

# Integrated Practical: Python in Geosciences



**Mathias Hauser, Doris Folini, Victor Wattin Hakansson**

**TAs: Stefanie Börsig, Svenja Seeber**

Former lecturers, contributors and TAs

Chahan Kropf, Alessio Cullo, Urs Beyerle, Sylvaine Ferrachat  
Boriana Chtirkova, Shuchang Liu, Ruolan Xiang

# Objectives

- Enable you to write python code for your own needs
- Use own laptop (become independent of pre-set environment)
- Write quality code (clean and re-usable, for you and others)
- Handle and visualize geoscience data (wide spread data formats, geographical maps)
- Collaborate in teams (team work, version control, commenting)
- Scripting (for efficient, robust, and transparent work-flow)

# Contents & Schedule

- Install python on your laptop
- Python basics
- Project work
  - 3 real-world projects
  - teams of 2-3 for 6 half-days
  - select by end of Tuesday

date	half-day	topic
<b>Mo 11.3.</b>	<b>HD1</b>	<b>Welcome, project presentation &amp; set up</b>
<b>Tu 12.3.</b>	<b>HD2</b>	<b>python intro (+ select project)</b>
<b>We 13.3.</b>	<b>HD3</b>	<b>python intro + questions</b>
	HD4	work on project
Mo 18.3.	HD5	work on project
<b>Tu 19.3.</b>	<b>HD6</b>	<b>mid-term status and questions + work on project</b>
We 20.3.	HD7	work on project
	HD8	work on project
Mo 25.3.	HD9	work on project
<b>Tu 26.3.</b>	<b>HD10</b>	<b>Presentations</b>

**Bold = attendance required**

# Course requirements

- To get the 2 ECTS:
  1. participate in introduction and mid-term question round
  2. hand in working and commented python code
  3. present your work (~10 min) on the last half-day
  4. be present in CHN E46 on 5 half-days:  
Mo 11.3. / Tu 12.3. / We 13.3. (morning) / Tu 19.3. / Tu 26.3.

# Presentation

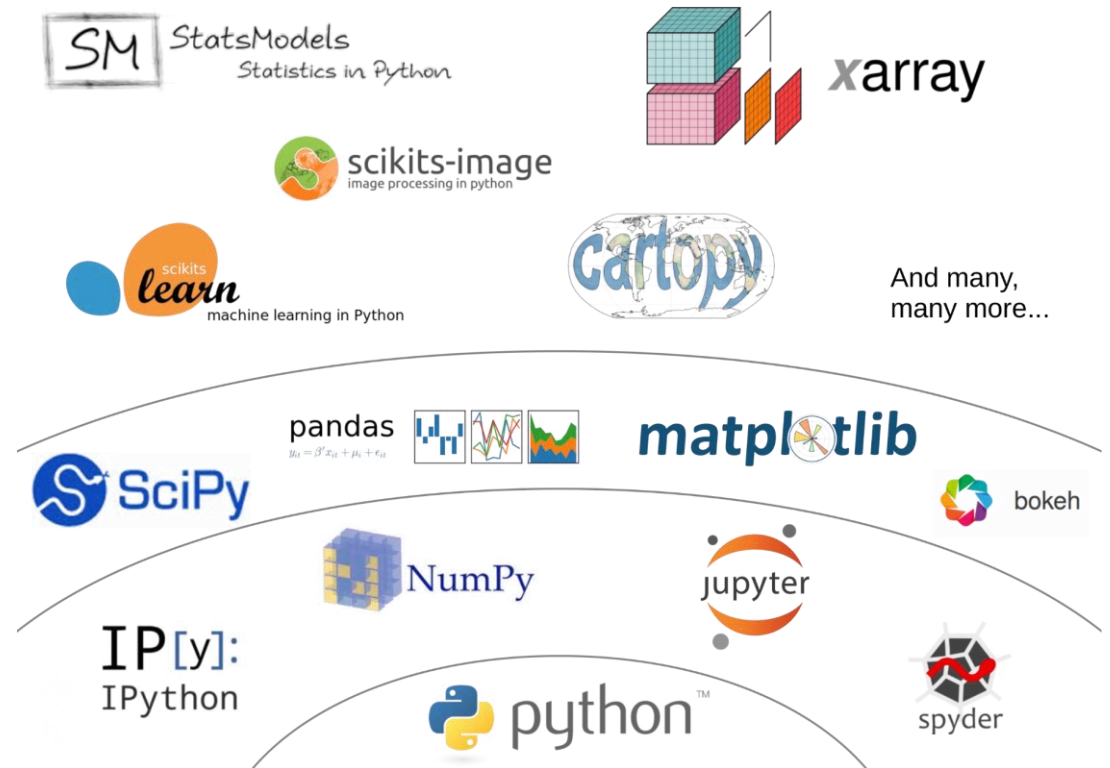
- On the last half day
- > 10 min overview of the project
  - What did work? Interesting parts? What did you learn?
  - What did not work? Challenges?

# Projects

- **Climate Risk:** estimate damage costs of storms hitting settlements
- **CMIP Data:** look at simulation data behind the IPCC reports
- **Climate in Climate Economics:** simple climate models for carbon taxes

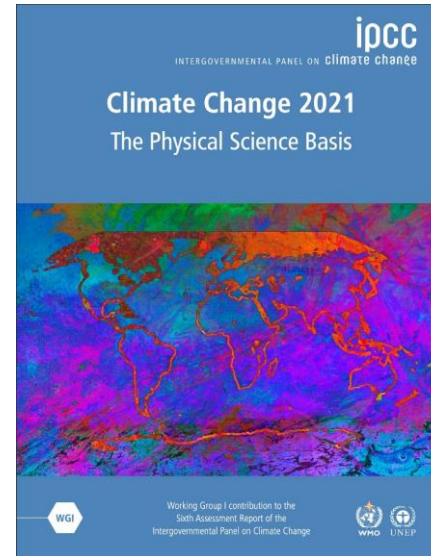
# Python

- a high level programming language, similar to matlab or R
- popular, wide spread, has a large and growing user community
- behind many open source projects – a universe of modules for different purposes



# Python applications

- analyze and plot existing data
- write numerical simulations (e.g. a computer code to calculate pi) and produce data
- front-end code (what PhD student sees / edits) of large simulation codes (e.g. climate model, 106 lines of Fortran or C code for 'number crunching')





# Using python

- **jupyter**: 'web interface', integrates code and results (text, images)
- **ipython**: command line interface, run (develop, test) python code
- **python**: command line interface like ipython, but even more basic

```
(base) mathause@poisson:~$ python  
Python 3.10.13 | packaged by conda  
Type "help", "copyright", "credits"
```

```
>>> a = 5  
>>> b = 7  
>>> a  
5  
>>> print(b)  
7
```

```
(base) mathause@poisson:~$ ipython  
Python 3.10.13 | packaged by conda-forge | (main, Dec 23 2023, 15:36:39) [GCC 12.3.0]  
Type 'copyright', 'credits' or 'license' for more information  
IPython 8.22.2 -- An enhanced Interactive Python. Type '?' for help.
```

```
In [1]: a = 5
```

```
In [2]: b = 7
```

```
In [3]: b
```

```
Out[3]: 7
```

```
In [4]: print(a + b)  
12
```

```
In [5]: █
```

The screenshot displays the JupyterLab web interface. The central pane shows a notebook titled 'Lorenz.ipynb' with the following content:

Introduction

We explore the Lorenz system of differential equations:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy \end{aligned}$$

Let's change  $(\sigma, \beta, \rho)$  with ipywidgets and examine the trajectories.

[2]:

```
sigma 12.40  
beta 2.63  
rho 28.00
```

The plot shows a Lorenz attractor with a color gradient from blue to red. Below the plot is a terminal window with the command 'Sepal.Length' and a 'Click to add a cell.' button.

On the right side, the 'Settings' panel is open, showing the 'CodeMirror' settings. The 'Default editor configuration' section is visible, including options for 'Auto Closing Brackets', 'Code Folding', 'Cursor blinking rate' (set to 1200), and 'Indentation unit' (set to 4).

The left sidebar shows the 'Extension Manager' with a list of installed and discoverable extensions, including 'jupyter-widgets', 'jupyterlab-manager', 'jupyterlab-pygments', 'codex-chat-notebook', 'databeantalk-theme', 'datalayer', 'dataplata-lab', 'digautoprofiler', and 'diginlineprofiler'.