Einführungsblatt

2. Februar 2023

Das Übungsblatt stellt ein paar Aufgaben für die Einarbeitung in python im Selbststudium. Es ist sehr empfehlenswert diese Aufgaben schon *vor* Semesterstart zu bearbeiten.

Im empfohlenen Buch Python Programming And Numerical Methods: A Guide For Engineers And Scientists finden Sie jeweilen am Ende des Kapitels Aufgaben, an denen Sie Ihr Wissen sehr gut testen können. Zum Einstieg sind die Grundlagenkapitel 1-5, optional Kapitel 6 empfehlenswert. Es müssen nicht alle Aufgaben lückenlos gelöst werden. Sie merken selber, wieviel Übung notwendig ist.

Aufgabe 1

Lösen Sie die Aufgaben aus dem Buch:

a) Chapter 1: Python Basics

b) Chapter 2: Variables and Basic Data Structures

c) Chapter 3: Functions

d) Chapter 4: Branching Statements

e) Chapter 5: Iteration

f) (optional) Chapter 6: Recursion

Im Folgenden noch ein paar eigene Aufgaben.

Aufgabe 2

Stellen Sie die folgenden Funktionen graphisch *geeignet* dar:

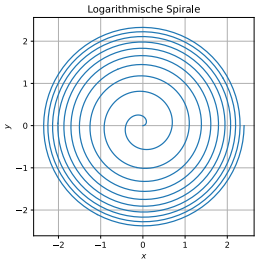
a) $x(t) = \log(t)$ für $t \in (0, 2]$

b) $x(t) = \cos(2\pi t)$ für $t \in [0, 2]$, $y(t) = \sin(2\pi t)$ für $t \in [0, 2]$

Stellen Sie die beiden Graphen im gleichen Plot dar. Beschriften Sie die Graphen mit einem Label und einer Legende, ebenso die x/y-Achsen.

c) $\boldsymbol{x}(t) = \log(1 + t) \cdot \begin{pmatrix} \cos(2\pi t) \\ \sin(2\pi t) \end{pmatrix}$ für $t \in [0, 10]$

Der Graph soll in x und y Richtung gleich skaliert dargestellt werden.



Beschriften Sie die Achsen und geben Sie dem Graph einen Titel.

Aufgabe 3

Die Aufgabe greift das Erstellen von Listen und daraus Arrays auf.

a) Erstellen Sie eine List mit den folgenden Zahlen: $0, 1, 1, 2, 3, 5, 8, 13, \ldots$

Beschreiben Sie die Folge und rechnen Sie 1000 Elemente aus. Stellen Sie diese graphisch dar.

Vorgehen 1: Erstellen Sie eine wachsende Liste und daraus ein Numpy Array.

Vorgehen 2: Erstellen Sie ein Numpy Array und speichern Sie die Glieder in diesem. Definieren Sie den Datentyp auf Integer.

Welchen Datentyp hat das Array, wenn Sie

(a) das Array mit 0.5 multiplizieren,

(b) das Array mit 5 multiplizieren,

(c) das Array mit 5. multiplizieren?

Bemerkung: Der Datentyp von Numpy Array's müssen wir gut im Auge behalten, er kann die Ursache von schwer zu lokalisierenenden Fehler sein!

b) Implementieren Sie die Funktion

$$f : [-2, 2] \to \mathbb{R}$$
$$x \mapsto f(x) = \begin{cases} 1 & x < 0 \\ 1 - x & 0 \leq x < 1 \\ 0 & 1 \leq x \end{cases}$$

so, dass diese für Numpy Array's ausführbar ist:

```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    <<<snipp, selber machen>>>
    return y

xi = np.linspace(-2,2,400)
plt.plot(xi,f(xi))
```

Aufgabe 4

a) Berechnen Sie $A \cdot b$ mit Hilfe von Numpy Array's, wobei

$$A = \left(i \cdot j\right)_{\substack{i=1,\ldots,10 \\ j=1,\ldots,10}} \quad \text{und}$$
$$b = \left(i^2\right)_{i=1,\ldots,10}.$$

Benutzen Sie dazu **nicht** die unten dargestellte minimalistische Lösung, sondern studieren Sie diese, **nach dem** Sie das Problem selber gelöst haben.

```
import numpy as np
b = np.arange(1,11)
A = np.outer(b,b)
b *= b
print(A@b)
```

b) Mit `np.diag` erhält man die Diagonale einer Matrix bzw. kann eine Diagonalmatrix erstellt werden.

   (a) Was erhalten Sie mit `np.diag(A)`

   (b) Berechnen Sie `np.diag(b)`

   mit `A` und `b` von a).

c) Ändern Sie die Diagnonale der Matrix A auf

$$A_{i,i} = i.$$

Aufgabe 5

a) Erstellen Sie eine Liste mit ± 1 Einträge (10 Paare).

b) Implementieren Sie das Sieb des Erastones (Quelle: wikipedia):

```
algorithm Sieve of Eratosthenes is
    input: an integer n > 1.
    output: all prime numbers from 2 through n.

    let A be an array of Boolean values, indexed by integers 2 to n,
    initially all set to true.

    for i = 2, 3, 4, ..., not exceeding sqrt(n) do
        if A[i] is true
            for j = i^2, i^2+i, i^2+2i, i^2+3i, ..., not exceeding n do
                set A[j] := false

    return all i such that A[i] is true.
```

Aufgabe 6

Implementieren Sie eine Funktion zur Berechnung der Fakultät $n!$

a) mit Hilfe einer `for`-Schleife

b) mit Hilfe eines geeigneten Numpy Array's.

c) rekursiv, dh. die Funktion ruft sich selber auf.

Testen Sie die drei Methoden auf Geschwindigkeit im Vergleich zur Scipy Implementierung `scipy.special.factorial`. In einem jupyter-notebook oder in der `ipython` Konsole können Sie dies mit Hilfe des Magic Commands `> %timeit f(100)` durchführen. Um wieviel langsamer ist Ihre schnellste eigene Implementierung? (M2 Macbook Air Faktor 4 langsamer).

Randnotiz: Unter Verwendung des *just in time compilers* (`numba`) ist die erste Implementierung sogar ein Faktor 3.3 **schneller**.

Aufgabe 7

In Python können Aufgaben sehr kompakt implementiert werden. Studieren Sie folgende Code Zeilen.

1. Was liefert der Ausdruck:

```
[i if i%2==0 else -i for i in range(10)]
```

2. Was berechnet die inline Funktion `f(n)`:

```
f = lambda n: n*f(n-1) if n>1 else n
```

3. Was berechnet die Funktion `g(n)`:

```
def g(n):
  A = np.ones(n-1,dtype = bool)
  i=2
  while i < np.sqrt(n):
    if A[i]:
      A[[i**2+k*i-2 for k in range(int((n-i**2)/i)+1)]]=False
    i+=1
  return np.arange(2,n+1)[A]
```

Aufgabe 8

Knacknuss Aufgabe: Gesucht ist die Zugfolge mit einem Springer auf einem Schachbrett, so dass jedes Feld **genau einmal** vor kommt. Das Startfeld kann beliebig definiert werden.

Erweiterung: gesucht sind alle Lösungen.