Unit 5

Dr. Günter Kolousek

21. Juli 2015

Lege wiederum ein Verzeichnis an. Nennes es `05_unit5`! In diesem Verzeichnis sollen alle Dateien der jeweiligen Einheit abgelegt werden.

1 Schulübungen

1. Schreibe ein Programm `quadrate3.py`, das 6 fächerartig angeordnete Quadrate zeichnet. Verwende wieder eine Funktion. Diesmal soll diese Funktion jedoch auch noch die Füllfarbe und den Winkel als Parameter bekommen!
2. Wandle die folgenden Dezimalzahlen jeweils mit der Restmethode in das binäre Zahlensystem um: 10, 16, 18, 31, 32, 62, 64, 120, 128.
3. Wandle die folgenden Dezimalzahlen jeweils mit der Restmethode in das oktale Zahlensystem um: 1, 8, 56, 99.
4. Wandle die folgenden Dezimalzahlen jeweils mit der Restmethode in das hexadezimale Zahlensystem um: 14, 25, 32, 64, 255, 256.
5. Schreibe ein Programm `rectangles2.py`, das 2 Seiten eines Rechtecks abfragt und danach den Umfang, die Fläche und die Größenklasse ausgibt. kleines Quadrat (Fläche < 100 Einheiten), großes Quadrat (Fläche >= 100 Einheiten), kleines Rechteck (Fläche < 200 Einheiten) und großes Rechteck (Fläche >= 200 Einheiten).

   Für die Berechnung des Umfanges soll eine Funktion `perimeter`, die zwei Parameter erhält, für die Fläche eine Funktion `area` (ebenfalls zwei Parameter) und für die Größenklasse eine Funktion `class_` (ein Parameter). Die Funktionen sollen jeweils einen entsprechenden Rückgabewert liefern.
6. Schreibe ein Programm `calculate.py`, das die Berechnung von x^y für ganzzahlige y >= 0 nach folgender Vorschrift vornimmt:

   - wenn x = 1, dann ist $x^y = 1$

- wenn x = -1 und y gerade, dann ist $x^y = 1$
- wenn x = -1 und y ungerade, dann ist $x^y = -1$
- wenn y = 0, dann ist $x^y = 1$
- wenn y = 1, dann ist $x^y = x$
- wenn y > 1, dann wird x^y berechnet
- wenn y < 0 oder x und y gleich 0, dann entsprechende Meldung ausgeben

Dazu soll eine Funktion `power` entwickelt werden, die die Berechnung vornimmt.

7. Erweitere das Programm `calculate.py` um eine Funktion `sqrt` zur Berechnung der Quadratwurzel nach der folgenden Näherungsmethode:

$$x_1 = (x_0 + a/x_0)/2$$
$$x_2 = (x_1 + a/x_1)/2$$
$$x_3 = (x_2 + a/x_2)/2$$

   x_0 und a sind jeweils mit der Zahl zu initialisieren, von der die Quadratwurzel bestimmt werden soll. Bei x_3 handelt es sich um die errechnete Näherung.

   Das Programm soll zuerst die Operation abfragen, z.B. **1** bedeutet "power" und **2** bedeutet "sqrt". Dann soll die entsprechende Berechnung durchgeführt werden und das Ergebnis ausgegeben werden.

8. Erweitere das Programm `calculate.py` um die Berechnung des Minimums dreier Zahlen. Das Minimum soll durch eine Funktion `minimum` ermittelt werden.

9. Schreibe ein Modul `mymath.py`, das alle Funktionen von `calculate.py` enthält und entwickle ein Programm `calculate2.py`, das sich wie `calculate.py` verhält, aber die Funktionen aus `mymath.py` verwendet.

10. Schreibe ein Programm `grading2.py`, das eine Punktezahl (von 0 bis 100) abfragt und daraus die Beurteilung gemäß des folgenden Benotungsschlüssels ermittelt und ausgibt:

    - 0-50 Punkte: Nicht genügend
    - 51-62 Punkte: Genügend
    - 63-78 Punkte: Befriedigend
    - 79-90 Punkte: Gut
    - 91-100 Punkte: Sehr gut

Die Berechnung soll allerdings als eine Funktion `grading` im Modul `mymath` durchgeführt werden.

11. Schreibe ein Programm `hello.py`, das zuerst den Vornamen, dann den Nachnamen abfragt und daraus den gesamten Namen ermittelt und unter `name` zugreifbar macht. Der gesamte Name soll danach drei Mal in je einer Zeile ausgegeben werden.

12. Schreibe ein Programm `hello2.py`, das zuerst den Vornamen, dann den Nachnamen abfragt und daraus den gesamten Namen ermittelt. Danach soll eine Begrüßung in der Form:

    ```
    Guten Tag, Maxi Mustermann!
    ```

    ausgegeben werden. Verwende dazu den `+` Operator!

13. Schreibe ein Programm `hello3.py`, das wie `hello2.py` funktioniert, aber keinen gesamten Namen ermittelt und anstatt des `+` Operators die `format`-Methode verwendet.

14. Schreibe ein Programm `numbers.py`, das den Benutzer auffordert, eine Zahl zwischen 1 und 100 einzugeben und das entsprechende Zahlwort ausgibt!

    Tipp: Ab zwanzig setzen sich alle Zahlworte aus den Silben, ein, zwei, drei, vier, fünf, sechs, sieben, acht, neun und zwanzig, dreißig, vierzig, fünfzig, sechszig, siebzig, achtzig, neunzig zusammen.

15. Schreibe ein Programm `gruss.py`, das Texte folgender Art für Urlaubsgrußkarten erzeugt::

    ```
    __________!

    Hier __________ gefällt es mir _________.
    Gestern waren wir _________. Das war _________!

    Bis bald,

    ------------
    ```

    Dazu soll eine Funktion `gruss` geschrieben werden, die diese Felder als Parameter bekommt und den fertigen String zurückliefert.

    Das Programm soll alle Felder abfragen, danach die Funktion aufrufen und letztendlich den fertigen Kartentext ausgibt.

16. Schreibe ein Programm `homes.py`, das eine Funktion `haus(seite)` enthält, die in einem Zug ein Haus der Länge `seite` zeichnet (siehe Kapitel 5). Mittels dieser Funktion sollen 3 Häuser nebeneinander (jedes kleiner als das vorhergehende) gezeichnet werden. Verwende die Funktion `jump` aus dem Buch!

2 Hausübung

1. Wandle die folgenden Dezimalzahlen mittels der Restmethode in das Binärsystem: 5, 9, 15, 11, 33, 61, 62, 65, 130.
2. Wandle die folgenden Dezimalzahlen mittels der Restmethode in das Oktalsystem: 5, 8, 16, 64, 132.
3. Wandle die folgenden Dezimalzahlen mittels der Restmethode in das Hexadezimalsystem: 15, 16, 32, 33, 255, 1024.
4. Schreibe ein Programm `smuecheck.py`, das eine Punktezahl (0-5) einliest und einen Text folgender Gestalt in Abhängigkeit der Punktezahl ausgibt.

   ```
   _______ Punkte sind ___________!
   ```

   Das erste Feld sind die Punkte als Zahlwort und das zweite Feld ist abhängig von der Punkteanzahl: 0 → absolut zu wenig, 1 → viel zu wenig, 2 → zu wenig, 3 → gerade genug, 4 → nicht schlecht, 5 → sehr gut.
5. Kapitel 5 lesen!