Einführung in die Programmierung

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

Universität Freiburg
Institut für Informatik
Wintersemester 2023

Übungsblatt 6

Abgabe: Montag, 27.11.2023, 9:00 Uhr morgens

Hinweis zu Typvariablen

In der Vorlesung haben Sie Typvariablen kennengelernt um generische Typen zu beschreiben. Wenn Sie beispielsweise eine Funktion schreiben, die 4 Werte vom gleichen Typ als Argumente nimmt, würden Sie die Typannotation wie folgt schreiben:

```
def foo[T](a: T, b: T, c: T, d: T):
    ...
```

Wenn nun aber nur sowohl a und b als auch c und d paarweise den gleichen Typ besitzen sollen, können Sie dafür *mehrere* Typvariablen verwenden:

```
def bar[T, U](a: T, b: T, c: U, d: U):
    ...
```

Aufgabe 6.1 (Typvariablen; 10 Punkte; Datei: `typevars.py`)

In dieser Aufgabe sollen Sie Funktionen schreiben, deren Typannotationen Typvariablen verwenden. Pro Teilaufgabe wird jeweils 1 Punkt für die richtige und möglichst genaue Typannotation und 1 Punkt für die Implementierung der Funktion vergeben.

(a) Schreiben Sie eine Funktion `head`, die eine Liste `xs` als Argument nimmt und das erste Element der Liste zurückgibt, falls dieses existiert. Ansonsten soll `None` zurückgegeben werden.

(b) Schreiben Sie eine Funktion `tail`, die eine Liste `xs` als Argument nimmt und alle Elemente der Liste bis auf das erste zurückgibt. Hat die Liste keine Elemente, so soll `None` zurückgegeben werden.

(c) Schreiben Sie eine Funktion `concat`, die eine Liste `xss` von Listen mit jeweils gleichem Typ als Argument nimmt, die Elemente von `xss` (in der selben Reihenfolge) miteinander verkettet, und das Ergebnis zurückgibt.

(d) Schreiben Sie eine Funktion `zip`, die zwei beliebige Listen `xs` und `ys` als Argumente nimmt und eine Liste von Tupeln zurückgibt, die die Elemente aus `xs` und `ys` paarweise beinhaltet. Sind `xs` und `ys` nicht gleich lang, so sollen die restlichen Elemente der längeren Liste ignoriert werden.

(e) Schreiben Sie eine Funktion `assoc`, die ein beliebiges Tupel `t` der Form `((x, y), z)` als Argument nimmt und das Tupel `(x, (y, z))` zurückgibt.

Falls Sie Ihre Funktionen mit `assert`s testen möchten, vergessen Sie nicht, diese hinter `if __name__ == '__main__'` zu schreiben.

Aufgabe 6.2 (Mastermind; 10 Punkte; Datei: `mastermind.py`)

In dieser Aufgabe sollen Sie eine Version des Spiels Mastermind mit Strings programmieren. Das Spiel soll wie folgt funktionieren: Das Programm erstellt zu Beginn eine zufällige Folge von Zeichen einer festen Länge. Die Benutzerin rät nun ein Wort. Das Programm gibt dann Rückmeldung darüber, wie viele der Zeichen der Eingabe mit dem Lösungswort übereinstimmen. Ein `"X"` ('perfekt') steht jeweils dafür, dass ein Zeichen im Lösungswort vorkommt und die richtige Position hat. Ein `"-"` ('richtig', aber nicht 'perfekt') steht jeweils dafür, dass ein Zeichen im Lösungswort vorkommt, aber nicht die richtige Position hat. Die Benutzerin rät dann solange weiter, bis jedes Zeichen im Lösungswort 'perfekt' erraten wurde. Ein möglicher Programmaufruf mit dem Lösungswort `"BAAEC"` kann wie folgt aussehen:

```
Länge: 5 Zeichen: ABCDE
ABCDE
Antwort: ----
BACDE
Antwort: XX--
BADDE
Antwort: XX-
BAECC
Antwort: XXX-
BABEC
Antwort: XXXX
BAAEC
Antwort: XXXXX
```

Um die Umsetzung zu vereinfachen, teilen wir das Programm in 5 Funktionen auf. Zu den ersten 4 haben wir auch wieder Tests bereitgestellt. Diese finden Sie in Ihrem git-Repo[1]

(a) `remove_char`; 2 Punkte
Schreiben Sie eine Funktion `remove_char`, die einen String `s` und einen String `c` der Länge 1[2] als Argumente nimmt. Die Funktion soll eine Kopie von `s` zurückgeben, in der jedoch das erste Vorkommen von `c` entfernt wurde. Kommt `c` nicht in `s` vor, soll `s` unverändert zurückgegeben werden.[3]

```
assert remove_char("DAEAB", "A") == "DEAB"
assert remove_char("hallo", "x") == "hallo"
```

(b) `perfect_chars`; 2 Punkte
Schreiben Sie eine Funktion `perfect_chars`, die einen String `inp` und einen String `sol` als Argumente nimmt.[4] Die Funktion soll einen String mit allen Zeichen zurückgeben, die in `inp` und `sol` in ihrer Position übereinstimmen (also 'perfekt' sind). Die Reihenfolge der Zeichen soll sich nicht ändern.

[1] https://git.laurel.informatik.uni-freiburg.de/2023WS-EiP?

[2] Sie dürfen davon ausgehen, dass die Länge immer 1 ist.

[3] Sie können diese Funktion für die folgenden Aufgabenteile verwenden, um ein Zeichen aus einem String zu 'entfernen'.

[4] Sie dürfen annehmen, dass `inp` und `sol` die gleiche Länge haben.

```
assert perfect_chars("ABBBBC", "BBAAAA") == "B"
assert perfect_chars("DABACC", "AAEACD") == "AAC"
```

(c) `correct_chars`; 2 Punkte
Schreiben Sie eine Funktion `correct_chars`, die einen String `inp` und einen String `sol` als Argumente nimmt.[4] Die Funktion soll einen String mit allen Zeichen aus `inp` zurückgeben, die in `sol` vorkommen (also 'richtig' sind). Ist ein Zeichen in `sol` mit einem Zeichen aus `inp` abgedeckt, darf es nicht erneut betrachtet werden. Die Zeichen sollen zudem in der Reihenfolge zurückgegeben werden, wie sie in `inp` vorkommen.

```
assert correct_chars("ABBDC", "AACCD") == "ADC"
assert correct_chars("AACCD", "ABBDC") == "ACD"
```

(d) `compare`; 2 Punkte
Schreiben Sie eine Funktion `compare`, die eine Benutzereingabe `inp` mit dem Lösungswort `sol` nach den Regeln im Einführungstext vergleicht und ein Tupel aus zwei Zahlen zurückgibt.[4] Die erste Zahl ist die Anzahl der 'perfekten' Zeichen und die zweite Zahl ist die Anzahl der 'richtigen' aber nicht 'perfekten' Zeichen.

```
assert compare("BAECC", "BAAEC") == (3, 1)
assert compare("ABCDE", "EADCB") == (0, 5)
```

(e) `game`; 2 Punkte
Schreiben Sie eine Funktion `game`, die eine Ganzzahl `length` und einen String `symbols` als Argumente nimmt. `length` steht für die Länge des Lösungswort und `symbols` für die möglichen Symbole. `game` soll nun das Spiel wie im Einführungstext beschrieben ausführen. Sie dürfen davon ausgehen, dass die Benutzerin immer eine gültige Eingabe tätigt.[5]
Mit `"".join(random.choices(symbols, k=length))` können Sie ein zufälliges Wort generieren.

(f) 0 Punkte
Starten Sie ein Spiel mit sinnvollen Werten. Vergessen Sie nicht, Ihren Funktionsaufruf hinter `if __name__ == '__main__'` zu schreiben.
Viel Spaß beim Rätseln!

Aufgabe 6.3 (Erfahrungen; 0 Punkte; 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.

[5] Sie dürfen ungültige Eingaben aber auch freiwillig abfangen.