

1. Einführung in Python

Markus REINERT¹

Leibniz-Institut für Ostseeforschung Warnemünde (IOW)

16. April 2021

¹ ✉ markus.reinert@io-warnemuende.de

Python-Kurs 1

Warum Python?

Wie ist Python?

Wie verwendet man Python?

Warm-up: Kopfrechnen

Hübsch formatierte Zeichenketten

Alles verändert sich

... oder doch nicht?

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)

Siehe: <https://m.xkcd.com/409/>

Interaktiv im Terminal:

```

markus@meiraum:~$ python
Python 3.7.9 (default, Aug 31 2020, 12:42:55)
[CC 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:

The screenshot shows a Jupyter Notebook in a web browser. The notebook title is "Analyse_79NG_bathymetry". The main content area displays the title "Bathymetry around NEG and 79NG" and a brief introduction. Below the text, there are two code cells. The first cell, labeled "In [1]:", contains the following code:

```

import xarray as xr
import numpy as np
import matplotlib.pyplot as plt
import crocena
from scipy.interpolate import interpfd
from scipy.optimize import fsolve

```

The second cell, labeled "In [2]:", contains:

```

ds = xr.open_dataset('NEG_DBR_V1_0.grd')
ds

```

The output of the second cell is a detailed summary of the xarray Dataset:

```

Out[2]:
xarray.Dataset
Dimensions: (side: 2, xysize: 6480021)
Coordinates: ()
Data variables:
  x_range (side) float64 ...
  y_range (side) float64 ...
  z_range (side) float64 ...
  spacing (side) float64 ...
  dimension (side) int32 ...
  z (xysize) float32 ...
Attributes:
  title: DMagic Data Export
  source: Created by DMagic

```

At the bottom of the notebook, there is a section titled "2 Extract the bathymetric data" with the text: "The x- and y-coordinates are converted to km. z is kept in m."

Als Skript im Texteditor:

The screenshot shows a Python script in a text editor. The script is titled "Markus Reintert, 2019-06-12". It imports various modules and defines a class "Forcing(Param)". The class has a method "add_forcing" and a property "buffer_scale". The script also sets fundamental model parameters and prints the working on apus, using four cores.

```

1 # Markus Reintert, 2019-06-12
2
3 from fluid2d import Fluid2d
4 from param import Param
5 from grid import Grid
6
7 import os
8 import numpy as np
9 import matplotlib.pyplot as plt
10 import netCDF4 as nc
11
12
13 class Forcing(Param):
14     """Forcing for a double gyre."""
15
16     def __init__(self, param, grid):
17         self.x, y = grid.xr, grid.yr
18
19         # Basin-scale forcing: double gyre configuration
20         self.forc = -EP['tau0'] * np.sin(2 * np.pi * y / param.Ly) * grid.msk
21
22         total = grid.domain_integration(self.forc)
23         self.forc -= (total / grid.area) * grid.msk
24
25         # Buffer zone West
26         self.buffer_scale = 1 / EP['b_tline'] * 1/2 * (1 + np.tanh(
27             (x - param.Lx + EP['b_wldth']) / EP['b_extend']
28         ))
29         self.buffer_scale *= grid.msk
30
31     def add_forcing(self, x, t, dxdt):
32         """Add the forcing term on x[0]=the vorticity."""
33         dxdt[0] += self.forc - self.buffer_scale * (x[0] - self.pvback)
34
35
36 # Set fundamental model parameters
37 param = Param(ens_file="M4.exp")
38 for EP in param.loop_experiment_parameters():
39     # Set type of model and domain, its size and resolution
40     param.modelname = "quasigeostrophic"
41     param.geometry = "closed"
42     param.nx = 128 * EP['resolution']
43     param.ny = 128 * EP['resolution']
44     param.Lx = 2560
45     param.Ly = 2560
46     # Set number of CPU cores used
47     if os.uname()[1] == "apus":
48         print("### Working on apus, using four cores.")
49         param.npx = 2
50         param.npy = 2
51     else:
52         param.npx = 1
53         param.npy = 1
54
55 #-----
56 #---- M4.py Top (1,0) (Python 11 Elpy)
57 Mark set

```

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 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

Mit Strings kann man rechnen:

- ▶ `"Hallo " + "Welt"` ergibt `"Hallo Welt"`
- ▶ `3 * "wiu"` ergibt `"wiuwuiwui"`

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"`

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

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
```