# 1 Berechnung von Summen (ca. 5 + 4 + 1 = 10 Punkte)

Gegeben sind natürliche Zahlen in einer Liste `n`, also z.B.

```
n = [1, 8, 5, 2]
```

(a) Schreiben Sie ein kurzes Stück Python–Code, das die Summe der Quadrate der Zahlen in der Liste `n` aufaddiert, in der Variable `a` speichert und ausgibt. Für das Beispiel oben, würde sich $1^2+8^2+5^2+2^2 = 94$ ergeben.

```
a = 0
for i in range(len(n)):
    a = a + n[i]**2
print a
```

(b) Gegeben ist wieder die Liste `n`. Addieren Sie nun die Zahlen in der Liste und quadrieren Sie anschliessend das Ergebnis. Speichern Sie das Ergebnis in der Variable `b`. *Geben Sie diesmal die gesamte Rechnung mit allen Summanden aus.* Für das Beispiel oben, würde sich folgende Ausgabe ergeben:

```
( 1 + 8 + 5 + 2 )^2 = 256
```

```
print "(",
for i in range(len(n)):
    print n[i],
    if(i < len(n) - 1):
        print "+",
b = sum(n)**2
print ")^2 =", b
```

(c) Subtrahieren Sie nun `b` (Teilaufgabe (b)) von `a` (Teilaufgabe (a)) und geben Sie wieder nicht nur das Ergebnis, sondern auch die einzelnen Terme aus. Verwenden Sie dafür nur genau eine Zeile Code! Im Beispiel ergibt sich die Ausgabe

```
94 - 256 = -162
```

```
print a, "-", b, "=", a - b
```

## 2 Liste von Listen (ca. 11 Punkte)

Der Inhalt eine Liste von Listen soll aus einer Text–Datei gelesen werden. Die Datei enthält dabei entweder eine Zahl oder das Zeichen # am Anfang jeder Zeile.
Zum Beispiel:

```
10
20
#
30
#
40
50
60
```

Schreiben Sie ein Python Programm, das eine Text–Datei mit dem Namen `daten.txt` öffnet, diese dann zeilenweise einliest, dabei die durch das Zeichen # getrennten Folgen von Zahlen jeweils in einer einfachen Liste speichert und diese Liste schließlich der aufzubauenden Liste von Listen als Element hinzugefügt. Achten sie darauf die eingelesen Daten als Integer und nicht als String in den Listen zu speichern. Geben Sie am Ende ihres Programms die von ihnen aufgebaute Liste von Listen aus.
Wenn die Datei `daten.txt` z.B. den obigen Inhalt hat, so soll die Ausgabe Ihres Programms etwa wie folgt aussehen: `[[10, 20], [30], [40, 50, 60]]`

```
lol = list()
l = list()
file = open("data.txt","r")
for line in file:
    if line == '#':
        lol.append(l)
        l = list()
    else:
        l.append(int(line))
file.close()
if len(l) > 0:
    lol.append(l)
print lol
```

# 3 BluRay Sammlung (ca. 3 + 2 + 2 + 1 = 8 Punkte)

Sie besitzen eine umfangreiche BluRay Sammlung und verleihen davon auch immer wieder Titel an Ihre Freundinnen und Freunde. Um nicht den Überblick zu verlieren, wollen Sie die Verwaltung der Sammlung ab jetzt von einem Pythonprogramm übernehmen lassen.

Im Folgenden sollen Sie Schritt für Schritt eine Klasse `BluR` erstellen, welche *eine* BluRay Disc modeliert. Eine BluRay Disc hat einen Namen und ein Variable welche angibt, ob Sie gerade verliehen ist oder nicht.

(a) Definieren Sie eine Klasse `BluR` mit einem Konstruktor dem man den Namen der BluRay übergibt und der diese Namen in einer Klassenvariable `name` speichert. Zusätzlich soll die Klassenvariable `verliehen` auf `False` gesetzt werden, d.h. die Disc ist nicht ausgeliehen.

```
class BluR:
    def __init__(self, name):
        self.name = name
        self.verliehen = False
```

(b) Schreiben Sie für die Klasse `BluR` eine Methode `istVerliehen` welche `True` zurückgibt, falls die BluRay verliehen ist und ansonsten `False`

```
def istVerliehen(self):
   return self.verliehen
```

(c) Erweitern Sie die Klasse `BluR` um die Methode `ausleihen`. Sie wird aufgerufen, wenn Sie eine BluRay Disc verleihen wollen. Ist Sie bereits verliehen, soll eine Fehlermeldung ausgegeben werden. Ansonsten setzen Sie die Klassenvariable `verliehen` entsprechend.

```
def ausleihen(self):
    if(self.verliehen != False):
        print "Bereits an",
          self.verliehen, "verliehen"
        return False
    self.verliehen = True
    return self.verliehen
```

(d) Jetzt fehlt noch die Methode `zurueck` der Klasse `BluR` welche aufgerufen wird, nachdem eine Disc zurückgebracht wurde. Sie soll die entsprechenden Klassenvariablen geeignet zurücksetzen.

```
def zurueck(self):
    self.verliehen = False
    return True
```

# 4 Wärmeleitung (ca. 1 + 4 + 2 + 2 + 4 = 13 Punkte)

Gegeben sei die eindimensionale Wärmeleitungsgleichung

$$\frac{\partial^2}{\partial x^2} T(x) = 0\,, \qquad x \in \Omega := [0;1] \tag{1}$$

$$T(x) = g(x)\,, \qquad x \in \partial\Omega \tag{2}$$

Der zur eindimensionalen Diskretisierung zugehörige Finite-Differenzen-Stern lautet $[1 \;\; -2 \;\; 1]$.

a) Wie lauten die Befehle, um die Module `numpy` und `sys` einzubinden?

```
from numpy import *
import sys
```

b) Schreiben Sie eine Funktion `computeHeatMatrix1D`, die den Parameter `n` benutzt, um die zugehörige Matrix $A$,

$$A := \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & & \vdots \\ 0 & 1 & -2 & \ddots & 0 \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \dots & 0 & 1 & -2 \end{pmatrix}, \qquad A \in \mathrm{R}^{n \times n}\,,$$

des linearen Gleichungssystems, das durch Verwendung von $n$ Freiheitsgraden im Inneren von $\Omega$ entsteht, mithilfe des Moduls `numpy` als `numpy` Array zu erstellen und zurückzugeben.

```
def computeHeatMatrix1D(n):
  a = identity(n)
  a = a*-2
  for i in range(n-1):
    a[i, i+1] = 1
    a[i+1, i] = 1
  return a
```

c) Formulieren Sie eine Funktion `computeHeatRHS`, die für den Parameter `n` die zugehörige rechte Seite des Gleichungssystems als `numpy` Array berechnet und zurückgibt. Benutzen Sie hierbei für $g(x)$ in (2) folgende Funktion:

$$g(x) \quad = \quad \begin{cases} 100, \text{ für } x = 0 \\ -100, \text{ für } x = 1 \\ 0, \text{ sonst} \end{cases}$$

```
def computeHeatRHS(n):
  b = zeros(n)
  b[0]   = -100
  b[n-1] = 100
  return b
```

d) Implementieren Sie eine Funktion `computeNorms`, die für einen Vektor `x` (`numpy` 1D array) der Länge $n$ unter Verwendung des Skalarprodukts von `numpy` die folgenden beiden Normen berechnet und beide Ergebniswerte zurückgibt:

Maximumsnorm: $||x||_\infty := \max_i |x_i|$

Euklidische Norm: $||x||_2 := \sqrt{\sum_{i=1}^{n} x_i^2}$

```
def computeNorms(x):
  norm_L2  = sqrt(dot(x,x))
  norm_max = max(abs(x))
  return norm_L2, norm_max
```

e) Schreiben Sie ein Codestück innerhalb eines python Skriptes, das einen Fehler ausgibt und das Programm stoppt, sofern kein Inputargument `n` auf der Kommandozeile beim Aufruf des Skriptes mit übergeben wurde. Lesen Sie `n` in eine lokale Variable ein, die Sie zur Erzeugung einer Matrix $A$ und einer rechten Seite $b$ der Dimension $n$ benutzen. Berechnen Sie die Lösung des linearen Gleichungssystems $Ax = b$ unter Zuhilfenahme einer passenden Funktion des `numpy` Moduls `linalg`.

```
from numpy import *
import sys

if len(sys.argv)!=2:
  print "error: missing input argument n (int)!"
  exit(1)

n = int(sys.argv[1])
A = computeHeatMatrix1D(n)
b = computeHeatRHS(n)
x = linalg.solve(A,b)
print "solution is:"
print x
```