Einführung in die Programmierung (Python)

Klausur für den Masterstudiengang Human Factors

11. Februar 2019

1. Das Osterfest lässt sich folgendermaßen berechnen: Man dividiert die Jahreszahl J ganzzahlig durch 4, sodass ein etwaiger Rest unberücksichtigt bleibt, und erhält die Zahl $q = J/4$. Nun berechnet man:

   $a = J \mod 19$
   $b = (M - 11a) \mod 30$
   $c = (J + q + b - D) \mod 7$

   Dann ist der (28 + b - c)te März bzw. für Zahlen über 31 der (28 + b - c - 31)te April Ostersonntag. D und M sind folgender Tabelle zu entnehmen:

   ```
   Zeitraum          D    M
   ------------------------
   1800 .. 1899     12  203
   1900 .. 2099     13  204 *
   2100 .. 2199     14  204 *
   ```

   Für die mit * gekennzeichneten Zeilen gilt: Ergibt sich b = 29 oder b = 28, so ist b um eins zu vermindern, also mit 28 bzw. 27 weiterzurechnen.

   Beispiele:

   - J = 1951. Dann sind: q = 487; a = 13; b = 1; c = 4.
     Ostersonntag ist der (28 + 1 - 4)te = 25. März.
   - J = 1954. Dann sind: q = 488; a = 16; b = 28; zu ersetzen durch b = 27; c = 6.
     Ostersonntag ist der (28 + 27 - 6 - 31)te = 18. April.

   Schreiben Sie ein Python-Programm, das die Funktion `ostersonntag` sowie einen Rahmen implementiert, der diese Funktion für die Jahre 1421, 1951, 1954, 2019 und 2401 aufruft. Dabei soll diese Ausgabe erscheinen:

   ```
   1421 --> ungültiges Jahr
   Ostersonntag 1951 --> 25.3.
   Ostersonntag 1954 --> 18.4.
   Ostersonntag 2019 --> 21.4.
   2401 --> ungültiges Jahr
   ```

   Die Funktion `ostersonntag(jahr)` übernimmt die betreffende Jahreszahl und liefert das Tupel `(Tag, Monat)` zur Angabe des Ostersonntags in dem benannten Jahr. Falls das Jahr ungültig ist, also außerhalb der in der Tabelle genannten Zeiträume liegt, wird eine Ausnahme (vorzugsweise `AssertionError` durch `assert`) geworfen, die beim Aufruf der Funktion geeignet zu behandeln ist.

2. Zur näherungsweisen Berechnung der Quadratwurzel einer reellen Zahl $a > 0$ kann das Heron-Verfahren verwendet werden. Die Iterationsgleichung lautet wie folgt:

   $$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right).$$

Der Startwert x_0 der Iteration kann, solange er nicht gleich Null ist, beliebig festgesetzt werden, die Iteration konvergiert immer. Bei negativen Startwerten konvergiert sie gegen die negative Quadratwurzel. Eine qualifizierte Schätzung für den Startwert ist $x_0 = \frac{a+1}{2}$.

Die Iteration wird beendet, wenn die Differenz zwischen x_n und x_{n+1} kleiner als ein vorgegebenes Epsilon ist, das mit 10^{-16} angenommen werden soll.

Implementieren Sie eine Python-Funktion `wurzel(a)`, die den Radikanden a übernimmt und dessen mit dem Heron-Verfahren berechnete Quadratwurzel zurückgibt.

Geben Sie ein Rahmenprogramm an, das die Radikanden aus einer Textdatei einliest (ganze oder reelle Zahlen in Python-Syntax), für jeden Radikanden die Funktion `wurzel` aufruft und die Zuordnung von Radikand und berechneter Quadratwurzel in einem Dictionary ablegt.

Die Textdatei könnte so aussehen:

```
  4 15.7 79
# 100
88 34 #101.9
117.5 11*3
R118.2
```

Pro Zeile existieren ggf. mehrere durch Leerräume getrennte Wörter, die als Radikanden zu betrachten sind. Ab dem ersten Wort, das mit einem # beginnt, wird der Rest der Zeile als Kommentar interpretiert und ignoriert. Auch Wörter, die keine gültige Zahl darstellen, werden stillschweigend ignoriert. In der obigen Datei wären folgende Radikanden zu betrachten: 4, 15.7, 79, 88, 34 und 117.5.

Am Ende soll das Programm den Inhalt des Dictionarys aufsteigend sortiert nach den Radikanden ausgeben. Zusätzlich soll auch mit der Standardfunktion `sqrt` des Moduls `math` die Quadratwurzel des jeweiligen Radikanden ermittelt und ausgegeben werden. Außerdem ist die Differenz zwischen den beiden Quadratwurzelwerten der Funktionen `wurzel` und `sqrt` auszugeben.

Hier ein Beispiel für die Ausgabe:

```
wurzel(4.0) = 2.0
sqrt(4.0) = 2.0
Delta: 0.0

wurzel(15.7) = 3.9623225512317894
sqrt(15.7) = 3.96232255123179
Delta: -4.440892098500626e-16
```

3. Palindrome sind Zeichenfolgen, die rückwärts gelesen genau denselben Text oder zumindest einen Sinn ergeben. Schreiben Sie ein Python-Programm, das eine rekursive Funktion `palindrom(s)` implementiert, die ohne Unterscheidung zwischen Groß-/Kleinschreibung für einen String ermittelt, ob er ein Palindrom darstellt. Rufen Sie diese Funktion für folgende Strings:

   ```
   '', 'a', 'Maoam', 'Auto', 'Anna', 'otto', 'arA', 'line'
   ```

   Geben Sie das Ergebnis folgendermaßen aus:

   ```
    ==> Palindrom
   a ==> Palindrom
   Maoam ==> Palindrom
   Auto ==> kein Palindrom
   Anna ==> Palindrom
   otto ==> Palindrom
   arA ==> Palindrom
   line ==> kein Palindrom
   ```

   Ersatzweise kann auch eine iterative Implementierung angegeben werden.