Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel
Dr. Stefan Wölfl, Thorsten Engesser, Tim Schulte
Wintersemester 2016/2017

Universität Freiburg
Institut für Informatik

Übungsblatt 2

Abgabe: Freitag, 4. November 2016, 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 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/ws1617/info1/guide/hinweise.html`

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: 3)

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 = int(log2(64)) + 2 ** abs(1+1j)
```

(b)
```
>>> from math import sqrt, floor, ceil
>>> res = sqrt(floor(3.7)) // 2 or ("42" and ceil(1.1))
```

(c)
```
>>> from math import isclose
>>> res = 6 * isclose(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 verwendeten Namen `spam, egg, foo, ham, bar, eve` sind an den mit (1), (2) und (3) markierten Stellen sichtbar (d.h. definiert)? Welche Werte haben die an diesen Stellen jeweils sichtbaren Variablen?

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

```
>>> spam = "spam"
>>> egg = spam
>>> def foo(i):
...     if i % 2 == 0:
...         global egg
...         egg *= 2
...     else:
...         ham = egg + spam
...         return ham
...
>>>                      # (1)
>>> bar = foo
>>> eve = bar(5)
>>>                      # (2)
>>> eve = not eve
>>> eve = int(eve)
>>> while eve < 2:
...     ham = foo(eve)
...     spam += ham or "ham"
...     eve += 1
...
>>>                      # (3)
```

Aufgabe 2.3 (Zwölfersystem; Dateien: `dozenal.py`, `test_dozenal.txt`; Punkte: 5)

Das Duodezimalsystem verwendet neben den Ziffern 0 bis 9 auch Ziffern für die Zahlen Zehn und Elf (siehe auch: `https://de.wikipedia.org/wiki/Duodezimalsystem`). Wir benutzen im Folgenden die umgedrehte 2 für die Zehn ("ƻ"; `zehn = "\u1614"`) und die umgekehrte 3 für die Elf ("ɛ"; `elf = "\u0190"`).

Definieren Sie eine Funktion `duodec`, die bei Eingabe einer nicht-negativen Zahl `n` (vom Typ `int`) die Repräsentation der Zahl im Duodezimalsystem berechnet und als String zurückgibt. Für ungültige Eingaben von `n` soll `None` zurückgegeben werden.

Bearbeiten Sie diese Aufgabe in der Python-IDE IDLE. Öffnen Sie in IDLE eine neue Datei mit dem Namen `dozenal.py` und definieren Sie in dieser Datei Ihre Funktion `duodec` ausgehend von folgendem Template:

```
def duodec(n):
    """Calculate the duodecimal representation of a natural number.

    Args:
      n (int): a natural number.

    Returns:
      string or None: string if `n` >= 0, None otherwise.

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

Benutzen Sie dann die IDLE-Funktion `Run module (F5)`, um Ihre Funktion in der Python-Shell zu testen, z.B. mit:

```
>>> duodec(10) == chr(5652)
True
>>> duodec(11) == "\u0190"
True
>>> duodec(-1) is None
True
>>> duodec(278) == '1\u01902'
True
```

Kopieren Sie mindestens vier Test-Beispiele wie oben in eine Datei `test_dozenal.txt` und committen Sie beide Dateien zum SVN-Repository.

Aufgabe 2.4 (Osterformel; Datei: `easter.py`; Punkte: 3+2)

(a) Definieren Sie eine Funktion `easterdate(year)`, die zu einem Jahr `year` das genaue Datum des Ostersonntags (als String und im Format `'march DD'` bzw. `'april DD'`, also mit ausgeschriebenem Monat und ggf. führender Null) zurückgibt. Benutzen Sie dazu die ergänzte Osterformel nach Heiner Lichtenberg, die in dem Wikipedia-Artikel: `https://de.wikipedia.org/wiki/Gau%C3%9Fsche_Osterformel#Eine_erg.C3.A4nzte_Osterformel` dargestellt ist.

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

```
>>> easterdate(2016)
'march 27'
>>> easterdate(2017)
'april 16'
```

(b) Definieren Sie eine Funktion `printeaster(first, final)`, die eine schön formatierte Tabelle aller Osterdaten vom Jahr `first` bis inkusive zum Jahr `final` ausgibt. Ihre Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
>>> printeaster(2016, 2020)
year    day
----------------
2016    march 27
2017    april 16
2018    april 01
2019    april 21
2020    april 12
```

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.).