Einführung in Python

Christian Wilms

Computer Vision Group
Universität Hamburg

Sommersemester 2020

20. April 2020

Gliederung

Übersicht

Motivation

Warum Python?

- sehr flexible Sprache: Skripte, große Klassensysteme
- leicht zu schreiben, da kein write/compile/re-compile Kreislauf
- interaktiver Modus animiert zum einfachen Ausprobieren
- Quelltext lässt sich sehr kompakt und lesbar schreiben
- frei verfügbar für viele Plattformen

By the way, the language is named after the BBC show "Monty Python's Flying Circus" and has nothing to do with reptiles.

The Python Tutorial

Python

- interpretierte Sprache (langsamer)
- interaktiver Modus vs. Texteditor
- Pythonskripte sind `.py`-Dateien
- dynamisches Typsystem
- automatisches Speichermanagement
- einfache Einbindung von (schnellem) C/C++-Code

Wie bekomme ich Python?

Notwendig

Python 3.x `www.python.org`

Spyder `pythonhosted.org/spyder`
gute IDE für Python

Praktisch

Anaconda `conda.io`
Python-Distribution, die (fast) alles Wichtige enthält

conda `conda.io`
Paketmanager von Anaconda, zum einfachen Installieren von Paketen

Übersicht

Wahrheitswerte

Literale für Falsch `False`, `0`

Literale für Wahr `True`, `1`

Operatoren für 'Booleans' `not`, `and`, `or`, mathematische Operatoren

```
>>> True and False
False
>>> not True
False
>>> False == 0
True
```

Zahlen

Datentypen für Zahlen `int`, `float`, `complex`, ...

Operatoren für Zahlen `+`, `-`, `*`, `/`, `==`, `<=`, `<`, ...

Das `math`-Modul bietet viele weitere Funktionen.

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5.0*6) / 4
5.0
>>> 5**2  # 5 squared
25
>>> float(5)
5.0
```

Strings

- nicht veränderbare Zeichenketten
- Literale mit einfachen oder doppelten Anführungszeichen
- Umwandlung von Zahlen zu Strings mit `str()`

```
>>> 'hello world' == "hello world"
True
>>> 'hello ' + 'world'
'hello world'
>>> 'hello ' + 'world ' + str(2)
'hello world 2'
>>> int('123')
123
```

Übersicht

Listen - Erzeugung I

- einfach zu erzeugender Container, der **sehr** oft genutzt wird
- Listen haben keinen Elementtyp, es können Äpfel und Birnen in einer Liste sein

```
>>> []
[]
>>> list()
[]
>>> l = [0,2,4]
>>> l.remove(4) #entfernt Element 4
>>> l
[0,2]
>>> l.append(5) #fügt Element 5 hinzu
>>> l
[0,2,5]
```

Listen - Erzeugung II

- auch mit Hilfe der Funktion `range` können wir Listen erzeugen
- `list(range(stop))`
- `list(range(start,stop[,step]))`

```
>>> list(range(3)) #Liste mit drei Elementen
[0, 1, 2]
>>> list(range(3,7)) #Liste von 3 bis vor 7
[3, 4, 5, 6]
>>> list(range(3,7,2)) #jedes zweite Element von 3
    bis vor 7
[3, 5]
```

Listen - Zugriff

```
>>> l = [1,2,3,4]
>>> l[1] #Index startet bei 0
2
>>> l[-1] #mit einem Minus kann man die Liste von
    hinten aufrollen
4
>>> l[0:2] #wie bei range
[1, 2]
>>> l[:2]
[1, 2]
>>> l[::2]
[1, 3]
>>> l[::-1] #Liste umdrehen
[4, 3, 2, 1]
```

Mengen - Erzeugung

- viele Operationen ähnlich wie bei Listen
- dazu klassische Mengenoperationen

```
>>> s = set([1,2,3])
>>> s.add(4) #fuegt ein Element hinzu
>>> s
set([1, 2, 3, 4])
>>> s.add(4) #doppeltes Hinzufuegen bewirkt nichts
>>> s
set([1, 2, 3, 4])
>>> s.pop() #entnimmt der Menge ein Element
1
```

Dictionaries (Map)

- bildet eindeutige Schlüssel (Keys) auf Elemente ab
- Typen können wie auch hier (fast) beliebig gewählt werden

```
>>> d = {1:'a', 2:'b', 3:'c'}
{1: 'a', 2: 'b', 3: 'c'}
>>> d[1] #liest den Wert zum Schlüssel 1 aus
'a'
>>> d.pop(1) #entfernt ein Paar
'a'
>>> d
{2: 'b', 3: 'c'}
>>> d[1]='d' #fügt ein neuen Paar hinzu
>>> d
{1: 'd', 2: 'b', 3: 'c'}
```

Tupel

- ähnlich wie Listen, aber unveränderbar
- beispielsweise zum Hashen benötigt

```
>>> tuple([1,2,3])
(1, 2, 3)
>>> (1,2,3)
(1, 2, 3)
>>> t = (1,2,3)
>>> a,b,c = t
>>> a
1
```

Übersicht

Funktionen - I

- wichtiges Abstraktionselement
- statt geschweifter Klammern werden Doppelpunkt und Einrückung genutzt
- keine Typen an den formalen Parametern
- Vorfestlegung von Parametern auf Default-Werte möglich

```
>>> def summe(a,b):
...     return a + b
...
>>> summe(5,9)
14
```

Funktionen - II

```
>>> summe(5,True)
6
>>> summe('hello ','world')
'hello world'
>>> def summe3(a,b=42,c=24):
...     return a + b + c
>>> summe3(5,9)
38
>>> summe3(5, c = 1)
48
>>> summe3(5,9,1)
15
```

Default-Parameter werden oft in Bibliotheken verwendet, um einfache und komplexe Benutzungen einer Funktion zu erlauben.

Bedingungen

- es gibt nur `if`, kein switch
- aber es gibt `elif` = `else if`

```
>>> x = 42
>>> if x < 0:
...     print('negative')
... elif x == 0:
...     pass #do nothing, implement later
... else:
...     print('positive')
...
'positive'
```

Schleifen - `for` I

- es gibt nur `for` und `while`, kein do while
- `for`-Schleifen haben selten einen expliziten Index → Iterator nutzen

```
>>> tiere = ['Hund', 'Vogel', 'Fisch']
>>> for t in tiere:
...     print(t, len(t))
...
Hund 4
Vogel 5
Fisch 5
```

Schleifen - for II

Durch die Funktion enumerate, lässt sich auch der Index explizit nutzen.

```
>>> tiere = ['Hund', 'Vogel', 'Fisch']
>>> for i,t in enumerate(tiere):
...     print(i, t)
...
0 Hund
1 Vogel
2 Fisch
```

Weitere Schlüsselwörter

break	eine Schleife abbrechen
continue	den Schleifendurchlauf abbrechen

Schleifen - while

Benutzung der while-Schleife wie üblich.

```
>>> ergebnis = 0
>>> while(ergebnis < 3):
...     ergebnis+=1
...     print(ergebnis)
...
1
2
3
```

Übersicht

Dokumentation

Kommentiert euren Code!

- kommentieren von Module und Funktionen mit `"""Inhalt"""`
- Zeilenkommentare mit `#` einleiten

```
"""
Modul mit aufregenden Funktionen.
"""

def helloWorld():
"""
Funktion schreibt 'Hello World' auf die Konsole.
"""
    print('Hello World') #Implementationskommentare
```

Module - I

- zur besseren Strukturierung ist eine Modularisierung anzustreben
- Module in Python sind `.py`-Skripte und können nach belieben selbst definiert werden

```
def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print (b)
        a, b = b, a+b
```

Speichern als Skript `fibonacci.py`.

Module - II

```
>>> import fibonacci as fibo
>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> from fibonacci import fib
>>> fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Nützliche Module

`math` viele Funktionen und Konstanten der Mathematik

`glob` Suche nach Dateien mit Wildcards

Übersicht

Was haben wir uns angesehen?

- Python als interpretierte Hochsprache
- einfach zu schreiben/lernen
- ermöglicht interaktives Arbeiten
- dynamisches Typsystem
- automatisches Speichermanagement
- Basistypen (Wahrheitswerte, Zahlen, `str`)
- Datenstrukturen (`list`, `set`, `dict`, `tuple`)
- Kontrollstrukturen (Funktionen, `if`, `for`, `while`)
- Kommentare, Module

Und wenn ich nicht mehr weiter weiß?

- Python-Tutorial `docs.python.org/3/tutorial/`
- Python-Referenz `docs.python.org/3/reference/index.html`

Nutzt Google!
Eure Probleme sind nicht exklusiv!
Aber kein blindes Kopieren!

Übersicht

Einstieg in Python

- Offizielles Tutorial zu Python (Kapitel 1-6 und 10.2 sowie 10.6 relevant) The Python Tutorial
- A. Sweigart, Automate the Boring Stuff with Python (Kapitel 0,1,2,3,4,5,6 relevant) E-Book unter CC-Lizenz
- Kombiniertes Kurztutorial zu Python und NumPy (mit Extra-Hinweisen für Matlab-Umsteiger) Python Numpy Tutorial von Justin Johnson
- J. Bernard, Python Recipes Handbook, 1st ed., apress, 2016 (Kapitel 1,2,3,6,10,11 tlw. relevant) Bib-Katalog/E-Book