

Datenstrukturen: Listen, Tupel, Maps

1 Listen

- Schreiben Sie eine Funktion `sub(list_a, list_b)`, die `list_b` von `list_a` „subtrahiert“. Mit anderen Worten: es sollen all jene Elemente von `list_a` zurückgegeben werden, die nicht in `list_b` enthalten sind!
- Bestimmen Sie das Minimum unter den Elementen einer Liste.
- Füllen Sie eine Liste der Länge 20 mit Zufallszahlen. Überlegen Sie sich einen einfachen Algorithmus, mit dem eine Liste nur mit Vergleich- und Vertauschoperationen sortiert werden kann. Wie könnte ein solcher Algorithmus aussehen? Wie würden Sie im Kopf Karten oder ähnliches sortieren? Versuchen Sie zuerst einen Algorithmus zu entwickeln, bevor Sie etwas implementieren! Sie sollten insbesondere in der Lage sein, einem Bekannten zu erklären wie dieser Algorithmus funktioniert!

Implementieren Sie ihren Algorithmus. Testen Sie ihre Implementierung mit der zufällig gefüllten Liste! Verwenden Sie *keine* vorimplementierte Sortierfunktion. *Hinweis*: Sie können Zufallszahlen mit der Funktion `random` aus dem Modul `random` erzeugen.

2 Tupel

- Implementieren Sie eine Funktion `add(tuple_a, tuple_b)`, die zwei Punkte eines kartesischen Koordinatensystems addiert. Es gilt $(a, b) + (x, y) = (a + x, b + y)$.
- Schreiben Sie eine Funktion `dist(tuple_a, tuple_b)`, die die Distanz zweier Punkte eines kartesischen Koordinatensystems zurückgibt.
- Füllen Sie eine Liste mit einem Deck Pokerkarten. Speichern Sie Karten als Tupel der Form `(value, suit)`.

3 Maps

- Implementieren Sie mit der Hilfe einer Map ein kleines Telefonbuch. Verwenden Sie den Namen als Schlüssel, und die Telefonnummer als Wert. Implementieren Sie eine Funktion `printContactInfo(phonebook, name)`, mit der das Telefonbuch durchsucht werden kann. Gibt es einen Eintrag unter dem Namen `name` werden Name und Telefonnummer auf der Kommandozeile ausgegeben. Gibt es keinen Eintrag unter diesem Namen, geben Sie eine Fehlermeldung aus (etwa: „No such name“). Implementieren Sie auch eine Funktion, mit der alle Kontakte ausgegeben werden können!

4 Zusatzübungen

Sollten Sie bereits mit allen Übungen fertig sein, können Sie versuchen folgende Zusatzprobleme zu lösen:

- Implementieren Sie eine Funktion mit der Sie das Kartenspiel aus Aufgabe 2 „mischen“ können.

- Erweitern Sie ihre Implementierung des Telefonbuchs aus Aufgabe 3, sodass Sie nicht nur Telefonnummer, sondern auch Geschlecht, Titel und Adresse einer Person speichern können. Ergänzen Sie das Programm um Funktionen zum anlegen und löschen von Einträgen!
- Modifizieren Sie ihren Sortieralgorithmus, sodass er nach jedem Schritt den aktuellen Zustand der zu sortierenden Liste grafisch ausgibt. Zum Beispiel könnten Sie die Zahl n als Zeichenkette der Länge n darstellen. Jede Zeile entspricht hier einer Zahl, die Liste $[10, 3, 5]$ wäre also:

```
#####  
###  
#####
```

Da der Computer natürlich viel zu schnell ist um eine sinnvolle Ausgabe zu erhalten, lassen Sie ihn nach jedem Schritt 1 Sekunde schlafen.