1. Einführung in Python

Markus Reinert[1]

Leibniz-Institut für Ostseeforschung Warnemünde (IOW)

16. April 2021

[1] ✉ markus.reinert@io-warnemuende.de

Übersicht

Python-Kurs 1

- Warum Python?
- Wie ist Python?
- Wie verwendet man Python?
- Warm-up: Kopfrechnen
- Hübsch formatierte Zeichenketten
- Alles verändert sich
- . . . oder doch nicht?

Warum Python?

TIOBE Index April 2021:
Indikator für die Beliebtheit von Programmiersprachen
`https://www.tiobe.com/tiobe-index/`

1	C
2	Java
3	**Python**
⋮	
16	R
⋮	
19	Matlab
20	Fortran

"Python plays a key role in our production pipeline. Without it a project the size of Star Wars: Episode II would have been very difficult to pull off."
`https://www.python.org/about/quotes/`

→ Python ist beliebt und praktisch
(und bestens für wissenschaftliche Anwendungen geeignet)

Wie ist Python?

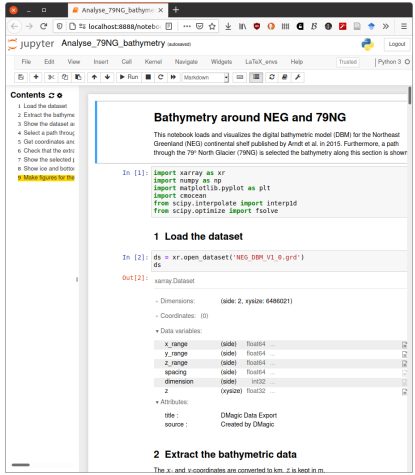
Siehe: `https://m.xkcd.com/409/`

Wie verwendet man Python?

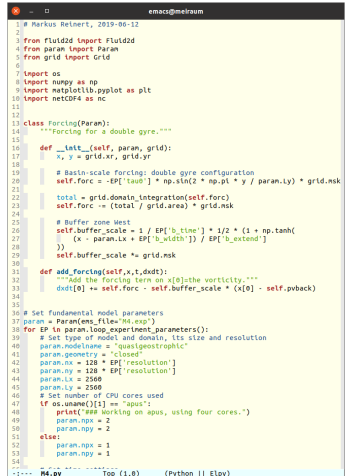
Interaktiv im Terminal:

```
markus@meiraum: ~
markus@meiraum$ python
Python 3.7.9 (default, Aug 31 2020, 12:42:55)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world!")
Hello world!
>>> # Python as a calculator
>>> 7 * 6
42
>>> # More complex tasks
>>> for i in range(5):
...     print("i =", i, "and i*i =", i*i)
...
i = 0 and i*i = 0
i = 1 and i*i = 1
i = 2 and i*i = 4
i = 3 and i*i = 9
i = 4 and i*i = 16
>>> institute = "IOW"
>>> institute
'IOW'
>>> print(institute)
IOW
>>> # Note the difference
```

Als Jupyter Notebook im Internet-Browser:



Als Skript im Texteditor:



Grundlegende Arithmetik

Operation	Operator	Beispiel
Addition	`+`	`1 + 2`
Subtraktion	`-`	`2 - 1`
Multiplikation	`*`	`3 * 4`
Division	`/`	`9 / 4` (gibt 2.25)
Ganzzahl-Division	`//`	`9 // 4` (gibt 2)
Modulo-Rechnung	`%`	`9 % 4` (gibt 1)
Potenzrechnung	`**`	`3 ** 2` (gibt 9)
Wurzelziehen	`**(1/2)`	`9 ** (1/2)` (gibt 3.0)

Beachte:
Integer (Ganzzahlen) werden automatisch in *Floats* (Fließkommazahlen) umgewandelt, wenn es notwendig ist oder wenn ein Operand ein Float ist.
. . . also eigentlich gibt es doch nichts zu beachten (außer man benutzt Python 2).
Außerdem: nur runde Klammern `()` dürfen in Rechnungen verwendet werden.

Strings (Zeichenketten) 1

Strings definiert man wahlweise so: `"Ich bin ein Text"` oder so `'Ich auch'`.

Und wenn der Text Anführungszeichen enthält?

1. `"Er sagte 'Hallo'."` (doppelte Anführungszeichen sind „unsichtbar“)
2. `'Er sagte "Hallo".'` (einfache Anführungszeichen sind „unsichtbar“)
3. `'Er sagte \'Hallo\'.'` (equivalent zu 1.)
4. `"Er sagte \"Hallo\"."` (equivalent zu 2.)

Und wenn der Text besonders lang ist?

- ```
  """Dieser Text geht über drei (!) Zeilen
  und kann " sowie ' enthalten.
  """
  ```
- hauptsächlich als Kommentar / Beschreibung (einer Funktion *etc.*) verwendet
  ```

Strings (Zeichenketten) 2

Mit Strings kann man rechnen:

- `"Hallo " + "Welt"` ergibt `"Hallo Welt"`
- `3 * "wiu"` ergibt `"wiuwiuwiu"`

Achtung: `"4" + "2"` ergibt `"42"` und `"1" * 5` ergibt `"11111"`

Um mathematisch zu rechnen:

- `int("4") + int("2")` ergibt `6`
- `float("0.5") + float("0.5")` ergibt `1.0`
- andersrum: `str(3/2)` ergibt `"1.5"`

Hilfreiche Funktionen:

- `"  Wörter mit Leerzeichen  ".strip()` ergibt `"Wörter mit Leerzeichen"`
- `"Das_ist_ein_Text".replace("_", " ")` ergibt `"Das ist ein Text"`

Strings (Zeichenketten) 3

Gute Möglichkeiten um Zahlen in Text einzufügen:

1. `"The density is {:.1f} kg/m³.".format(1030.49)`
   - die Zahl im Argument von `format` wird verwendet

2a. `f"The density is {rho:.1f} kg/m³."`
   - der Wert der Variable `rho` wird verwendet

2b. `f"The density anomaly is {rho-1000:.1f} kg/m³."`
   - einfache Rechnungen sind möglich
   - komplexe Rechnungen auch, aber dafür ist evtl. Möglichkeit 1 übersichtlicher

Allgemein gilt:

- die Zahl wird anstelle der geschweiften Klammern eingefügt (`f` für Float)
- die Zahl wird auf 1 Nachkommastelle gerundet (wegen `.1`)
- und es gibt noch viele weitere Formatierungsmöglichkeiten

Variablen:

- werden in Python mit = definiert
- können beliebige Daten/Objekte enthalten:
  - Zahlen: `x = 3`
  - Strings: `text = "Hallo"`
  - Listen, Funktionen, ...
- können Namen aus Buchstaben, Zahlen und Unterstrichen haben, aber Namen dürfen nicht mit einer Zahl beginnen
- sollten **sinnvolle** Namen haben

Konstanten:

- gibt es in Python nicht
- jede Variable kann überschrieben bzw. verändert werden
- Konvention: Variablennamen in GROSSBUCHSTABEN stehen für Konstanten

Grundlegende Datensammlungen

Liste

Sammlung beliebiger Daten

```
temp = [20, 22, 20.5, "error", 21]
```

meist für *unbekannte* Anzahl von Daten *desselben* Typs verwendet

Tuple

unveränderbare Liste

```
dimensionen = (128, 256, 32)
```

meist für *bekannte* Anzahl von Daten *unterschiedlichen* Typs verwendet

Named Tuple – oft besser geeignet als ein einfaches Tuple:

```
# zu Beginn des Programms:
from collections import namedtuple
# bevor das Tuple verwendet wird:
Dimension = namedtuple("Dimension", ("x", "y", "z"))
# Definition des NamedTuples:
dim = Dimension(128, 256, 32)     # oder: Dimension(x=128, y=256, z=32)
# Verwendung:
dim.x     # ergibt 128
dim.z     # ergibt 32
```