

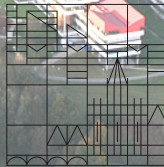
Computerphysik II

Python Einführung

S. Gerlach

WiSe 2017

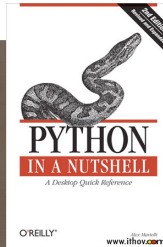
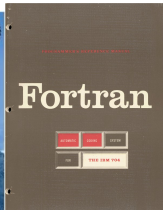
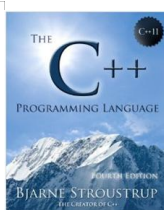
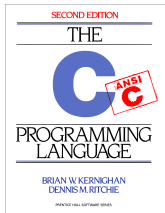
Universität
Konstanz



Programmiersprachen



In der Wissenschaft:



Programmiersprachen - Einteilung

Eigenschaften:

- ▶ **Skriptsprachen:** Shell, Python, Perl, ...
- ▶ **Kompilierte Sprachen:** C/C++, Fortran, Pascal, ...
- ▶ **Objektorientiert:** C++, Java, ...
- ▶ **Anwendungssprachen:** R, Mathematica, Matlab, ...

Anwendungen:

- ▶ **Skripte, Automatisierung:** Shell, Python, Make, ...
- ▶ **Datenauswertung & Tests:** C/C++, Python, Matlab, ...
- ▶ **Analytische Rechnungen:** Mathematica, Maple, SAGE, Python (SymPy), ...
- ▶ **HPC + Scientific Computing:** C/C++, Fortran, Python (NumPy, Scipy), ...
- ▶ **Grafische Darstellung:** gnuplot, Python (matplotlib), Mathematica, ...

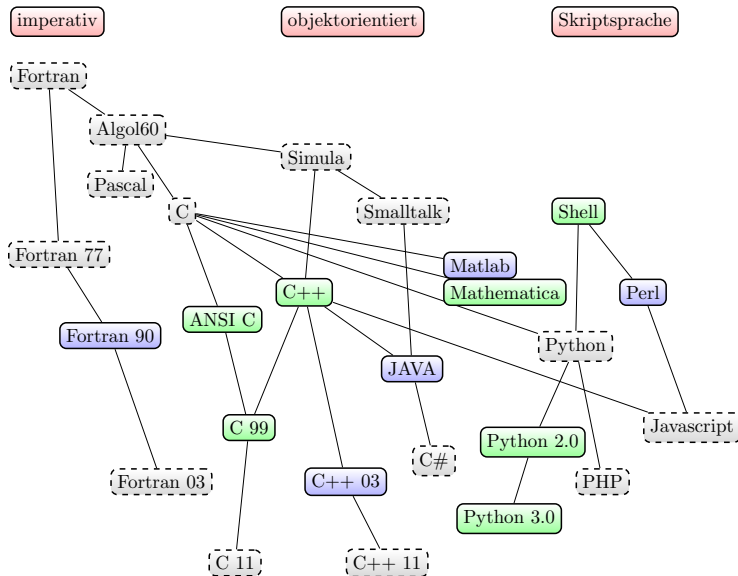
Programmiersprachen - Verbreitung (TIOBE 2015)

Very Long Term History

To see the bigger picture, please find the positions of the top 10 programming languages of many years back. Please note that these are average positions for a period of 12 months.

Programming Language	2015	2010	2005	2000	1995	1990	1985
C	1	2	1	1	2	1	1
Java	2	1	2	3	-	-	-
Objective-C	3	23	39	-	-	-	-
C++	4	3	3	2	1	2	12
C#	5	5	8	8	-	-	-
PHP	6	4	4	30	-	-	-
Python	7	6	6	24	22	-	-
JavaScript	8	8	9	7	-	-	-
Perl	9	7	5	4	9	19	-
Visual Basic .NET	10	-	-	-	-	-	-
Pascal	16	13	77	12	3	20	5
Lisp	18	16	12	15	5	3	2
Ada	30	26	15	16	6	4	3

Programmiersprachen in der Wissenschaft



Programmiersprache Python

- ▶ Hohe **Verbreitung** (gut bekannt, viele Bibliotheken)
- ▶ Freie Software
- ▶ interaktive Skriptsprache
- ▶ **performant** durch Verwendung von optimierten Bibliotheken bzw. Anbindung an C
- ▶ Einfach & flexibel (wenige Sprachfeatures, viele Bibliotheken)

aktuell: Python 2.7.14 (bis 2020 gepflegt), Python 3.6.3

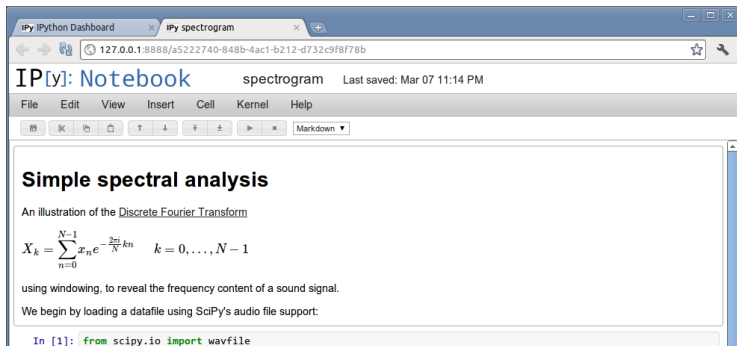


https://en.wikibooks.org/wiki/Python_Programming

[http://www.scipy-lectures.org/_downloads/
ScipyLectures-simple.pdf](http://www.scipy-lectures.org/_downloads/ScipyLectures-simple.pdf)

Verwendung - interaktiv

- ▶ **Terminal:** python, python3
>>> 1+1
- ▶ **IPython** - opt. Terminal: ipython3, ipython3 qtconsole
In[1]: %hist
In[2]: hist?
In[2]: help(abs)
- ▶ **Im Browser** - IPython-Notebook: ipython3 notebook



"Hello, World" in Python

A) Interaktiv:

```
>>> print('Hello, World!')  
Hello, World!
```

B) Python-Skript:

```
1 print ( ' Hello , _ World ! ' )
```

```
$ python3 skript.py
```

C) Shell-Skript:

```
1 #!/usr/bin/env python3  
2  
3 # Kommentar  
4 print ( ' Hello , _ World ! ' )
```

```
$ chmod +x skript.py
```

```
$ ./skript.py
```

```
Hello, World!
```

Datentypen in Python

- ▶ **Integer** (ganzzahlig): int

```
>>> 1+2
```

```
1 >>> 3/2
2 1.5 (Python 2)
3 1.5 (Python 3)
```

- ▶ **Fließkommazahl**: float

1.5, 2.

- ▶ **Komplexe Zahl**: complex

1.5+0.4j

- ▶ **Boolean**: bool

False, True

Variablen in Python

```
1 a = 4
2 b = 7.
3 c = 1.+2.j
4 d = True
```

→ Datentyp wird automatisch zugewiesen. Keine Deklaration nötig!

```
1 a, b = 1, 2
2 a = b = 2.
3 a += 1; a -= 1; a *= 2; a /= 2
4 a, b = b, a
```

Variablenausgabe

```
1 print(a)
2
3 print(a, b)
4
5 print('Ergebnis:', a)
6
7 print('\%e' \% a)
8 # e – Exponent, f – Floating, g – Automatic
9 # (wie in C)
```

Listen: Sammlung verschiedener Typen

```
1 a = [42, 1+2j, 'blau']
2
3 # Index beginnt bei 0 (wie C)
4 >>> a[0]
5 42
6 >>> a[-1]
7 'blau'
8 >>> a[1:]
9 [1+2j, 'blau']
```

Slicing: `a[start:stop:step]` ($start \leq i < stop$)

`len(a)`

Achtung: `b=a` (gleiches Objekt!), `a[:]=3,4,5` (in place)

String: unveränderbare Liste

Tuples: unveränderbare Listen `a = (1,2,3)`

Mengen: ungeordnete, unique Listen `a = set('a','b','c')`

Wörterbuch: Key-Value-Liste

```
1 tel = { 'Stefan':3825, 'Hans': 3815}
2 tel.keys()
3 tel.values()
```

if: Vergleich (==, !=, <, <=, > , >=)
(Einrückung!)

```
1  if i < 0:  
2      i = 0  
3  elif i == 0:  
4      i = 1  
5  else :  
6      print(i)
```

Varianten:

```
1  if 0 < x <= 1:  
2      ...  
3  if e in a:  
4      ...
```

for: Schleife mit fester Anzahl an Durchläufen

```
1 for e in a:  
2     print(e)
```

(Einrückung!)

```
1 for e in range(10):  
2     print(e)  
3  
4 # range(10) : [0, ..., 9]  
5 # range(6, 15, 3) : [6, 9, 12]
```

```
1 for index, item in enumerate(a):  
2     print(index, item)
```

Funktionale Programmierung:

```
1 zahlen = [1, 2, 3, 4, 5]
2 quadrate = [n**2 for n in zahlen]
3 print(quadrate)
```

while: Schleife mit unbekannter Anzahl an Durchläufen

```
1 b = 0
2 while b < 10:
3     print(b)
4     b += 2
```

```
1 def q(x):  
2     return x**2  
3  
4 # q(2) : 4  
5 # default parameter: f(a, b=0)
```

Funktionale Programmierung:

```
1 a = range(10)  
2  
3 map(q, a)  
4 # [0, 1, 4, 9, ..., 81]
```

verschiedene Möglichkeiten:

```
1 import bib
2 import bib as alias
3 from bib import * ____ # schlecht
4 from bib import function
```

Beispiele:

```
1 import numpy
2 numpy.array([1,2,3])
```

```
1 import numpy as np
2 np.array([1,2,3])
```

```
1 from numpy import array
2 array([1,2,3])
```

- ▶ **sys**: argv, version, path, ..
- ▶ **os**: getcwd(), listdir(), mkdir(), remove(), ..
- ▶ **math**: sin(), cos(), log(), exp(), pi, e, ..
- ▶ **cmath**: für komplexe Zahlen
- ▶ **random**: Zufallszahlen
- ▶ **tkinter**: Tk (GUI)
- ▶ **numpy**: NumPy (Numerische Methoden)
- ▶ **scipy**: SciPy (wissenschaftliche Anwendungen)
- ▶ **sympy**: SymPy (symbolisches Rechnen)
- ▶ **matplotlib**: Matplotlib (grafische Darstellung)

```
1 In [1]: import math
2 In [2]: help(math)
3 In [3]: print(dir(math))
```

Standardbibliotheken - GUI

```
1 from tkinter import *
2 fenster = Tk()
3 fenster.geometry("200x100")
4 label = Label(fenster, text="Hallo Welt!")
5 label.pack()
6 def befehl():
7     fenster.destroy()
8 button = Button(fenster, text="OK", command=befehl)
9 button.pack()
10 fenster.mainloop()
```



NumPy - NumPy-Arrays

NumPy-Arrays: optimiert für Zahlen (int/float)

```
1 from numpy import array
2 v = array([1,2,3,4])
3 M = array([[1.,2.],[3.,4.]])
4
5 v.shape_____# (4,1)
6 M.shape_____# (2,2)
7 v.size_____# 4
8 M.size_____# 4
9 v.dtype_____# int64
10 M.dtype_____# float64
11
12 v.reshape(2,2)____# array([[1, 2], [3, 4]])
13 v = array([1,2,3,4], dtype=complex)
```

NumPy - Listen

```
1 from numpy import arange, linspace, diag
2
3 arange(0,1,0.1)  $\rightarrow$   $[0, 0.1, \dots, 0.9]$ 
4
5 linspace(0.1,6)  $\rightarrow$   $[0, 0.2, 0.4, 0.6, 0.8, 1.0]$ 
6
7 diag([1,2])  $\rightarrow$   $[[1, 0], [0, 2]]$ 
```

```
1 from numpy import sin, cos
2
3 v=array([1,2,3])
4 sin(v)  $\rightarrow$   $[0.84147098, 0.90929743, 0.14112001]$ 
5 cos(v)  $\rightarrow$  ...
6
7 v+1, v*v, 2*x  $\rightarrow$  elementweise
```

NumPy - Matrix, etc.

```
1 from numpy import dot, matrix
2
3 dot(M, v)____# M.v
4 x=matrix(v).T____# transpose
5 x.T*x____# Skalarprodukt
6 M**2____# Matrixprodukt
```

- ▶ Mathematische Funktionen
- ▶ Statistik
- ▶ **numpy.polynomial**: Polynome
- ▶ **numpy.linalg**: Lineare Algebra
- ▶ **numpy.fft**: Fourier-Transformation
- ▶ **numpy.random**: Zufallszahlen/-verteilungen
- ▶ ...

- ▶ **scipy.constants**: (physik.) Konstanten
- ▶ **scipy.special**: Spezielle Funktionen
- ▶ **scipy.stats**: Statistik
- ▶ **scipy.optimize**: Optimierung
- ▶ **scipy.interpolate**: Interpolation
- ▶ **scipy.linalg**: Lineare Algebra
- ▶ **scipy.integrate**: Numerische Integration
- ▶ **scipy.fft**: Fourier-Transformation
- ▶ **scipy.signal**: Signalanalyse
- ▶ ...

```
1 from scipy.special import binom, j0
2
3 print(binom(4,2))_# 6
4 print(j0(1))_# 0.765..
```

Im Skript:

```
1 # in "python(3)":  
2 #import matplotlib as mpl  
3 #mpl.use("Qt4Agg")  
4 import pylab as pl  
5  
6 x = pl.arange(0,7,0.01)  
7 pl.plot(x,x**2)  
8 pl.show()
```

Im Notebook:

```
$ ipython notebook  
In[1]: %pylab inline  
In[2]: x=arange(0,10)  
In[3]: plot(x**2)
```

```
1 xlabel("x-Achse")
2 title("$\sin \pi x$")
3 xlim(0,5)
4 plot(x,y, '--r')
```

Legende:

```
1 plot(x,y1, label="sin")
2 plot(x,y2, label="cos")
3 legend(loc='best')
```

SymPy - Symbolisches Rechnen

```
1 import sympy as sp
2
3 e = sp.sqrt(8) # 2*sqrt(2)
4 e.evalf() # 2.1828...
5 e.evalf(50) # 50 Stellen
6 x,y = sp.symbols('x y')
7 e = x**2 - 2*y
8 e + x**2 # 2*x**2 - 2*y
9 e.subs(x, 1.5) # Einsetzen
```

```
1 >>> sp.init_printing() # nette Ausgabe
2     ---
3 2*\ / 2
4 >>> sp.latex(sp.sqrt(x**2))
5 2 \\sqrt{2}
```

- ▶ `expand((x+1)**2)`
 $x^2 + 2x + 1$
- ▶ `factor(x**3 - x**2 + x - 1)`
 $(x-1)(x^2+1)$
- ▶ `collect(x*y + x*x + x, x)`
 $x^2 + x(y+1)$
- ▶ `cancel((x**2+2*x+1)/(x**2+x))`
 $(x+1)/x$
- ▶ `apart((x**2+2*x)/(x+1))`
 $x+1 - 1/(x+1)$
- ▶ `trigsimp((sin(x))**2 + (cos(x))**2)`
 1
- ▶ `expand_trig(sin(x+y))`
 $\sin(x)\cos(y) + \sin(y)\cos(x)$

- ▶ `limit(sin(x)/x, x, 0)`
`1`
- ▶ `series(sp.cos(x))`
`1+x**2/2 + ...`
- ▶ `sp.sin(x).series(x,0,10)`
`x - x**3/6 + ...`
- ▶ `diff(x**2,x)`
`2*x`
- ▶ `solve(x**2-2, x)`
`(-sqrt(2), sqrt(2))`
- ▶ `summation(2*i+1,(i,1,n))`
`n**2 + 2*n`
- ▶ `integrate(x**2,x)`
`x**3/3`
- ▶ `integrate(sp.sin(x),(x,0,10)).evalf()`
`1.8390715...`