Unit 9

Dr. Günter Kolousek

21. Juli 2015

Lege wiederum ein Verzeichnis an. Nennes es `12_unit9`! In diesem Verzeichnis sollen alle Dateien der jeweiligen Einheit abgelegt werden.

1 Schulübungen

Verwende in dieser Einheit **ab dem Punkt 2** ausschließlich `while`Schleifen!

1. Kopiere `regular_polygon2.py` auf `regular_polygon3.py` und schreibe das Programm so um, sodass jetzt jeweils die Farbwerte in Zahlen eingegeben werden können. Jede Farbe ist somit ein Tupel mit den Zahlen für den Rotanteil, den Grünanteil und den Blauanteil.
2. Weitere Programme zum Üben des Akkumulator-Musters:
   a) Schreibe ein Programm, das die ersten 100 geraden Zahlen aufsummiert und auf der Konsole ausgibt.
   b) Schreibe ein Programm, das die ersten 50 ungeraden Zahlen aufsummiert und auf der Konsole ausgibt.
   c) Schreibe ein Programm, das den Durchschnitt der ersten 100 geraden Zahlen ermittelt und auf der Konsole ausgibt.
3. Schreibe ein Programm `show_numbers.py`, das eine Tabelle ausgibt, die die Zahlen von 1 bis 32 sowohl in Dezimaldarstellung als auch in Binär- und Hexadezimaldarstellung enthält. Verwende die Funktionen `bin()` und `hex()`.
4. Schreibe in einer Datei `factorial.py`, eine Funktion `fak(n)`, die die Faktorielle (auch Fakultät genannt) einer eingegebenen Zahl n ermittelt und zurückliefert. Die Faktorielle ist das Produkt aller n ersten Zahlen.

   $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$

   Teste im interaktiven Interpreter!

5. Schreibe ein Programm `e.py`, das die Zahl e berechnet.

   $e = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \frac{1}{1 \cdot 2 \cdot 3 \cdot 4} + ...$

   Das Programm soll insgesammt n+1 Summanden aufsummieren. n wird wieder vom Benutzer eingegeben.

   In den Brüchen kommt jeweils die Faktorielle vor. Verwende dazu das schon entwickelte Modul `factorial`.

6. Schreibe ein Programm `ggt.py`, das von dem Benutzer zwei Zahlen erfragt und den größten gemeinsamen Teiler ermittelt und ausgibt.

   Dazu soll eine Funktion `ggt(a, b)` in einer Datei `utils.py` geschrieben werden.

   Der euklidische Algorithmus zum Errechnen dess ggT funktioniert folgendermaßen (in Pseudocode):

   ```
   solange b nicht gleich 0:
       h = a mod b
       a = b
       b = h
   ```

   Nach Ausführung ist in `a` der ggT abgelegt!

7. Schreibe ein Programm `kgv.py`, das von dem Benutzer zwei Zahlen erfragt und das kleinste gemeinsame Vielfache der beiden Zahlen ermittelt und danach ausgibt.

   Das kgV berechnet sich folgendermaßen:

   $\frac{ab}{\text{ggt}(a,b)}$

   Verwende dazu die schon entwickelte Funktion `ggt()` aus dem Modul `utils`.

8. Einfache Ausdrücke mit booleschen Operatoren. Was ergibt?

   a) `True or False`, `True and False`, `not(False) and True`

   b) `not (True or False)`, `(not True) and (not False)`, `not (True and False)`

   c) `True or 42`, `42 or False`, `True and 0.0`

   d) `"a" and "b"`, `"a" or "b"`, `"" and "sad"`

9. Forme folgende Ausdrücke am Papier mit Hilfe des Gesetzes von De Morgan um:

   a) $\neg$ (a $\wedge$ b), $\neg$ (a $\vee$ b)

   b) $\neg$ (a $\wedge \neg$ b), $\neg$ ($\neg$ a $\vee$ b)

   c) $\neg$ (a $\vee \neg$ b $\vee$ c), $\neg$ ($\neg$ a $\wedge$ b $\wedge \neg$ c)

d) $\neg$ (a $\wedge$ $\neg$ b $\vee$ c), $\neg$ ($\neg$ a $\vee$ b $\wedge$ $\neg$ c)

Achte auf die Prioritäten!

Hier nochmals eine kurze Zusammenfassung der wichtigsten Operatoren, wobei die höchste Priorität am Anfang und die kleinste Priorität am Ende der folgenden Liste ist:

a) Klammern: `()`

b) Indexzugriff: `x[]`

c) `**`

d) unär: `+x`, `-x`, `~x`

e) `*`, `/`, `//`, `%`

f) `+`, `-`

g) `>>` `<<`

h) `&`

i) `^`

j) `|`

k) `in`, `not in`, `<`, `<=`, `==`, `!=`, `>=`, `>`

l) `not`

m) `and`

n) `or`

10. Konstruiere boolesche Ausdrücke! Teste indem die Variablen jeweils mit Werten belegt werden. Gesucht ist jeweils ein Ausdruck, der…

    a) … `True` liefert, wenn eine Zahl `n` zwischen 1 und 10 (inkl.) liegt. In Python gibt es dafür 2 Möglichkeiten: Finde beide Varianten!

    b) … `True` liefert, wenn die Länge eines Strings `s` nicht größer als 5 beträgt.

    c) … `True` liefert, wenn die Zahl `n1` kleiner als das Doppelte von `n2` ist.

    d) … `False` liefert, wenn die der Index 3 der Sequenz `seq` von Zahlen größer als 10 aber kleiner gleich 15 ist.

    e) … `False` liefert, wenn das Doppelte von `n1` nicht größer als die Hälfte von `n2` beträgt.

11. Schreibe ein Programm `and1.py`, das eine Tabelle für das logische UND (engl. AND) ausgibt:

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

Verwende die `format`-Methode!

12. Schreibe ein Programm `or1.py`, das analog zum vorhergehenden Beispiel eine Tabelle für die logische Operation ODER (engl. OR) ausgibt.
13. Schreibe weiters ein Programm `logical1.py`, das analog zu den vorhergehenden Beispielen eine Tabelle für den logischen Ausdruck `a and b or not c` ausgibt.
14. Gegeben sind die Mengen a = {1, 2, 3, 4}, b = {3, 4, 5, 6} sowie die Menge c = {3, 4}. Berechne mit Python

    - den Durchschnitt von a und b (also a ∩ b),
    - die Vereinigung von a, b und c (also a ∪ b ∪ c),
    - die Differenz von a und c (also a \ b),
    - die symmetrische Differenz von a und b (also $a \triangle b$).

      Die symmetrische Differenz berechnet sich zu:

      (a \ b) ∪ (b \ a)

      oder zu:

      (a ∪ b) \ (a ∩ b)

    Prüfe weiterhin (wieder mit Python), ob

    - das Element 3 in a enthalten ist (also 3 ∈ a),
    - a eine Teilmenge von c ist (also $a \subseteq b$),
    - c eine Teilmenge von a ist (also $c \subseteq a$),
    - a eine echte Teilmenge von a ist (also a ⊂ a).

2 Hausübung

Kap 9 durchlesen!