MATHEMATISCHES INSTITUT
PROF. DR. CHRISTIANE HELZEL
ANDREAS TROLL

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

2.11.2017

Computergestützte Mathematik zur linearen Algebra – 4. Übungsblatt

Vergessen Sie nicht Ihren Quelltext sinnvoll zu kommentieren!

Aufgabe 9: (for-Schleife)

Befehle: `input`, `int()`, `for`, `if`, `range`
Erstellen Sie ein PYTHON-Skript *meine_erste_schleife.py*:

(a) Erstellen Sie eine Funktion `summe()`, die über die Eingabekonsole eine ganzzahlige Zahl N einliest und mit Hilfe einer `for`-Schleife die Summe $\sum_{n=1}^{N} n^2$ berechnet und zurückgibt.

(b) Testen Sie Ihre Funktion mit den Zahlen 2, 9, -2 und 23

Aufgabe 10: (for-Schleifen und if-Abfragen)

Befehle: `for`, `if`, `elif`, `else`, `continue`
Erstellen Sie ein PYTHON-Skript mit dem Namen *zahlen_analyse.py*:

(a) Schreiben Sie eine Funktion `analyse(eingabe)`, die eine Liste oder ein Tupel `eingabe` übergeben bekommt und mittels einer `for`-Schleife jedes Element folgendermaßen „analysiert“:

- Als erstes soll ein Text ausgegeben werden der zeigt, welche Zahl analysiert wird.
- Sollte das Element 0 sein, so wird es nicht weiter analysiert (was ausgegeben wird) und die Schleife geht sofort zum nächsten Element (gucken Sie sich dafür den Befehl `continue` an).
- Dann soll ausgegeben werden ob das Element positiv oder negativ ist.
- Danach soll geprüft werden, ob die Zahl durch 2 teilbar ist, was auch ausgegeben wird.
- Für die Übersichtlichkeit soll am Ende der Analyse eine neue Zeile ausgegeben werden.

Für `analyse((1,0,-2))` könnte die Ausgabe beispielsweise so aussehen:

```
1 wird analysiert:
1 ist positiv
1 ist nicht durch 2 teilbar

0 wird analysiert:
0 wird in der Analyse ignoriert

-2 wird analysiert:
-2 ist negativ
-2 ist durch 2 teilbar
```

WICHTIG: Verwenden Sie sinnvoll alle Befehle die oben stehen (auch `elif`), aber nicht die Funktion `range`.

(b) Testen Sie `analyse` mit `a=(2,-1,0,6.8,-0.59)`.

Aufgabe 13: (Slicing)
Skriptname *slicing.py*
Erstellen Sie folgendes Objekt des Typs `list`:
`a=[3,1,4,1,5,9,2,6,5,3,5]`
Greifen Sie nun mit Hilfe von *Slices* auf die folgenden Elemente zu. Ihre Zugriffe sollen so allgemein sein, dass diese auch bei Listen anderer Länge funktionieren.

(a) Das dritte bis achte Element.

(b) Die Elemente mit ungeradem Index.

(c) Das erste, vierte, siebte und zehnte Element.

(d) Alle Elemente ab dem zweitem.

(e) Das elfte bis fünfte Element (also rückwärts).

(f) Das vor-, viert-, sechst- und achtletzte Element.

Aufgabe 14: (*Arrays*)

Befehle: `array`, `cumsum`, `reshape`, `random` (alle aus dem `numpy`-Modul), `range`
Skriptname *mvmultiplikation.py*

(a) Erstellen Sie ein `array a` mit den Zahlen 1 bis 5. Verwenden Sie dabei den Befehl `range` oder `arange`.

(b) Erstellen Sie nun mit dem Befehl `ones` ein `array b` der Dimension 5×1 gefüllt mit Dreien.

(c) Erstellen Sie außerdem ein `array C` der Dimension 5×5 mit zufälligen natürlichen Zahlen aus $[0, 2]$ mit Hilfe des `np`-Untermoduls `random` (siehe Blatt 1).

Überlegen Sie sich welche Ausgaben folgende Befehle erzeugen und überprüfen Sie Ihre Überlegungen. Welche sind mathematisch sinnvoll? Wieso erzeugen einige Eingaben eine Fehlermeldung?

(d) `C.size`
(e) `a.shape`
(f) `b.shape`
(g) `7 * a`
(h) `a + b`
(i) `a + C`
(j) `a * C`
(k) `b * C`
(l) `C * C`
(m) `a * b`
(n) `C @ a`
(o) `C @ b`
(p) `C @ C`
(q) `b @ C`
(r) `np.cumsum(a,1)`
(s) `np.reshape(a,(1,5))`

Besprechung in den Übungen vom 6.-10. November 2017.