



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FACHBEREICH PHYSIK
HAMBURGER STERNWARTE

WOLFRAM SCHMIDT & MARCEL VÖLSCHOW

PYTHON/JUPYTER FÜR ASTRONOMIE UND ASTROPHYSIK I



1. ALLGEMEINES UND INSTALLATION

WARUM PYTHON?

- Einfach zu lernen, einfach in der Anwendung
- Interpreter: Code kann direkt ausgeführt werden
- Vielseitige Datenstrukturen wie Listen und Wörterbücher
- Generische Datentypen für Operatoren und Funktionen
- Zahlreiche nützliche Pakete verfügbar
- Datenanalyse und Visualisierung in der Astrophysik



ONLINE-TUTORIALS ZUM SELBSTSTUDIUM


Das Python-Tutorial (deutsch, Version 3.3):
py-tutorial-de.readthedocs.io/de/python-3.3

The Python Tutorial (englisch, Version 3.6):
docs.python.org/3/tutorial

PYTHON IM WEBBROWSER: DIE JUPYTER NOTEBOOK APP

- Notebooks erlauben strukturiertes Arbeiten
- Eingabe, Ausführen, Ansehen auf einmal möglich
- Grafische Ausgaben direkt sichtbar
- Kernel kann auf leistungsfähigerem Rechner laufen und interaktiv über das Internet bedient werden
- Mehr unter jupyter-notebook.readthedocs.io/en/stable



 jupyter Hello world (autosaved)



Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3 



```
In [1]: 1 # This program prints Hello, world!  
        2  
        3 print('Hello, world!')
```




Hello, world!

```
In [ ]: 1
```

CONDA

- Paket- und Umgebungsmanagementsystem für Python
- Schnelle und einfache Installation auf allen Computern
- Kein spezielles Knowhow, keine Administratorrechte erforderlich
- Je nach verfügbarem Speicherplatz und Internetverbindung: Miniconda oder Anaconda

Miniconda

	 Windows	 Mac OS X	 Linux
Python 3.6	64-bit (exe installer) 32-bit (exe installer)	64-bit (bash installer)	64-bit (bash installer) 32-bit (bash installer)
Python 2.7	64-bit (exe installer) 32-bit (exe installer)	64-bit (bash installer)	64-bit (bash installer) 32-bit (bash installer)

Installationspakete für alle Betriebssysteme unter conda.io/miniconda.html verfügbar

INSTALLATION

- Anleitungen für Basisinstallation:
 - Windows: conda.io/docs/user-guide/install/windows.html
 - Mac: conda.io/docs/user-guide/install/macos.html
 - Linux: conda.io/docs/user-guide/install/linux.html
- Aktualisieren über Anaconda-Prompt (Windows) bzw. im Terminal (Mac/Linux)

```
conda update -n base conda
```

JUPYTER INSTALLIEREN UND STARTEN

- Installation (nur Miniconda):

```
conda install jupyter
```

- Starten eines lokalen Notebook-Servers (Kernels) und Öffnen des Dashboards im Standard-Browser:

```
jupyter notebook
```

- Siehe jupyter.readthedocs.io/en/latest/running.html



2. VARIABLEN, FUNKTIONEN, LISTEN UND SCHLEIFEN

(ASTRO)PHYSIKALISCHE VARIABLEN MIT EINHEITEN

- Zahlreiche physikalische Konstanten und Einheiten sind im Paket `astropy` definiert (docs.astropy.org/en/stable/constants)
- Pakete oder Module lassen sich mit dem Befehl `import` laden, bestimmte Namen daraus mittels `from`

```
from astropy import constants as const # importiere constants aus astropy und verwende es unter dem Namen const
```

```
print(const.c)
```

```
Name      = Speed of light in vacuum
Value     = 299792458.0
Uncertainty = 0.0
Unit      = m / s
Reference = CODATA 2010
```

- Variablen lassen sich mit den üblichen arithmetischen Operatoren definieren und verknüpfen ([tutorial/introduction.html#numbers](https://www.astropy.org/astropy-tutorial/introduction.html#numbers))

Leuchtkraft $L = \dot{M}c^2$

```
from astropy import units as u
```

```
mass_defect_rate = 4.3e9*u.kg/u.s # Massendefekt durch Kernfusion in kg/s
```

```
luminosity = mass_defect_rate*(const.c)**2
```

```
print(luminosity) # in W
```

```
3.8646472685683156e+26 kg m2 / s3
```

```
print(luminosity.to(u.W))
```

```
3.8646472685683156e+26 W
```

```
print(luminosity.cgs) # in cgs-Einheiten
```

```
3.864647268568316e+33 erg / s
```

FUNKTIONEN <tutorial/controlflow.html#defining-functions>

- Definition wird durch Schlüsselwort **def** eingeleitet
- Anweisungen müssen eingerückt sein!
- Rückgabe eines Wertes mittels **return**

Strahlungsenergie pro Zeit- und Flächeneinheit: $F = L/(4\pi r^2)$

```
import math
```

```
def F(r): # Parameter: Radius in beliebiger Einheit
    print("Abstand r =", r)
    return const.L_sun/(4*math.pi*(r.to(u.m))**2)
```

```
F(1.0*u.au)
```

Abstand r = 1.0 AU

1361.1665 $\frac{\text{W}}{\text{m}^2}$

if-ANWEISUNG tutorial/controlflow.html#if-statements

- Anweisungen werden in Abhängigkeit eines logischen Ausdrucks (ergibt **True/False**) ausgeführt

```
def F(r, cgs=False):          # optionaler Parameter: Rückgabewert in cgs-Einheiten, wenn True
    surface = 4*math.pi*r**2 # nur innerhalb der Funktion definiert (lokale Variable)
    if cgs:
        return const.L_sun.to(u.erg/u.s)/surface.to(u.cm**2)
    else:
        return const.L_sun/surface.to(u.m**2)
```

```
F(1.0*u.au)
```

1361.1665 $\frac{\text{W}}{\text{m}^2}$

```
F(1.0*u.au, cgs=True)
```

1361166.5 $\frac{\text{erg}}{\text{s cm}^2}$

LISTEN

<tutorial/introduction.html#lists>

- Bestehen aus beliebigen indizierten Elementen
- Hinzufügen neuer Elemente mit `append`

Mittlere Abstände der inneren Planeten von der Sonne in Mio. km

```
data = [58, 108, 150] # Liste von Zahlen
```

```
data[0] # Merkur
```

58

```
data[2] # Erde
```

150

```
data.append(228) # füge Element für Mars hinzu
print(data)
```

[58, 108, 150, 228]

for-SCHLEIFEN [tutorial/controlflow.html#for-statements](https://www.python.org/tutorial/controlflow.html#for-statements)

- Durch Schleifen können Anweisungen der Reihe nach für jedes Element einer Liste ausgeführt werden

```
r_planets = [] # Anlegen einer noch leeren Liste
print("Liste enthält {:1d} Elemente".format(len(r_planets)))

for value in data: # Schleife über Datenwerte
    r_planets.append(value*1e6*u.km) # füge r in km hinzu

print("Liste enthält {:1d} Elemente".format(len(r_planets)))
```

```
Liste enthält 0 Elemente
Liste enthält 4 Elemente
```

```
r_planets[3] # Mars
```

```
2.28 × 108 km
```

```
r_planets[-1] # letztes Element der Liste
```

```
2.28 × 108 km
```

FORMATIERTE AUSGABE

pyformat.info

- Alte Formatierung im Stil von Fortran/C
- Neue Formatierung sehr vielseitig, aber kompliziert
- Verwendung anhand von Beispielen

Berechnung der Strahlungsflussdichte

```
for r in r_planets:
    print("Strahlungsflussdichte = {0:4.0f} im Abstand {1:.2e} von der Sonne".format(F(r), r))
```

```
Strahlungsflussdichte = 9055 W / m2 im Abstand 5.80e+07 km von der Sonne
Strahlungsflussdichte = 2612 W / m2 im Abstand 1.08e+08 km von der Sonne
Strahlungsflussdichte = 1354 W / m2 im Abstand 1.50e+08 km von der Sonne
Strahlungsflussdichte = 586 W / m2 im Abstand 2.28e+08 km von der Sonne
```