Einführung in die Programmierung

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

Universität Freiburg
Institut für Informatik
Wintersemester 2025

Übungsblatt 4

Abgabe: Montag, 10.11.2025, 9:00 Uhr

Abgaberegel: Typannotationen

In der Vorlesung haben Sie bereits Funktionen mit Typannotationen für Argumente und Rückgabewerte kennengelernt. Ab diesem Übungsblatt sind Typannotationen für **alle** von Ihnen implementierten Funktionen verpflichtend.

Verwenden Sie dabei möglichst präzise Typangaben. Gibt eine Funktion beispielsweise eine Liste von ganzen Zahlen zurück, sollte der Rückgabetyp `list[int]` lauten.

Eine vollständige Übersicht aller gültigen Abgaberegeln finden Sie hier auf unserer Website.

Aufgabe 4.1 (Funktionen über Listen; 3 Punkte; Datei: `lists.py`)

(a) (1 Punkt) Schreiben Sie eine Funktion `map_pot`, die eine Liste von ganzen Zahlen `xs` und eine ganze Zahl `n` als Argumente nimmt und eine Liste zurückgibt in der alle Zahlen aus `xs` mit `n` potenziert wurden.

Hinweis: Sie dürfen dabei annehmen, dass in `xs` keine `0` vorkommt, wenn `n` negativ ist.

Zum Beispiel:

```
>>> map_pot([], 1)
[]
>>> map_pot([-3, -1, 0, 2, 5], 0)
[1, 1, 1, 1, 1]
>>> map_pot([-3, -1, 0, 2, 5], 2)
[9, 1, 0, 4, 25]
>>> map_pot([-3, -1, 2, 5], -3)
[-0.037037037037037035, -1.0, 0.125, 0.008]
```

(b) (1 Punkte) Schreiben Sie eine Funktion `filter_n` die eine Liste von ganzen Zahlen `xs` und eine ganze Zahl `n` als Argumente nimmt. Die Funktion soll ein Liste zurückgeben, in der alle Zahlen aus `xs` enthalten sind, die ungleich `n` sind.

Zum Beispiel:

```
>>> filter_n([], 5)
[]
```

```
>>> filter_n([-1, -2, 0, 3, 5, -4], 3)
[-1, -2, 0, 5, -4]
>>> filter_n([1, 1, 1, 1], 1)
[]
```

(c) (1 Punkte) Schreiben Sie eine Funktion `safe_map_pot`, die sich wie `map_pot` aus Aufgabenteil (a) verhält, aber auch bei negativem `n`, die Zahl `0` in `xs` erlaubt. Tritt ein solcher Fall auf, so sollen alle `0`en aus `xs` ignoriert werden und nicht in der Rückgabeliste erscheinen. Verwenden Sie hierzu Ihre selbst definierten Funktionen `map_pot` und `filter_n` aus den Aufgabenteilen (a) und (b).

Zum Beispiel:

```
>>> safe_map_pot([], 1)
[]
>>> safe_map_pot([-3, -1, 0, 2, 5], 0)
[1, 1, 1, 1, 1]
>>> safe_map_pot([-3, -1, 0, 2, 5], 2)
[9, 1, 0, 4, 25]
>>> safe_map_pot([-3, -1, 0, 2, 5], -3)
[-0.037037037037037035, -1.0, 0.125, 0.008]
```

Aufgabe 4.2 (Eulersche Exponentialfunktion; 3 Punkte; Datei: `euler.py`)
Die eulersche Exponentialfunktion $exp(x)$ kann durch die Potenzreihe

$$exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

dargestellt werden. Dabei ist $n! = \prod_{k=1}^{n} k$ die Fakultät von n.

Im Allgemeinen lässt sich diese unendliche Summe nicht exakt berechnen. Wir können jedoch eine Annäherung $S_m(x) \approx exp(x)$ berechnen, indem wir die ersten m Summanden der Reihe aufsummieren:

$$S_m(x) = \sum_{n=0}^{m} \frac{x^n}{n!}$$

Schreiben Sie eine Funktion `approx_euler`, die eine Gleitkommazahl `x` und eine ganze Zahl `m` als Argumente nimmt und die m-te Annäherung $S_m(x)$ berechnet und zurückgibt. Sie dürfen davon ausgehen, dass `m` nicht negativ ist.

Zum Beispiel:

```
>>> approx_euler(1.0, 0)
1.0
>>> approx_euler(1.0, 1)
2.0
```

```
>>> approx_euler(1.0, 4)
2.708333333333333
>>> approx_euler(1.0, 16)
2.718281828459043
>>> approx_euler(3.14, 10)
23.094025449451095
```

Aufgabe 4.3 (Zahlen und Farben; 4 Punkte; Datei: `hex.py`)

(a) (2 Punkte) Hexadezimalzahlen sind Zahlen im Zahlensystem zur Basis 16. Die ersten neun Ziffern `0` bis `9` entsprechen dabei den gleichen Werten wie im Dezimalzahlsystem. Die Werte von `10` bis `15` werden hingegen durch die Buchstaben `A` bis `F` dargestellt. Um eine Hexadezimalzahl $h = h_n h_{n-1} \dots h_1 h_0$ in eine Dezimalzahl d zu konvertieren, wird die folgende Formel verwendet:

$$d = \sum_{i=0}^{n} h_i \cdot 16^i$$

Beispielsweise entspricht die Hexadezimalzahl 3AF2 der Dezimalzahl $3 \cdot 16^3 + 10 \cdot 16^2 + 15 \cdot 16^1 + 2 \cdot 16^0 = 15090$.

Schreiben Sie eine Funktion `to_decimal`, die eine Hexadezimalzahl `hex` als Zeichenkette (`str`) entgegennimmt und diese in eine Dezimalzahl (`int`) umwandelt und zurückgibt. Sie können dabei annehmen, dass die Eingabe aus mindestens einem Zeichen besteht und nur die Zeichen `0-9` und `A-F` vorkommen.

Verwenden Sie bei Ihrer Implementierung **nicht** die `int`-Funktion aus der Standardlibrary!

Zum Beispiel:

```
>>> to_decimal("0")
0
>>> to_decimal("17")
23
>>> to_decimal("FF")
255
>>> to_decimal("00FF")
255
>>> to_decimal("3AF2")
15090
```

Hinweis: Sie *können* die Funktion `ord`[1]-Funktion verwenden, um die Zeichen A-F in Dezimalzahlen umzuwandeln, ohne eine Fallunterscheidung zu verwenden:

[1] https://docs.python.org/3/library/functions.html#ord

```
>>> ord("A")  # ASCII Wert von A
65
>>> ord("C")  # ASCII Wert von C
67
>>> ord("C") - ord("A") + 10  # Differenz plus 10
12
```

(b) (2 Punkte) Farben werden häufig im dreidimensionalen Rot-Grün-Blau (RGB)-Farbraum dargestellt. Dieser Farbraum basiert auf einem 3-Tupel, das jeweils die Intensität der Farben Rot, Grün und Blau im Bereich von 0 bis 255 beschreibt. Statt das 3-Tupel direkt zu verwenden, wird es oft als sechsstellige Hexadezimalzahl dargestellt. Die ersten beiden Ziffern repräsentieren die Intensität von Rot, die nächsten beiden die Intensität von Grün und die letzten beiden die Intensität von Blau.

Zum Beispiel:

$$ACB5F2 \leftrightarrow (\underbrace{172}_{=AC_{16}}, \underbrace{181}_{=B5_{16}}, \underbrace{242}_{=F2_{16}})$$

Schreiben Sie eine Funktion `parse_and_convert`, die eine RGB-Hexadezimalzahl `hex_color` entgegennimmt und ein RGB-Tupel in Dezimaldarstellung zurückgibt. Verwenden Sie zur Umwandlung der Hexadezimalzahlen in Dezimalzahlen die Funktion `to_decimal` aus Aufgabenteil (a).

Zum Beispiel:

```
>>> parse_and_convert("000000")
(0, 0, 0)
>>> parse_and_convert("FFFFFF")
(255, 255, 255)
>>> parse_and_convert("FA7BCC")
(250, 123, 204)
>>> parse_and_convert("123456")
(18, 52, 86)
```

Aufgabe 4.4 (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 `5.5 h` steht für 5 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.