Repetition 4

Dr. Günter Kolousek

21. Juli 2015

Lege wiederum ein Verzeichnis an. Nennes es `20_repetition4`! In diesem Verzeichnis sollen alle Dateien der jeweiligen Einheit abgelegt werden.

1 Wiederholung 4

1. Schreibe eine Funktion `length(a)` in einer Datei `lists2.py`, die eine Liste von Zahlen bekommt und die Anzahl der darin enthaltenen Zahlen zurückliefert. Die eingebaute Funktion `len` darf nicht verwendet werden.
2. Schreibe eine Funktion `count_numbers(a)` (wieder im Modul `lists2`), die eine Liste erhält. Jedes Listenelement ist entweder eine Zahl oder eine Liste von Zahlen. Die Funktion soll die Anzahl **aller** enthaltenen Zahlen zurückliefern:

   ```
   >>> count_numbers([1, 2, 3]):
   3
   >>> count_numbers([1, [2, 3, 4], [5], 6])
   6
   ```

3. Schreibe eine Funktion `max_numbers(a)` (wieder im Modul `lists2`), die eine Liste erhält. Jedes Listenelement ist entweder eine Zahl oder eine Liste von Zahlen. Die Funktion soll das Maximum **aller** enthaltenen Zahlen zurückliefern:

   ```
   >>> max_numbers([2, 5, 1]):
   5
   >>> max_numbers([1, [2, 6, 5], [4], 3])
   6
   ```

4. Schreibe eine Funktion `add(a, b)` (wieder im Modul `lists2`), die 2 gleich lange Listen mit Zahlen als Parameter bekommt und eine **neue** Liste zurückliefert, die jeweils die Summe der Listenelemente enthält. Z.B.:

```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> add(a, b)
[2, 4, 6]
>>> a
[1, 2, 3]
>>> b
[1, 2, 3]
```

5. Schreibe eine Funktion `mul(a, b)` (wieder im Modul `lists2`), die 2 Listen mit Zahlen als Parameter erhält. Jedes Listenelement von `b` soll mit dem korrespondierenden Listenelement von `a` multipliziert werden und eine **neue** Liste zurückgeliefert werden.

   Sind die beiden Listen ungleich lang, dann hat die Ergebnisliste als Länge das Minimum der Längen der beiden Listen:

   ```
   >>> mul([1, 2, 3, 4, 5], [2] * 3)
   [2, 4, 6]
   >>> mul([1, 2, 3], [2] * 6)
   [2, 4, 6]
   ```

6. Schreibe eine Funktion `sub(a, b)` (wieder im Modul `lists2`), die 2 Listen mit Zahlen als Parameter erhält. Jedes Listenelement von `b` soll von dem korrespondierenden Listenelement von `a` abgezogen (subtrahiert) werden. Das Ergebnis wird direkt in `a` abgelegt und zurückgeliefert::

   ```
   >>> a = [1, 2, 3]
   >>> b = list(range(5))
   >>> sub(a, b)
   [1, 1, 1]
   >>> a
   [1, 1, 1]
   >>> b
   [0, 1, 2, 3, 4]
   ```

7. Schreibe eine Funktion `filter_greater(a, number)`, die die Liste aller Zahlen der Liste `a` zurückliefert, die größer als `number` sind:

   ```
   >>> filter_greater([5, 2, 3, 8, 4, 1], 3)
   [5, 8, 4]
   ```

8. Schreibe eine Funktion `sum_values(d)` in einer Datei `dict.py`, die alle Werte eines Dictionary addiert und das Ergebnis zurückliefert. Verwende die `values()` Methode (alle Werte des Dictionary sind Zahlen).

9. Schreibe eine Funktion `mul_values(d)` im Modul `dict`, die alle Werte eines Dictionary miteinander multipliziert und das Ergebnis zurückliefert. Verwende die `keys()` Methode (alle Werte sind wieder Zahlen).

10. Schreibe eine Funktion `sub_values(d)` wieder im Modul `dict`, die alle Werte eines Dictionary subtrahiert und das Ergebnis zurückliefert. D.h. von dem ersten Wert wird der nächste abgezogen, davon wiederum der Nächste und so weiter. Verwende die `items()` Methode (alle Werte sind wieder Zahlen).

    Was ist hier das Problem?

11. Schreibe eine Funktion `sort_values(d)` (wiederum im Modul `dict`), die alle Werte eines Dictionary aufsteigend sortiert und diese Werte als Liste zurückliefert.

12. Schreibe eine Funktion `sort_values2(d)` (wiederum im Modul `dict`), die alle Werte eines Dictionary aufsteigend sortiert und diese Werte als Liste zurückliefert. Als Werte sind in dem Dictionary jetzt allerdings keine Zahlen sondern Tupels enthalten. Als Sortierkriterium soll das Element mit dem Index 1 herangezogen werden.