Repetition 5

Dr. Günter Kolousek

21. Juli 2015

Lege wiederum ein Verzeichnis an. Nennes es `23_repetition5`! In diesem Verzeichnis sollen alle Dateien der jeweiligen Einheit abgelegt werden.

1 Wiederholung 5

Diese Wiederholung beschäftigt sich nochmals mit:

- Mengen
- einfache und verschachtelten Schleifen
- Schreibe eine Funktion `cross_product(a, b)` in einer Datei `sets2.py`, die das Kreuzprodukt der Mengen a und b berechnet und zurückliefert:

  ```
  >>> cross_product({1, 2, 3}, {4, 5})
  {(1, 4), (1, 5), (2, 5), (3, 4), (2, 4), (3, 5)}
  ```

- Schreibe eine Funktion `in_set(e, a)` wiederum im Modul `sets2`, die `True` zurückliefert, wenn das Objekt `e` in der Menge `a` enthalten ist, anderenfalls `False`. Der Operator `in` darf nicht verwendet werden.
- Schreibe eine Funktion `output_mul_table(n)`, die die Multiplikationstabelle bis zur übergebenen Zahl `n<10` ausgibt. Die Tabelle ist so zu formatieren, dass max. das kleine 1x1 schön dargestellt werden kann. Beispielhafte Ausgabe für n=5:

  ```
  x   1   2   3   4   5
  1   1   2   3   4   5
  2   2   4   6   8  10
  3   3   6   9  12  15
  4   4   8  12  16  20
  5   5  10  15  20  25
  ```

- Schreibe eine Funktion `reverse_list(lst)`, die alle Elemente der übergebenen Liste in umgekehrter Reihenfolge durch Beistriche getrennt ausgibt:

```
>>> reverse_list([1, "a", 2.5])
2.5,a,1
```

- Schreibe eine Funktion `to_str(lst)`, die eine Liste als Paramter bekommt und die Elemente dieser Liste als einen durch Beistriche getrennten String zurückliefert:

```
>>> to_str([1, "a", 2.5])
'1,a,2.5'
```

- Schreibe eine Funktion `reverse_number(n)`, die eine ganze Zahl als Argument bekommt und diejenige Zahl zurückliefert, die entsteht wenn man die Zahl von hinten nach vorne liest::

```
>>> reverse_number(123)
321
```

- Schreibe eine Funktion `is_palindrome(n)`, die überprüft, ob die übergebene Zahl ein Palindrom darstellt, dann liefert diese `True` zurück anderenfalls `False`. Verwende die Funktion `reverse_number`. Beispiel:

```
>>> is_palindrome(123)
False
>>> is_palindrome(12321)
True
```

- Schreibe eine Funktion `add(lst)`, die alle Elemente einer zweidimensionalen Liste aus Zahlen addiert und das Ergebnis zurückliefert:

```
>>> add([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
45
```

- Schreibe eine Funktion `minmax(lst)`, die das kleinste und das größte Element einer zweidimensionalen Liste als Tupel zurückliefert:

```
>>> minmax([[4, 5, 6], [7, 8, 9], [1, 2, 3]])
(1, 9)
```

- Schreibe eine Funktion `change_case(s)`, die einen String als Parameter bekommt und eine String zurückliefert, in dem jeder Kleinbuchstabe in einen Großbuchstaben und jeder Großbuchstabe in einen Kleinbuchstaben gewandelt sind. Alle anderen Zeichen sollen gleich bleiben:

```
>>> change_case("aABb1 _Cc")
'AabB1 _cC'
```