Unit 10

Dr. Günter Kolousek

21. Juli 2015

Lege wiederum ein Verzeichnis an. Nennes es `13_unit10`! In diesem Verzeichnis sollen alle Dateien der jeweiligen Einheit abgelegt werden.

1 Schulübungen

In diesem Aufgabenblock sind ab dem Punkt 3 auschließlich `for` Anweisungen **ohne** `range()`-Ausdrücke zu verwenden, wenn Schleifen notwendig sind und nicht explizit etwas anderes verlangt wird. Damit ist gemeint, dass dazu direkt über die Sequenz iteriert werden soll!

1. Schreibe ein Programm `fibonacci.py`, das für eine vom Benutzer eingegebene Zahl n, alle n ersten Zahlen der Fibonacci-Folge ermittelt und ausgibt.

   Die Fibonacci-Folge ist eine unendlich lange Folge von Zahlen, die sich dadurch berechnet, dass sich jede Zahl dieser Folge aus der Summe der beiden vorhergehenden Zahlen berechnet. Die beiden ersten Zahlen dieser Folge sind jeweils 1! D.h. die Fibonacci-Folge beginnt folgendermaßen:

   1, 1, 2, 3, 5, 8, 13, 21,...

   Als Formel angeschrieben ergibt sich das n.te Glied dieser Folge:

   $$x_1 = 1$$
   $$x_2 = 1$$
   $$x_n = x_{n-1} + x_{n-2}$$

   Für n mit dem Wert 7 soll die Ausgabe folgendermaßen aussehen:

```
1
1
2
3
5
8
13
```

Für dieses Beispiel eignet sich ebenfalls gut eine `while` Schleife.

2. Schreibe ein Programm `arithmean3.py`, das den Benutzer nach einer unbestimmten Anzahl von Zahlen fragt und das arithmetische Mittel zurückliefert.

   Die Eingabe der Zahlen wird abgeschlossen, wenn der Benutzer `q` eingibt.

   Das könnte folgendermaßen aussehen:

   ```
   Bitte geben Sie eine Zahl ein: 10
   Bitte geben Sie eine Zahl ein: 20
   Bitte geben Sie eine Zahl ein: q

   Das arithmetische Mittel beträgt: 15
   ```

   Für das Aufsummieren der Zahlen soll der Operator `+=` verwendet werden.

   In diesem Beispiel ist jedoch nur eine `while` Schleife möglich!

3. Schreibe ein Programm `check_triangle.py`, das den Benutzer nach drei Seiten eines Dreieckes fragt und ausgibt, ob es sich um ein gültiges Dreieck handelt oder nicht.

   Ein Dreieck ist gültig, wenn die Summe zweier (beliebiger) Seiten stets größer oder gleich der dritten Seite ist.

4. Schreibe eine Funktion `is_divisible(a, b)`, die `True` zurückliefert, wenn `a` durch `b` teilbar ist, anderenfalls `False`.

   Tritt ein Fehler auf, dann soll ebenfalls `False` zurückgeliefert werden.

5. Schreibe ein Programm `leap_year.py`, das überprüft, ob eine eingegebene Jahreszahl ein Schaltjahr ist.

   Schreibe dazu eine eigene Funktion `check_leap_year(year)`, die `True` zurückliefert, wenn es sich um ein Schaltjahr handelt, anderenfalls `False`.

   Seit Ende 1582 gilt der Gregorianische Kalender, bei dem folgende Regel für die Schaltjahrbestimmung anzuwenden sind:

   - In allen Jahren, deren Jahreszahl durch vier teilbar ist, ist der 29. Februar ein Schalttag und damit ist dieses Jahr ein Schaltjahr.

- Eine Ausnahme bilden allerdings die vollen Jahrhundertjahre 1700, 1800, 1900 usw., auch Säkularjahre genannt. Hiervon erhalten nur diejenigen einen Schalttag, deren Jahreszahl durch 400 teilbar ist. Jedes vierte Säkularjahr ist somit ein Schaltjahr.
- Für alle Jahre kleiner oder gleich 1582 liefert die Funktion `False`, weil da das Schaltjahr nicht definiert ist.
- Trifft keine der obigen Regeln zu, dann handelt es sich um kein Schaltjahr.

6. Schreibe ein Programm `name_spaces1.py`, das den Benutzer nach einem Namen fragt und alle Zeichen des Namens ausgibt, aber mit je einem Leerzeichen zwischen jeweils zwei Zeichen.

   Das sollte so aussehen, wenn der Benutzer "Maxi" eingibt:

   ```
   Bitte geben Sie Ihren Namen ein: Maxi
   M a x i
   ```

7. Schreibe ein Programm `name_spaces2.py`, das wie das wie `name_spaces1.py` funktioniert, jedoch eine Funktion `name_spaces(s)` enthält, die einen String als Parameter erhält und einen String zurückliefert, der den ursprünglichen String enthält jedoch mit je einem Leerzeichen zwischen je zwei Zeichen.

8. Probiere folgendes im interaktiven Interpreter:

   - 2^{10}
   - 2^{100}
   - 2^{1000}
   - 2^{10000}

9. Schreibe ein Programm `and2.py`, das eine Tabelle für das logische UND (engl. and) ausssieht, wobei jetzt eine `for` Schleifen über ein Tupel von Tupeln jeweils mit den Wahrheitswerten `True` und `False` verwendet werden soll. Solch eine Sequenz sieht folgendermaßen aus:

   ```
   ((False, False), (False, True),...)
   ```

   Die Ausgabe soll folgendermßen aussehen:

   ```
   a | b | a and b
   --+---+--------
   0 | 0 |    0
   0 | 1 |    0
   1 | 0 |    0
   1 | 1 |    1
   ```

Verwende wiederum die `format`-Methode!

10. Schreibe ein Programm `or2.py`, das analog zum vorhergehenden Beispiel eine Tabelle für die logische Operation ODER (engl. or) ausgibt.

    Jetzt sollen 2 verschachtelte `for` Schleifen über eine Sequenz von `True` und `False` verwendet werden!

11. Schreibe weiters ein Programm `logical2.py`, das analog zu den vorhergehenden Beispielen eine Tabelle für den logischen Ausdruck `a and b xor not c` ausgibt.

2 Hausübung

Kapitel 10 lesen!