Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel
Dr. Stefan Wölfl, Thorsten Engesser
Wintersemester 2015/2016

Universität Freiburg
Institut für Informatik

Übungsblatt 2

Abgabe: Freitag, 6. November 2015, 20:00 Uhr

Wichtige Hinweise: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet02` im Wurzelverzeichnis Ihrer Arbeitskopie des SVN-Repositories an. Ihre Lösungen werden dann entsprechend dem ersten Übungsblatt in Dateien in diesem Unterverzeichnis erwartet.

Beachten Sie bitte bei allen Aufgaben die *Hinweise zur Bearbeitung der Übungsaufgaben* unter der folgenden URL:

`http://gki.informatik.uni-freiburg.de/teaching/ws1516/info1/wiki/hinweise.html`

Überprüfen Sie, dass Sie alle Lösungen ins Repository hochgeladen haben (z.B. mit dem Befehl `svn status`). Überprüfen Sie auch die Webseite Ihres SVN-Unterverzeichnisses:

`https://daphne.informatik.uni-freiburg.de/ws1516/InformatikI/svn/$RZLOGIN`

Bewertet wird bei allen Aufgaben die letzte Version, die zur Deadline des Übungsblattes auf dem SVN-Server eingereicht ist.

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

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. Konsultieren Sie dazu in der Python-Shell auch die Hilfe zu den jeweiligen arithmetischen Funktionen.

(a)
```
>>> from math import log2
>>> res = 2 ** int(log2(128))
```

(b)
```
>>> from math import pi, sin, cos, radians
>>> res = cos(pi/4)**2 + sin(radians(45))**2
```

(c)
```
>>> from math import sqrt, floor, ceil
>>> res = sqrt(floor(2.3 * 7) * ceil(2 ** 3.7)) // 2
```

(d)
```
>>> res = 6 * round(0.1, 1) == round(.6, 1)
```

Aufgabe 2.2 (Variablen und ihre Werte; Datei: `variablen.txt`; Punkte: 5)

Angenommen, Sie geben in der Python-Shell die folgende Befehlssequenz ein. Welche der darin vorkommenden Variablen `a, i, f, s, x` sind an den mit (1), (2) und (3) markierten Stellen sichtbar? Welche Werte haben die an diesen Stellen jeweils sichtbaren Variablen?

```
>>> s = "spam"
>>> s = 3 * s
>>> a = s
>>> def foo(i):
...     global a
...     i *= 7
...     if i % 5 == 0:
...         s = "ham"
...         a = "egg"
...         return True
...
>>> foo(0)
True
>>>                    # (1)
>>> f = foo
>>> x = f(6)
>>> if x:
...     a = "egg"
... else:
...     s = "ham"
...
>>>                    # (2)
>>> x = not x
>>> i = int(x)
>>> while i < 3:
...     i += 1
...     s *= 2
...
>>>                    # (3)
```

Hinweis: Bearbeiten Sie die Aufgabe zunächst auf Papier und überprüfen Sie Ihre Antworten anschließend mit der Python-Shell oder dem Python-Tutor.

Aufgabe 2.3 (Stellenzahl in verschiedenen Stellenwertsystemen; Dateien: `lennum.py`, `test_lennum.txt`; Punkte: 5)

Definieren Sie eine Funktion `baselen`, die bei Eingabe einer nicht-negativen Zahl `n` und einer Zahl `base` ≥ 2 (beide vom Typ `int`) die Anzahl der Stellen von `n` im Stellenwertsystem mit der Basis `base` zurückgibt Für ungültige Eingaben von `n` oder `base` soll `None` zurückgegeben werden.

Hinweis: Zu Stellenwertsystemen siehe:

`https://de.wikipedia.org/wiki/Stellenwertsystem`.

Definieren Sie die Funktion ausgehend von folgendem Template:

```python
def baselen(n, base):
    """Calculate the length of a natural nummber in positional notation.

    Args:
      n (int): a natural number.
      base (int): base for the positional notation.

    Returns:
      int or None: int if `n` >= 0 and `base` >= 2, None otherwise.

    """
    # Anstelle dieses Kommentars folgt hier Ihre Definition
```

Testen Sie Ihre Funktion an geeigneten Beispielen, also z.B. mit:

```
>>> baselen(0, 7) == 1
True
>>> baselen(3, 2) == 2
True
>>> baselen(100, 8) == 3
True
>>> baselen(10000, 10) == 5
True
>>> baselen(0xABCDEF, 16) == 6
True
>>> baselen(-0b1010, 2) is None
True
>>> baselen(10, 1) is None
True
```

Bearbeiten Sie diese Aufgabe in der Python-IDE IDLE. Öffnen Sie in IDLE eine neue Datei mit dem Namen `lennum.py` und definieren Sie in dieser Datei Ihre Funktion `baselen`. Benutzen Sie dann die IDLE-Funktion `Run module (F5)`, um Ihre Funktion in der Python-Shell zu testen (mindestens vier Test-Beispiele wie oben). Sichern Sie die Shell, in der Sie Ihre Tests durchgeführt haben in der Datei `test_lennum.txt` und löschen Sie aus der Datei *alle* Zeilen mit Ausnahme Ihrer Test-Beispiele und deren Ausgaben in der Python-Shell (wie in obigem Beispiel). Committen Sie beide Dateien zum SVN-Repository.

Aufgabe 2.4 (Schulnoten; Datei: `grading.py`; Punkte: 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 Schülerinnen mit einer Punktzahl von 20% der erzielbaren Punkte (das sind 12 Punkte) oder weniger in jedem Fall die Note 6.0 erhalten, Schülerinnen 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 $-5/45 \cdot (p - 57) + 1$ berechnet. Dieser wird dann zur nächstgelegenen Notenstufe gerundet.

(a) Implementieren Sie die oben beschriebene Benotungsvorschrift als Funktion `lineargrade(p)`, die bei Eingabe einer Punktzahl `p` ≥ 0 (vom Typ `float` oder `int`) die zugehörige Note als `float` zurückgibt.

Testen Sie Ihre Funktion an geeigneten Beispielen, also z.B. mit:

```
>>> abs(lineargrade(0) - 6.0) < 1e-10
True
>>> abs(lineargrade(36.5) - 3.5) < 1e-10
True
>>> abs(lineargrade(37) - 3.0) < 1e-10
True
>>> abs(lineargrade(60.0) - 1.0) < 1e-10
True
```

(b) Im Folgenden bezeichnen wir eine Funktion wie in Aufgabenteil (a), die jeder nicht-negativen Punktzahl ≤ 60 einen der Notenwerte $1.0, 1.5, \ldots, 5.5, 6.0$ zuordnet, als *Benotungsfunktion*, sofern sie *monoton fallend* ist, d.h. einer besseren (größeren) Punktzahl auch immer eine gleich gute oder bessere (kleinere) Note zuordnet.

Implementieren Sie eine Funktion `print_grades` zur Ausgabe einer Notentabelle für eine beliebige solche Benotungsfunktion `grade`. Zu jeder erzielbaren Schulote, ausgehend von der schlechtesten bis zur besten, soll die hierfür minimal nötige ganzzahlige Punktzahl ausgegeben werden. Achten Sie auf eine ordentliche Formatierung Ihrer Ausgabe.

Testen Sie Ihre Funktion mit der in (a) definierten Benotungsfunktion `lineargrade`. Sofern Sie alles richtig gemacht haben, sollte Ihre Ausgabe so oder so ähnlich aussehen:

```
>>> print_grades(lineargrade)
grade   points
--------------
6.0     0
5.5     15
5.0     19
4.5     24
4.0     28
3.5     33
3.0     37
2.5     42
2.0     46
1.5     51
1.0     55
```

Aufgabe 2.5 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet02` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Interessantes, etc.).