Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Dr. Felix Schindler

27. September 2017

Übung zum Kompaktkurs

Einführung in die Programmierung zur Numerik mit Python

Sommersemester 2017 — Blatt 3

Aufgabe 1 (Bruchrechnung)

Schreiben Sie ein Programm, welches zu zwei Brüchen $\frac{a_1}{b_1}$, $\frac{a_2}{b_2}$ mit vom Benutzer über die Kommandozeile angegebenen a_1, a_2, b_1, b_2 die Summe $\frac{a_1}{b_1} + \frac{a_2}{b_2}$ sowie das Produkt $\frac{a_1}{b_1} \cdot \frac{a_2}{b_2}$ berechnet und als gekürzte Brüche im Terminal ausgibt.

Hinweise:
- *Modularisieren Sie ihr Programm, indem Sie Funktionen* `ggt(a,b)`, `kgv(a,b)`, `erweitern(x, k)` *sowie* `kuerzen(x, k)` *schreiben.*
- *Stellen Sie Brüche in Ihrem Programm als Python-Liste mit zwei Elementen dar.*

Aufgabe 2 (Regula-falsi-Verfahren)

Schreiben Sie eine Funktion `regula_falsi(f,a,b,tol)`, welche eine Nullstelle der Funktion `f` mit Hilfe des Regula-falsi-Verfahrens bestimmt, wobei `a` und `b` der linke und rechte Startwert des Verfahrens sind, sowie `tol` die Toleranz für den Abbruch des Verfahrens.

Mit $a_0 := a$, $b_0 := b$, sodass $f(a_0) \cdot f(b_0) < 0$, ist dabei das Regula-falsi-Verfahren gegeben durch die Iterationsvorschrift:

$$
\begin{aligned}
c_{n+1} &:= a_n - \frac{b_n - a_n}{f(b_n) - f(a_n)} \cdot f(a_n), \\
a_{n+1} &:= \begin{cases} c_{n+1} & f(a_n) \cdot f(c_{n+1}) > 0 \\ a_n & f(a_n) \cdot f(c_{n+1}) < 0 \end{cases}, \\
b_{n+1} &:= \begin{cases} c_{n+1} & f(b_n) \cdot f(c_{n+1}) > 0 \\ b_n & f(b_n) \cdot f(c_{n+1}) < 0 \end{cases}.
\end{aligned}
$$

Das Verfahren wird abgebrochen, sobald $|f(c_{n+1})| < tol$ gilt. c_{n+1} ist dann die gesuchte Approximation einer Nullstelle von f.

Testen Sie ihre Funktion, indem Sie diese mit verschiedenen von Ihnen definierten Funktionen `f`, Startwerten und Toleranzen aufrufen.

Hinweis: Die Python-Funktion `abs(x)` *berechnet den Absolutbetrag von* `x`*.*

Aufgabe 3 (Fibonacci-Zahlen (II))

(a) Schreiben Sie Funktionen `fib_loop(n)` und `fib_rec(n)`, welche die n-te Fibonacci-Zahl mit Hilfe einer Schleife bzw. mit Hilfe der Rekursionsformel berechnen. Schreiben Sie ein Hauptprogramm, welches für ein über die Kommandozeile übergebenes n beide Funktionen aufruft und das jeweilige Ergebnis auf dem Terminal ausgibt.

(b) Nutzen Sie die eingebaute Python-Hilfe, um im Modul `time` der Standardbibliothek eine Funktion zu finden, mit deren Hilfe sie die Laufzeit der Funktionsaufrufe in Sekunden bestimmen können.

Geben Sie die Laufzeiten auf dem Terminal aus.

(c) Schreiben Sie eine weitere Funktion `fib_dict(n)`, die ebenfalls rekursiv die n-te Fibonacci-Zahl berechnet, alle Zwischenergebnisse jedoch in einem global definierten Dictionary zwischenspeichert. Falls sich das Ergebnis bereits in dem Dictionary befindet, soll dieses anstelle der Rekursionsvorschrift verwendet werden.

Vergleichen Sie die Laufzeit von `fib_dict(n)` mit den Laufzeiten von `fib_loop(n)` und `fib_rec(n)`.