“*Quantitative* Naturwissenschaften”
→ Python

Sebastian Haase

Computerphysik

182bA2.4

20101201 VORLESUNG

Computergestützte Methoden der exakten Naturwissenschaften (Petra Imhof)

Zeit: Di 12:00-14:00, Do 12:00-14:00 (Erster Termin: 17.10.2017)

Ort: Di 1.3.14 Hörsaal A (Arnimallee 14), Do 1.3.14 Hörsaal A (Arnimallee 14)

Ich zeige Euch heute “was ihr eigentlich noch nicht wissen dürft”
(Zitat Ende – frei adaptiert ...)

```
You don't learn programming by learning a programming language.

Languages are tools that allow you to do something you've already
learned - situational analysis, also known as programming.

(Programming isn't typing things into a computer, it's thinking.)
```

Al Klein, 45 years of earning a living developing systems.

Was ist ein Computer ? Was ist ein Compiler ? Was ist “high-level”-Programmierung ?

1. von Neumann Architektur
   a. Eine CPU
   b. Ein Speicher
   c. Ein “Bus”

2. Interpreter vs Compiler
    a. JIT
3. Objekt-Orientiert

https://docs.python.org/3/tutorial

Python ist eine “schöne” Programmiersprache
- keine extra Zeichen … { } ; $
- Dynamische (aber strenge) Typisierung

Python ist strukturiert:
- Module
- Funktionen
- Objekte
    - Methoden

Python ist flexibel
- Einfache Programme: einfach ein paar Kommandos aneinanderreihen....
- Funktionen um Kommandos zu “gruppieren”
- Module um Funktionen zu “gruppieren”
- Objekte um “Daten und Algorithmen” zu kombinieren.
- .
- List-Comprehensions sind sehr praktisch: [x**2 for x in range(10) if x%2 ==0]

Python ist ready-to-go …
- Module für Web, Math, Linear-Alg., GUI, …

Python 1994
Numpy 2005 Numeric 1998? Numarray 2001?
SciPy 2001
AstroPy
Priithon Meins… http://msg.ucsf.edu/sedat/Priithon/PriithonHandbook.html
PyTables
SymPy
http://www.sympy.org/en/index.html
Gamma !
http://www.sympygamma.com/input/?i=integrate%28log%28x%29%2C+%28x%2C+1%2C+a%29%29

5**5**5

There are a few guidelines one has to realize when using Python:

1. A **"variable" is really "just a name"** for something !!
   Every assignment to a variable, like a=5 or a=F.fft(a) throws the old meaning (value) of **a** away and **reassigns a** to whatever is the result of what's on the rights side of = -sign.
   But note: a[:] = F.fft(a) is *NOT* an assignment: there is more than **"just the variable name"** left of the =-sign.
   In other words: is a was a big array, a[:]= 2 would overwrite a's values (each 'pixel' would be set to 2, whereas a=2 would destroy that array and a would from then on refer to the number 2.
2. **Indentation** matters: This must be the strangest "feature" of python:
   - Instead of using curly brackets (like in C/C++/Java)
   - or BEGIN/END (like in PASCAL)
   - or do/done (still others) Python infers the "bracketing" from the way a code-section is indented.
   - This feature seems very strange at first - but it is supposed to make everyone's code look more alike;
     Decide if you want to use tabs or spaces to indent; there are *pros* and *cons* for each choice -- but **stick** with it and don't mix!!
3. Everything is **case-sensitive**
4. Every variable lives inside its **modules**
5. **Importing** a module makes its variables visible - **all code** inside the module's defining file gets **execute**d only the first time it gets imported (by anything - since python started)

Nochmal…

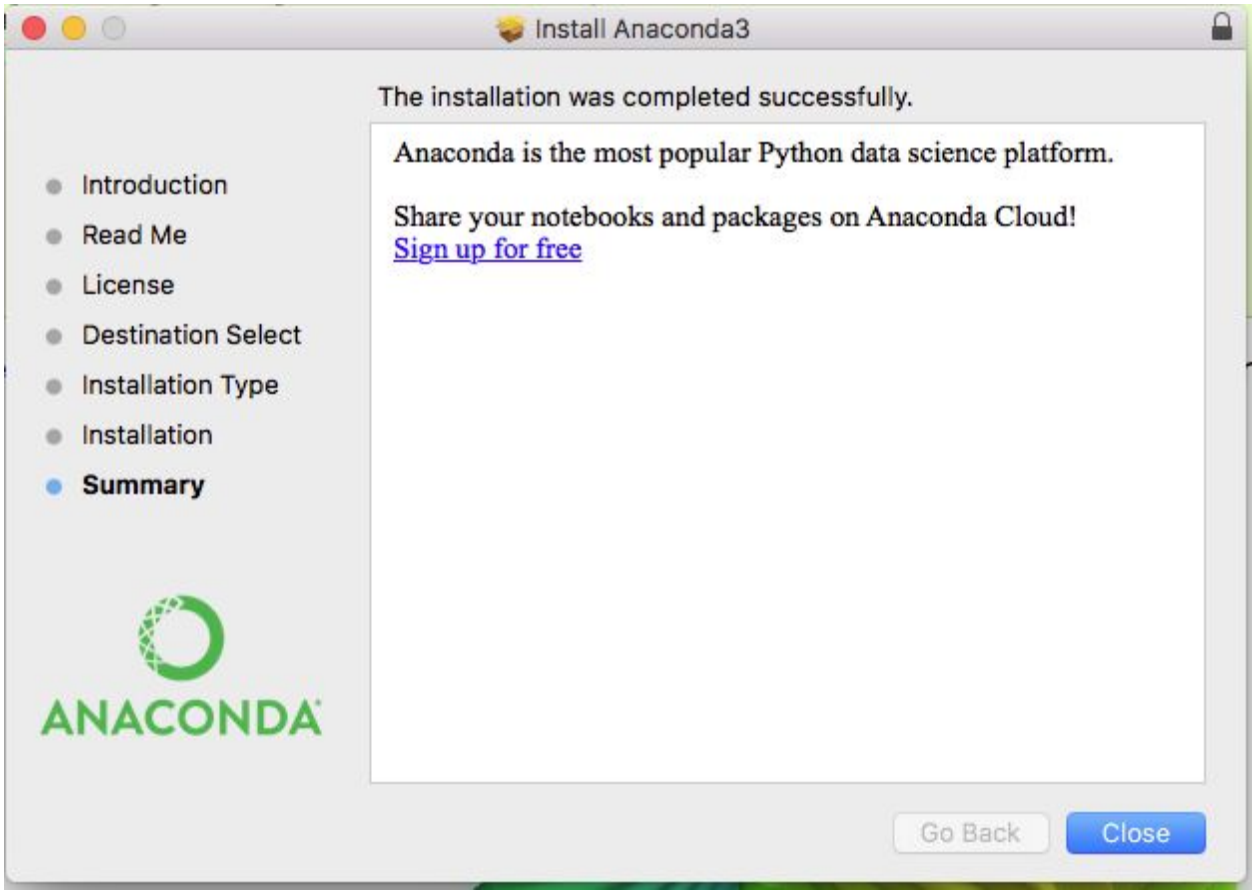
range(5) fängt bei 0 an hört VOR 5 auf also [0,1,2,3,4]
range(0, 5, 2) ? [0, 2, 4]
Argumente können auch als Keyword-Argument übergeben werden …. Macht Code oft lesbarer !!

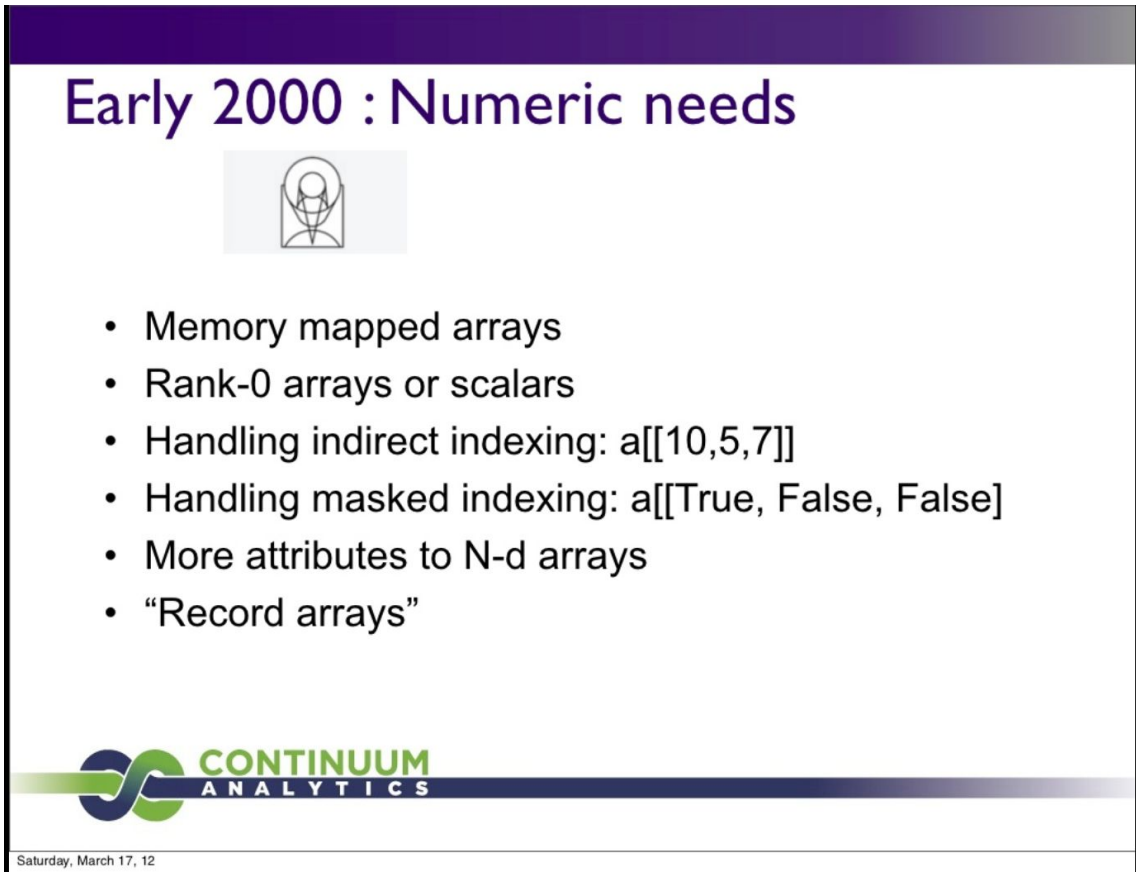
N.arange(5, step=.5)
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])

(Ausnahme: built-in range(10,step=2) → TypeError: range() takes no keyword arguments)

List-Comprehensions sind sehr praktisch: [x**2 for x in range(10 if x%2 ==0]

Praktisches print – bzw. praktische Strings:
"-" * 80
print(3,4,5, sep=' - ') # übrigens ' ist identisch zu " und """ und ''' sind gut für mehrzeilige Strings

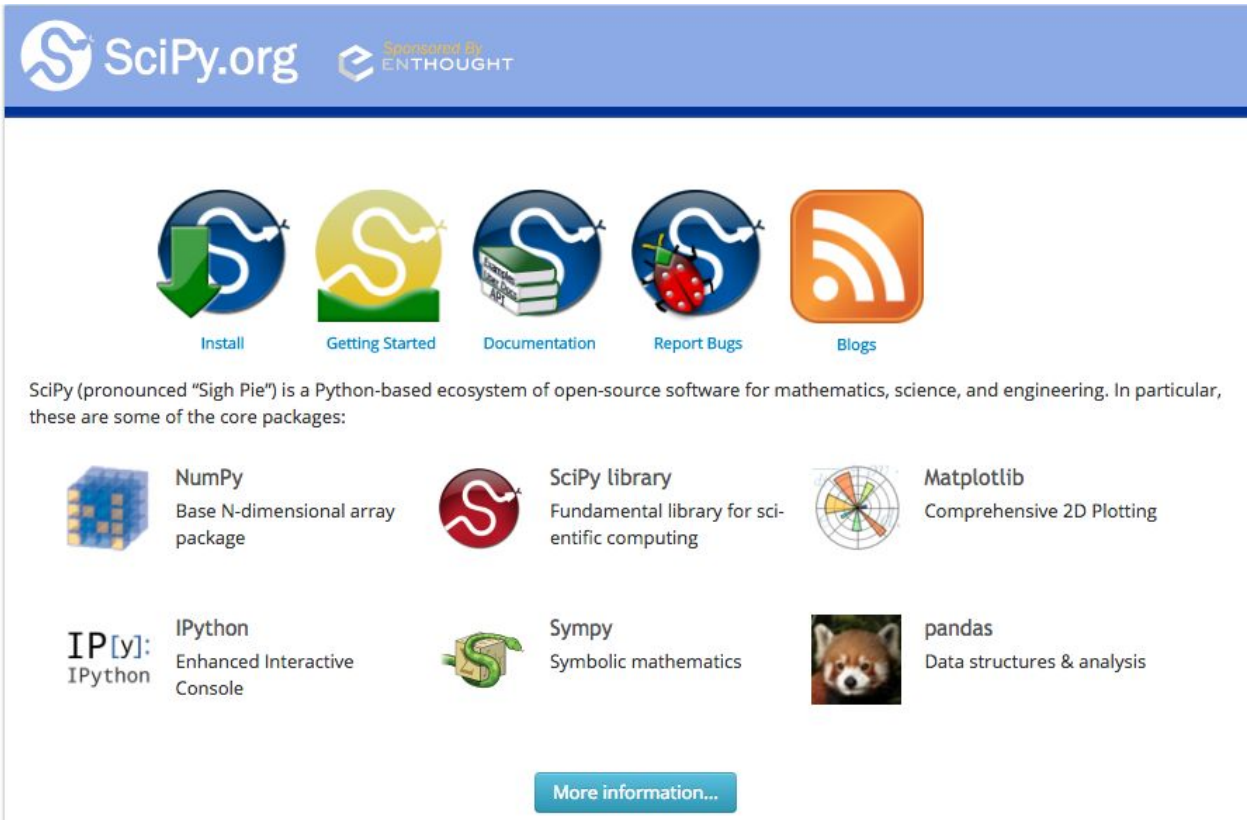

Install Anaconda3
The installation was completed successfully.
Introduction
Read Me
License
Destination Select
Installation Type
Installation
Summary
Anaconda is the most popular Python data science platform.
Share your notebooks and packages on Anaconda Cloud!
Sign up for free
ANACONDA
Go Back
Close



Travis E. Oliphant, "NumPy and SciPy: History and Ideas for the Future"
https://www.slideshare.net/shoheihido/sci-pyhistory

NumPy and SciPy for Data Mining and Data Analysis Including iPython, SciKits, and matplotlib
https://www.slideshare.net/bytemining/numpy-and-scipy-for-data-mining-and-data-analysis-including-ipython-scikits-and-matplotlib

https://www.scipy.org/



http://scipy-cookbook.readthedocs.io/

https://www.dataquest.io/blog/numpy-tutorial-python/

https://jakevdp.github.io/PythonDataScienceHandbook/02.07-fancy-indexing.html

Und nochmal…

Python Assignment != memory-copy / CPU-work
b   = a   # no work !   same object !
b[:] = a # work !

Numpy:
Zeilen-Vektoren   Spalten-Vektoren

Broadcasting ….
a=np.arange(5)
a[[0,0,1,0,1] = 99

%timeit   numpy ….

Jupyter

https://blog.dominodatalab.com/lesser-known-ways-of-using-notebooks/

z.B.

```
%%latex
\begin{align}
\nabla \cdot \vec{\mathbf{E}} & = 4 \pi \rho \\
\nabla \times \vec{\mathbf{E}}\, +\, \frac1c\,
\frac{\partial\vec{\mathbf{B}}}{\partial t} & = \vec{\mathbf{0}} \\
\nabla \cdot \vec{\mathbf{B}} & = 0
\end{align}
```

In [31]:
```
%%latex
\begin{align}
\nabla \cdot \vec{\mathbf{E}} & = 4 \pi \rho \\
\nabla \times \vec{\mathbf{E}}\, +\, \frac1c\, \frac{\partial\vec{\mathbf{B}}}{\partial t} & = \vec{\mathbf{0}} \\
\nabla \cdot \vec{\mathbf{B}} & = 0
\end{align}
```

$$\nabla \cdot \vec{\mathbf{E}} = 4\pi\rho$$

$$\nabla \times \vec{\mathbf{E}} + \frac{1}{c}\frac{\partial \vec{\mathbf{B}}}{\partial t} = \vec{\mathbf{0}}$$

$$\nabla \cdot \vec{\mathbf{B}} = 0$$

https://www.python-course.eu/numpy.php

Swap rows: a[2], a[1] = a[1], a[2].copy() # .copy() weil zwischendurch überschrieben!!

https://stackoverflow.com/questions/14933577/swap-slices-of-numpy-arrays/14933939#14933939

http://jakevdp.github.io/blog/2013/06/15/numba-vs-cython-take-2/

https://stackoverflow.com/questions/36661876/what-is-the-difference-between-importing-matplotlib-and-matplotlib-pyplot

https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/__init__.py

Mayavi

A demo

To get you started, here is a pretty example showing a spherical harmonic as a surface:

```
# Create the data.
from numpy import pi, sin, cos, mgrid
dphi, dtheta = pi/250.0, pi/250.0
[phi,theta] = mgrid[0:pi+dphi*1.5:dphi,0:2*pi+dtheta*1.5:dtheta]
m0 = 4; m1 = 3; m2 = 2; m3 = 3; m4 = 6; m5 = 2; m6 = 6; m7 = 4;
r = sin(m0*phi)**m1 + cos(m2*phi)**m3 + sin(m4*theta)**m5 + cos(m6*theta)**m7
x = r*sin(phi)*cos(theta)
y = r*cos(phi)
z = r*sin(phi)*sin(theta)

# View it.
from mayavi import mlab
s = mlab.mesh(x, y, z)
mlab.show()
```

Bulk of the code in the above example is to create the data. One line suffices to visualize it. This produces the following visualization: