, 9:00 Uhr morgens

Achtung: Abgaben von nicht-ausführbaren Programmen werden ab diesem Übungsblatt nicht bewertet. Als *nicht-ausführbar* gelten Programme, die bei versuchter Ausführung einen `SyntaxError` verursachen, da sie keinen gültigen Python-Code enthalten.

Aufgabe 4.1 (Boolsche Ausdrücke; Datei: `bool.txt`; 4 Punkte)

Beschreiben Sie den Definitionsbereich (alle möglichen Belegungen für `x` bzw. `y`) für welche die folgenden Python-Funktionen `True` zurückgeben. Falls der Definitionsbereich leer ist, begründen Sie kurz (in 1-2 Sätzen) wieso dies so ist.

(a)
```
def foo(x: float) -> bool:
    return not x >= 35
```

(b)
```
def bar(x: int) -> bool:
    if x > 3 and x % 3 == 0:
        return False
```

(c)
```
def baz(x: int, y: int) -> bool:
    if y % x == 0:
        return x % y == 0
    else:
        return False
```

(d)
```
def boo(x: int, y: int) -> bool:
    if x > y:
        print(True)
    else:
        print(False)
```

Lösung:

(a) Alle Gleitkommazahlen `x` die kleiner `35` sind. In mathematischer Schreibweise: $\{x \in \texttt{float} \mid x < 35\}$.

(b) Leere Menge. Die Funktion gibt entweder `False` zurück oder implizit `None`, da kein zweites `return`-Statement vorhanden ist. In mathematischer Schreibweise: $\emptyset$.

(c) Alle ganzen Zahlen `x` und `y` mit `x != 0` und `abs(y) == abs(x)`. In mathematischer Schreibweise: $\{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid x \neq 0 \wedge y = |x|\}$.

(d) Leere Menge. Ausgaben sind keine Rückgabewerte, es wird also stehts `None` zurückgegeben. In mathematischer Schreibweise: $\emptyset$.

Aufgabe 4.2 (Schaltjahre; Datei: `leap.py`; 5 Punkte)

Schreiben Sie eine Funktion `leapyear(year: int) -> bool`, welche für ein gegebenes Jahr `year` berechnet, ob dieses Jahr ein Schaltjahr (im gregorianischer Kalendar) ist oder nicht. Ein Jahr ist ein Schaltjahr, wenn das Jahr ganzzahlig durch 4 teilbar ist, ohne dabei ganzzahlig durch 100 teilbar zu sein, es sei denn, das Jahr ist außerdem durch 400 teilbar.

Beispielsweise ist 1997 kein Schaltjahr, 1996 ist ein Schaltjahr. 1900 ist kein Schaltjahr, 2000 ist ein Schaltjahr.

Lösung:

```
def leapyear(year: int) -> bool:
    return year % 4 == 0 and (not year % 100 == 0 or year % 400 == 0)
```

Aufgabe 4.3 (Schulnoten; Datei: `grading.py`; Punkte: 5+2+2)

Im Folgenden geht es darum, einen Bewertungsschlüssel zur Verteilung von Schulnoten für eine Klausur mit 60 erreichbaren Punkten zu implementieren. Gültige Notenstufen sind $1.0, 1.5, \ldots, 5.5$ und 6.0. Idee hierbei ist, dass Schüler mit einer Punktzahl von 20% der erzielbaren Punkte (das sind 12 Punkte) oder weniger in jedem Fall die Note 6.0 erhalten, Schüler mit einer Punktzahl von mindestens 95% (also 57 Punkten) in jedem Fall eine 1.0 erzielen. Ansonsten wird folgendes Verfahren angewendet: für eine Punktzahl $12 < p < 57$ wird zunächst ein „exakter" Notenwert mit der Formel $-\frac{5}{45} \cdot (p - 57) + 1$ berechnet. Dieser wird dann zur nächstgelegenen Notenstufe gerundet.

(a) Implementieren Sie die oben beschriebene Benotungsvorschrift als Funktion `lineargrade(p: float) -> float`, die bei Eingabe einer Punktzahl `p` ≥ 0 die zugehörige Note zurückgibt. Testen Sie Ihre Funktion an geeigneten Beispielen, also z.B. mit:

```
>>> from math import isclose
>>> isclose(lineargrade(0), 6.0)
True
>>> isclose(lineargrade(36.5), 3.5)
True
>>> isclose(lineargrade(37), 3.0)
True
>>> isclose(lineargrade(60.0), 1.0)
True
```

Hinweis: Da es sich bei `float` um Gleitkommazahlen und nicht um echte reelle Zahlen mit unendlicher Präzision handelt, kann man nicht einfach `lineargrade(0) == 6.0` testen, da Rundungsfehler auftreten können (siehe Foliensatz 2). Stattdessen kann man die Funktion `isclose` aus dem `math`-Modul verwenden, die geringe Abweichungen vom tatsächlichen Ergebnis zulässt.

(b) Implementieren Sie eine Funktion `passed(p: float) -> bool` welche für eine Punktzahl p genau dann `True` zurückgibt, wenn die Klausur nach obigem Bewertungsschlüssel mindestens eine Note von 4.0 erzielt.

```
>>> passed(0)
False
>>> passed(27.5)
False
>>> passed(28)
True
>>> passed(33)
True
```

(c) Schreiben Sie die folgende Funktion so um, dass lediglich ein einziges `if`-Statement verwendet wird. Verwenden Sie hierzu mehrere `elif`-Anweisungen.

```
def mark(grade: float) -> str:
    if grade >= 6.0:
        return "F"
    else:
        if grade >= 5.0:
            return "E"
        else:
            if grade >= 4.0:
                return "D"
            else:
                if grade >= 3.0:
                    return "C"
                else:
                    if grade >= 2.0:
                        return "B"
                    else:
                        return "A"
```

Lösung:

```
def lineargrade(p: float) -> float:
    if p <= 12:
        return 6.0
    elif p >= 57:
        return 1.0
    else:
        return round((-5 / 45 * (p - 57) + 1) * 2) / 2

def passed(p: float) -> bool:
    return lineargrade(p) <= 4.0

def mark(grade: float) -> str:
    if grade >= 6.0:
        return "F"
    elif grade >= 5.0:
        return "E"
    elif grade >= 4.0:
        return "D"
    elif grade >= 3.0:
        return "C"
    elif grade >= 2.0:
        return "B"
    else:
        return "A"
```

Aufgabe 4.4 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet04` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, Bezug zur Vorlesung, Interessantes, benötigter Zeitaufwand, etc.).