Einführung in Python - Felix Köster

October 9, 2018

Warum Python?

- Einfache Syntax
- keine Variablentypendeklaration
- Skriptsprache $\rightarrow$ Wird nicht kompiliert
- Viele Bibliotheken und riesige Community, die sich um eine Verbesserung und Optimierung kümmert
- Open Source
- Sehr schön zur Visualisierung (Plotten der Daten) geeignet
- Filmempfehlung, alle Filme von Monty Python

Python vs C/C++ (Hello World)

- Vergleich "Hello World":
- Python:

```python
print("Hello, world!")
```

- C++:

```cpp
#include <iostream>

int main()
{
        std::cout << "Hello, world!" << std::endl;
        return 0;
}
```

- Python ein guter Einstieg in die Programmierwelt
- Häufig wird C/C++ in Arbeitsgruppen verwendet, wenn die Simulationen größer werden
- In diesem Kurs jedoch reicht Python ohne Probleme aus, um alle Aufgaben zu bearbeiten

Erstes richtiges Beispiel

- Beispiel zum ausrechnen eines Funktionswertes:

```
import math
x = 0.5
y = x*x + math.sin(x)
print("x*x + sin(x) = " + str(y))
```

- Ausgabe: "$x * x + \sin(x) = 0.729425538604$"

Erläutern der einzelnen Zeilen

- Module importieren:

```
#Dies ist ein Kommentar und wird vom Interpreter ignoriert

import math #Hier wird ein Modul importiert
#auf dessen Funktionen mit "math." zugegriffen werden kann

import numpy as np #Hier wird das Modul "Numpy" importiert
#auf dessen Funktionen mit "np." zugegriffen werden kann

from matplotlib import * #Hier wird das Modul "Matplotlib" importiert
#auf dessen Funktionen ohne einen prefix zugegriffen werden kann
```

- Variablen deklarieren und Wert zuweisen:

```
x = 0.5 #Hier wird die Variable mit dem Namen "x" deklaiert
#und ihr der Wert 0.5 zugewiesen
```

Erläutern der einzelnen Zeilen

- Arithmetik und Funktionen nutzen:

```
y = x*x + math.sin(x) #Hier wird die Variable "y" deklariert
#und ihr der Wert von x*x + sin(x) zugewiesen
```

- Ausgabe in die Konsole:

```
print("x*x + sin(x) = " + str(y)) #Hier werden erst der String
#"x*x + sin(x) = "und der Wert von y zu einem String zusammengefügt
#und dann in der Konsole ausgegeben
```

Listen

- Listen sind eine Ansammlung von Größen unter einem Variablennamen:

```
a = [3, "b", 4.7, "Blub"] #Eine Liste mit 4 Eintraegen
print(a[2])          #Gibt Eintrag Nummer 3, also 4.7, wieder!
#Zaehlung faengt bei Null an
```

- Listen können Listen enthalten und dynamisch vergrößert werden:

```
a = [9.05, 5.1, 3.5] #a ist eine List mit 3 Eintraegen
a.append(7.1) #Fuegt zu Liste a 7.1 hinzu
b = [a, "Bliblub", "9"] #b ist eine Liste mit 3 Eintraegen,
#wobei Eintrag 1 aus einer Liste besteht
print(b[0][3]) # Gibt 7.1 zurueck
```

Schleifen

- Schleifen wiederholen Befehle bis bzw. solange eine Bedingung erfüllt ist:

```
x = 0 # x wird deklariert und der Wert 0 zugeordnet
dx = 0.1 # Schrittweite von x fuer jeden Schleifendurchlauf
while x < 10:    # Die Schleife laeuft solange x kleiner 10 ist
        print(x*x)  # Es wird pro Schritt der Wert von x*x ausgegeben
        x = x + dx  # Dann wird x um dx erhöht
```

Plotten

```
import matplotlib.pyplot as plt
# Importiere das pyplot modul von matplotlib zum plotten

x = 0 # x wird deklariert und der Wert 0 zugeordnet
dx = 0.1 # Schrittweite von x fuer jeden Schleifenschritt
listx = [] # Eine leere Liste, welche die x-Werte speichern soll
listy = [] # Eine leere Liste, welche die y-Werte speichern soll
while x < 10: # Die Schleife laeuft solange x kleiner 10 ist
        listx.append(x) #Fuege Liste x den Wert von x zu
        listy.append(x*x) # Fuege Liste y den Wert von x*x zu
        x = x + dx # Erhoehe x um den Wert dx

#Plottet Liste y ueber Liste x
plt.plot(listx, listy)

#Setzt die Achsenbeschriftung
plt.xlabel("x")
plt.ylabel("x*x")
plt.show() #Zeigt den Plot
```

Ordinary differential equation (ODE) loesen

- Beispiel ODE:

$$\dot{x} = -\gamma x - \omega y \tag{1}$$
$$\dot{y} = \omega x - \gamma y \tag{2}$$

- Kann diskretisiert umgeschrieben werden als:

$$dx = (-\gamma x - \omega y)dt \tag{3}$$
$$dy = (\omega x - \gamma y)dt \tag{4}$$

- Hier geben die Klammerterme sozusagen die zeitliche Änderung von x und y, also $\dot{x}$ und $\dot{y}$ am Phasenpunkt x,y an.

$$(-\gamma x - \omega y) \tag{5}$$
$$(\omega x - \gamma y) \tag{6}$$

- Multipliziert mit einem klein gewählten dt approximiert man somit die Änderung von x und y mit den Werten dx und dy.

- Das ganze muss jetzt in Code umgewandelt werden. Somit kann die ODE stückweise integriert und somit der Verlauf numerisch gefunden werden.