Übungen zu Computergrundlagen WS 2012/2013

Übungsblatt 7: Python

30. 11. 2012

Allgemeine Hinweise

- Abgabetermin für die Lösungen ist **Freitag, 7. 12., 13:30**.
- Die Abgabe sind diesmal zwei Python-Skripte mit Kommentaren, die Du im Anhang einer Email an Deinen Tutor schickst.
- Vergiss nicht, Deinen Namen als Kommentar an den Anfang der Python-Dateien zu schreiben.
- Benutze bitte *keine* spezielle Codierung, sondern reines ASCII. Das alles in Ordnung ist, merkst Du daran, dass sich Python nicht beschwert, da es bei Nicht-ASCII-Text zwingend einen PEP0263-Encoding-Marker erwartet.
- Ich möchte nochmals darauf hinweisen, dass wir für nicht lauffähige Abgaben verstärkt Punkte abziehen werden. Daher bitte *immer* ausprobieren, ob das Programm auf den Poolrechnern läuft und die gewünschte Ausgabe erzielt.

Aufgabe 7.1: Das Heronverfahren (5 Punkte)

Hat man keinen Taschenrechner zur Hand, liefert das mehr als 3000 Jahre alte Heronverfahren gute Nährungen für die Wurzel. Auf speziellen Prozessoren wie Graphikkarten wird das Verfahren sogar heute noch eingesetzt, um die Genauigkeit von Wurzelberechnungen nachträglich zu erhöhen. Das Heronverfahren besagt, dass die Folge

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

quadratisch gegen die Wurzel aus a konvergiert, d.h. mit jedem Schritt verdoppelt sich die Anzahl der signifikanten Stellen.

Das Verfahren lässt sich so verstehen: ist $x_n = \sqrt{a}$, so gilt $x_n = \frac{a}{x_n}$. Die neue Näherung wird daher als der Mittelwert aus diesen beiden Werten gebildet.

7.1.1: Beschreibe das Heronverfahren als Algorithmus. Was ist gegeben? Welche Größen brauchst Du außer a? Welche Datentypen sollten die Größen sinnvollerweise haben? Was soll das Programm ausgeben? Füge die *vollständige* Problemstellung und das Lösungsverfahren als Kommentar in das Pythonprogramm für die folgende Teilaufgabe ein. (2 Punkte)

Hinweis: Auch der Computer kann natürlich nicht unendliche viele Folgenglieder von Gleichung (1) berechnen. Du musst Dir eine Abbruchbedingung überlegen, also, wann das Programm enden soll. Überlege Dir, wann sich die neue Näherung nur noch marginal von der alten unterscheiden wird?

7.1.2: Implementiere nun das Heronverfahren in Python. (2 Punkte)

Hinweise:

- Nutze eine der in der Vorlesung gezeigten Möglichkeiten, die Parameter in das Programm zu bekommen. Am einfachsten ist es, Konstanten im Programm zu definieren. Schöner ist, die Parameter von der Kommandozeile (`sys.argv`) zu lesen.
- Du musst Dir die Folgenglieder nicht alle merken. Erinnere Dich daran, dass Variablen ihren Wert ändern können.

7.1.3: Teste nun Dein Programm für $a = 0, 2, \pi, -1$, und füge die Ausgabe dieser Tests als Kommentar am Ende Deines Programms hinzu. Aus den Ausgaben muss hervorgehen, dass das Program korrekt arbeitet bzw. Fehler ($a = -1$) abfängt! (1 Punkt)

Aufgabe 7.2: Bisektion (5 Punkte)

Ein einfaches Verfahren, um eine Nullstelle einer stetigen Funktion in einem gegebenen Intervall $[a, b]$ zu suchen, ist die Bisektion. Diese macht sich zu nutze, dass eine Nullstelle immer einen Vorzeichenwechsel bedingt. Habe also $f(x)$ einen Vorzeichenwechsel in $[a, b]$, und sei $m = (a + b)/2$ die aktuelle Intervallmitte. Unterscheiden sich das Vorzeichen von $f(m)$ und das von $f(a)$, so muss sich eine Nullstelle im Intervall $[a, m]$ befinden, andernfalls im Intervall $[m, b]$.

7.2.1: Anstatt die Bisektion selber zu implementieren, übernimmst Du den Code eines Freundes:

```
def f(x): return x**2 - 2
tolerance = 10^{-10}
a = 0
b = 2

fa = f(a)
fb = f(b)
while abs(a - b) > tolerance do:
    m = 0.5*(a + b)
    fm = f(m)
    if fm*fa < 0
      b = m
        fb = fm
    else:
        a = m
        fa = fm
print "zero at m=", m, "f(m) =", f(m)
```

Leider hat dieser Code einige Syntaxfehler. Gib eine korrigierte Version des Codes ab! Füge die Ausgabe des funktionierenden Programms als Kommentar am Ende Deines Programms hinzu. (2 Punkte)

Hinweis: Falls Du bei einem Fehler gar nicht weiterweist, dann füge die Fehlermeldung an der entsprechenden Stelle als Kommentar ins Programm ein und *erkläre*, wie Du den Fehler verstehst, und warum Du an dieser Stelle nicht weiterkommst. Kommentare der Art „Hier gibt es einen komischen Fehler, keine Ahnung, warum“ führen zu Punktabzug!

7.2.2: Der Code hat nicht nur Syntaxfehler, sondern er ist auch undokumentiert. Dokumentiere ihn ausführlich! Wozu dienen die Variablen? Was sind die Bedingungen an `f`, `tolerance`, `a` und `b`? (1 Punkt)

7.2.3: Nun teste Deine korrigierte Version mit folgenden Werten:

- `a`= 0, `b`= 2, `tolerance`= 10^{-10}
- `a`= 2, `b`= 0, `tolerance`= 10^{-10}
- `a`= 0, `b`= 1, `tolerance`= 10^{-10}
- `a`= 0, `b`= 2, `tolerance`= 0

Achte auf das Ergebnis! Einige der Tests liefern offensichtlich noch Unsinn. Welche Bedingungen für das Verfahren sind verletzt? Füge in das Programm aussagekräftige Fehlermeldungen bei ungültigen Angaben ein! (2 Punkte)

Hinweis: Es genügt, wenn das Programm in den obigen Fällen stets eine korrekte Ausgabe (bzw. eine Fehlermeldung) liefert. Weitere Nebenbedingungen, wie korrekte Datentypen oder die Signatur von `f`, müssen nicht überprüft werden.