Institut für Informatik
Dr. Tobias Neckel
Christoph Kowitz
Alfredo Parra

Technische Universität München
WS 2015/2016

Einführung in die wissenschaftliche Programmierung

Übungsblatt 4

1.) Telefonbuch (Hands-on)

Implementieren Sie in Python ein interaktives Telefonbuch mit den Funktionen "neuer Eintrag", "Eintrag suchen" und "Alles anzeigen". Verwenden Sie eine geeignete Datenstruktur, um die Einträge (d.h. Name und Nummer) zu speichern. Wenn man das Python-Programm startet, soll ein (einfaches, text-basiertes) Menü erscheinen, z.B.

```
(1) new entry
(2) find entry
(3) list entries
(4) quit
```

Je nach Auswahl des Benutzers, soll dann die gewünschte Funktion durchgeführt werden, d.h.,

- `new entry`: erwartet zunächst die Eingabe eines Namens und anschl. die Eingabe der Telefonnummer. Ist eine der Eingaben fehlerhaft, springt das Programm ins Hauptmenü zurück und gibt eine knappe Fehlermeldung „`Wrong name / phone number`“ aus.
- `find entry`: sucht nach einem Namen im Telefonbuch.
- `list entries`: gibt Namen und zugehörige Telefonnummern aus.
- `quit`: beendet die Telefonbuch-Applikation.

Optional: Achten Sie im Menü auf ungültige Eingaben und reagieren Sie entsprechend darauf.

2.) Numerische Quadratur

Die numerische Quadratur bezeichnet die näherungsweise Berechnung von bestimmten Integralen, d.h. Integralen der Form

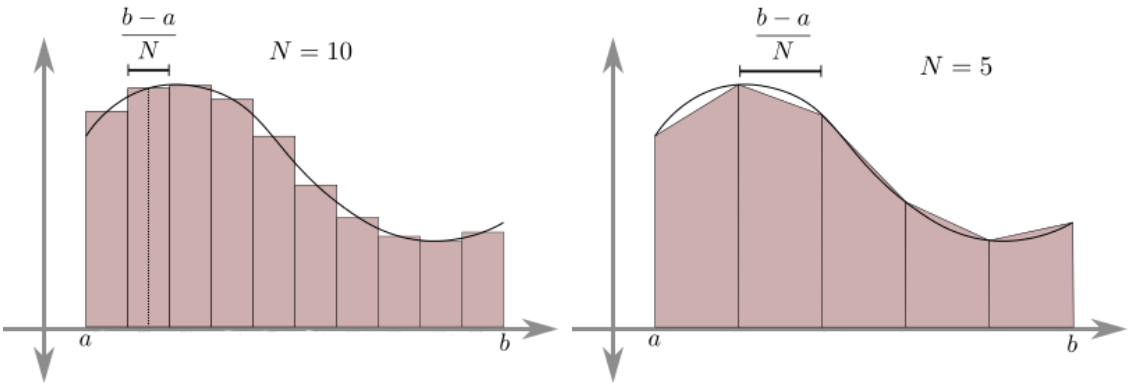
$$I(f) = \int_a^b f(x)\,\mathrm{dx}\,.$$

Viele Quadraturregeln lassen sich als gewichtete Summe

$$I(f) \approx I_Q(f) = \sum_{i=1}^{n} w_i f(x_i) \tag{1}$$

schreiben. Zwei bekannte Quadraturregeln dieser Form sind die zusammengesetzte Rechtecksregel I_R und die zusammengesetzte Trapezregel I_T

$$I_R(f) = \frac{b-a}{n} \sum_{i=0}^{n-1} f\left(a + \left(i + \frac{1}{2}\right)\frac{b-a}{n}\right),$$
$$I_T(f) = \frac{b-a}{n-1}\left(\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-2} f\left(a + i\frac{b-a}{n-1}\right)\right).$$



(a) Zusammengesetzte Rechtecksregel $I_R(f)$ (b) Zusammengesetzte Trapezregel $I_T(f)$

Implementieren Sie zwei Funktionen `Midpoint(f,a,b,n)` und `Trapezoidal(f,a,b,n)`, welche die Summen $I_R(f)$ und $I_T(f)$ berechnen. Beide Funktionen sollen in einem

Bilder: http://calculus.seas.upenn.edu/?n=Main.DiscreteIntegration

Modul `Integrals.py` implementiert werden. Definieren Sie die Standardwerte `a=0`, `b=1`, und `n=100` für beide Funktionen.
Verwenden Sie `Integrals.py` als Modul. Schreiben Sie ein Skript `test.py`, das das Modul `Integrals.py` importiert, und testen Sie sie mit einer Sinusfunktion `f = math.sin` und einem Polynom `f = lambda x: polyval(c,x)` mit `c = [2.0,0,0.1]`.

Hausaufgabe

Untersuchen Sie die Genauigkeit der beiden Integrationsverfahren. Betrachten Sie hierzu die Integration von $\sin(x)$ mit anwachsender Anzahl von Stützstellen 2^N. Was passiert mit dem Fehler für beide Verfahren, wenn die Anzahl der Stützstellen verdoppelt wird?

3.) Ratespiel (Hausaufgabe)

Schreiben Sie ein Programm, mit welchem der Benutzer eine Zufallszahl erraten kann. Dies soll so ablaufen: Das Programm erzeugt eine Zufallszahl (z.B. natürliche Zahlen zwischen 0 und 1000) und der Benutzer kann seine Vermutung eingeben. Das Programm gibt dann entweder "zu groß", "zu klein" oder "richtig" zurück. Der Benutzer darf so lange raten, bis er die Zahl erraten hat. Verwenden Sie z.B. die `raw_input()` Funktion.