Epistemic Programming - Exploring your own environment

Description of the activity:	The activity introduces the didactic programming concept of Epistemic Programming as well as an exemplary teaching project about collecting and analyzing environmental data with Arduinos and Jupyter Notebooks
Target group(s):	(Future) Secondary school teachers in STEAM-subjects
Keywords:	Insight-driven Programming, Data Exploration, Local Environment, Self-Expression, Jupyter Notebooks
Duration of activity:	5 hours
Description of activity environment and materials needed:	• Powerpoint • Jupyter Notebook from the Material-Folder • Anaconda as a software to open the Jupyter-Notebook-Environment (https://anaconda.org)
Contact	Sven Hüsing, Paderborn University sven.huesing@upb.de

Funded by the Erasmus+ Programme of the European Union

NACIONALINĖ ŠVIETIMO AGENTŪRA

The Development of STEAM education is co-funded by the Erasmus+ Programme of the European Union (Grant Agreement No. 2020-1-LT01-KA201-077935). The content of this document is the sole responsibility of the organizer and does not represent the opinion of the European Commission (EC), nor is the EC responsible for any use that might be made of information contained.

DOSE
DEVELOPMENT OF STEAM EDUCATION

Epistemic Programming

Exploring your own environment

Collection and Analysis of Environmental Data with Arduinos and Jupyter Notebooks

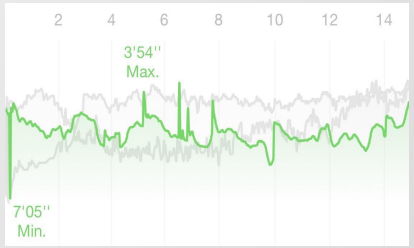
DOSE
DEVELOPMENT OF STEAM EDUCATION

Sources:

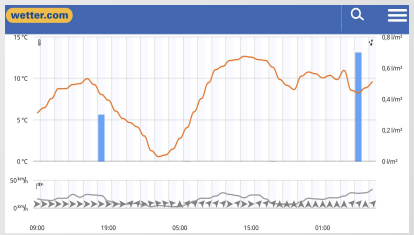
- NikeRunClub app (iPhone)
- wetter.com
- stocks-app (iPhone)
- JupyterNotebooks

Data Collection and Data Analysis in our everyday life

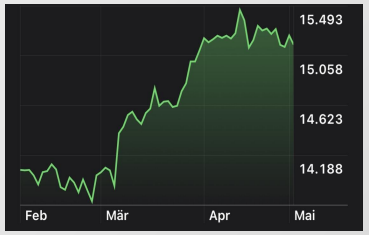
sports



weather



stocks



medicine

“Epistemic Programming”

- Perceive programming as a **means to follow one’s own interests and ideas**
- Reduce the “prejudices” regarding the “nerdiness” of programming by showing how programming can help to **discover one’s environment/one’s own world**
  - Foster an understanding of both one’s own (local) environment (=external insights), as well as the competent usage of digital artefacts for this purpose (=internal insights) (see Schulte, 2013)
  - Communicate the **intertwined process of programming and knowledge acquisition**
- Focus on:
  - Insights one can gain through the programming process
  - Making the (programming) process itself perceivable, comprehensible and reproducible

Idea: Gaining knowledge through “Epistemic Programming”

Why should one make the programming process perceivable??

- For oneself:
  - “Strengthening” of specific programming and project processes
  - Comprehensible expression of own thoughts and ideas
  - Later: Being able to reproduce the programming process (maybe slightly adapted)

- For others:
  - Communicate aspects of programming in an understandable way
  - Adapt or expand the programmed artefact based on the representation and documentation of the programming process according to own interests
    - Educate others so they can themselves engage in programming projects (maybe with a slightly different goal)

Idea: Gaining knowledge through “Epistemic Programming”

„How can you record insights and the (programming) process at the same time??”

Reproducible Research in the Context of Data Analysis

Goal: Comprehensible and reproducible programming projects

Here: Focus on (data-)analyses

- Document data analyses in such a way, they can be repeated with the same or similar data in order to get comparable results
- Connecting results and explanatory text
- Making the process comprehensible (by documenting the individual steps)

McNamara (2019) and Sandve et al. (2013)

Computational Essays – A possibility to implement reproducible research

Idea:

- „digital essay“ which contains small visualisations of the process and the results

Goal: Communication of an Idea or of Insights, Support for a hypothesis etc.

- Computational Essays report about the knowledge acquisition process in a kind of “dialogue” between its author and the digital artefact
- Viewers can interact with programs/visualisations
- Through the interactions with computational essays, its “readers” can explore the process of knowledge acquisition and might change their role from reader to programmer by tinkering with the code in the Computational Essays

B Odden & Malthe-Sørenssen (2021), DiSessa (2000) and Wolfram (2017)

Computational Essays with Jupyter Notebooks

Jupyter Notebooks – a possibility to prepare a programming environment in which students can perform data analyses (also see the concept of Worked Examples (Atkinson et al., 2000; Caspersen & Bennedsen, 2007; Merrienboer & Krammer, 1987))

```
In [9]: df.iplot(kind='box')
```


```
In [10]: df['temperature'].iplot(xTitle = 'Zeit', yTitle = 'Temperatur in °C')
```


```
In [27]: #Mittelwert
print("Mittelwert: "+ str(round(df['value'].mean(),2)) + " °C")
#Standardabweichung
print("Standardabweichung: "+ str(round(df['value'].std(),2)) + " °C")
#unteres Quartil
print("unteres Quartil: "+ str(round(df['value'].quantile(q=0.25),2)) + " °C")
#oberes Quartil
print("oberes Quartil: "+ str(round(df['value'].quantile(q=0.75),2)) + " °C")
```

```
Mittelwert: 18.5 °C
Standardabweichung: 0.53 °C
unteres Quartil: 18.47 °C
oberes Quartil: 18.74 °C
```

Granger & Perez (2021)

Computational Essays with Jupyter Notebooks

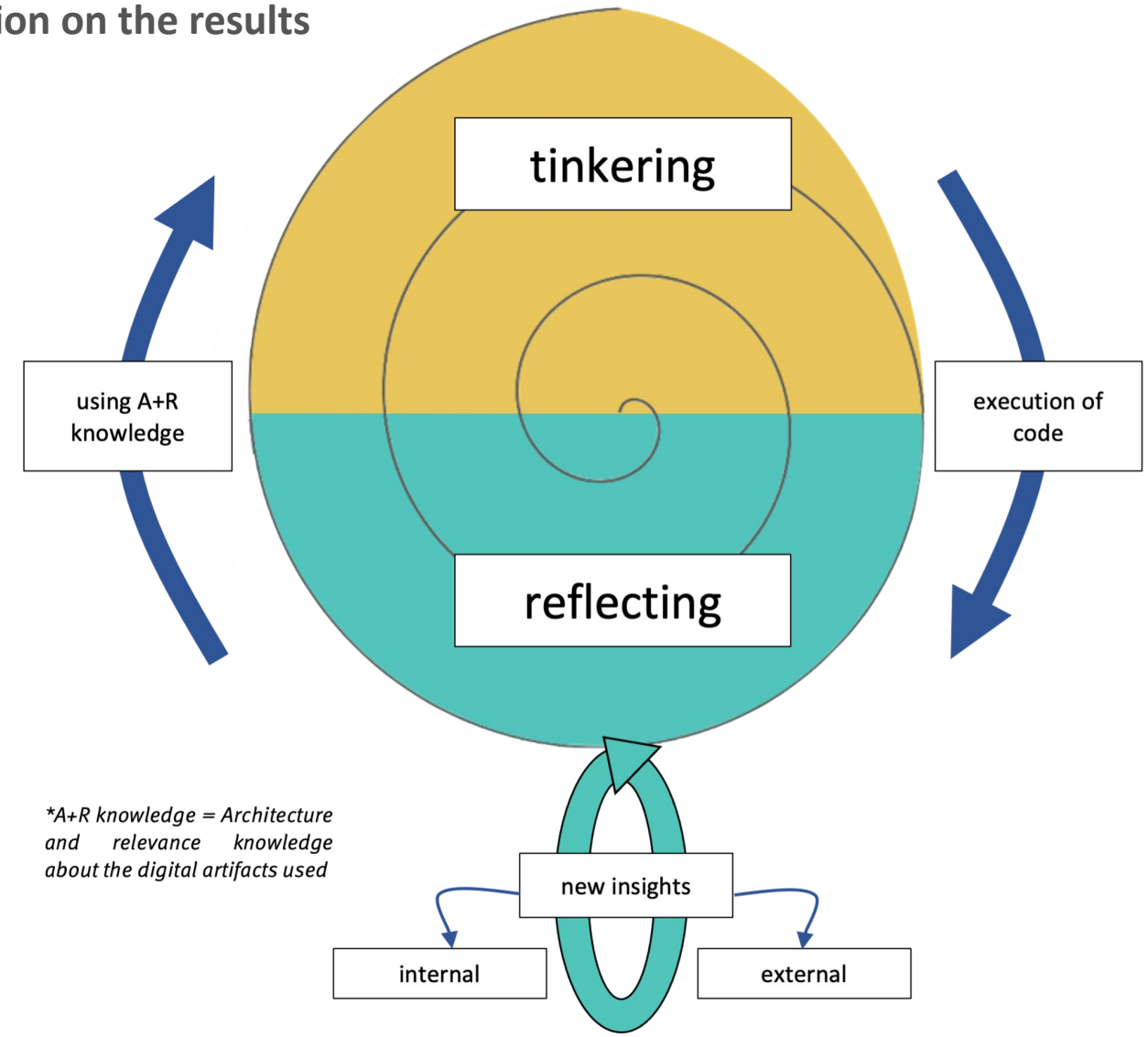
Computational Essays in Jupyter Notebooks have four different cell types:

- Technical-Code-Cells (provide the “invisible” prerequisites for the data analyses, e.g. read the dataset)
- Output-Code-Cells (generate output (e.g. visualisations), readers/viewers can interact with)
- Explaining-Markdown-Cells (explain the code and the process)
- Interpreting-Markdown-Cells (interpret/comment the results)

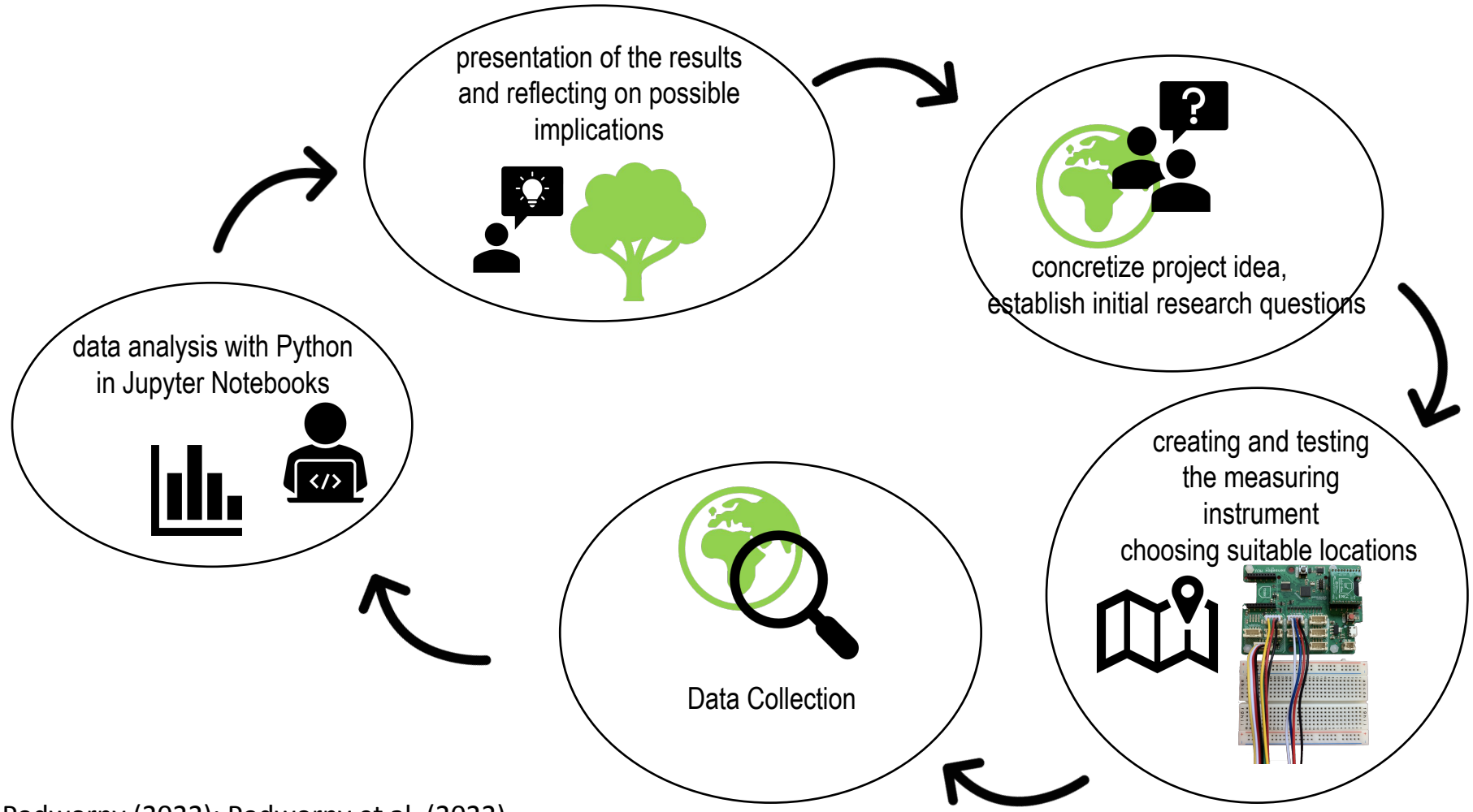
Intertwined Knowledge- and Programming-Process

spiral-like interplay of tinkering and reflection on the results

1. Tinkering (with different programming snippets)
2. Execution of the Code
3. Reflection on the programming results
   - new internal and external insights can emerge:
     - *external:*
       about the explored fields of interest
     - *internal:*
       the adequate use/combination/adaptation of digital artefacts for this exploration and knowledge processes (Schulte, 2013)
4. using these insights & architecture- and relevance-knowledge about the interaction of actors within the cognitive system (de Ridder, 2007; Kroes & Meijers, 2006; Schulte & Budde, 2018; Hollan et al., 2000)
5. identify sections in the code, where it can be changed in order to improve the programming results regarding the personal interests
6. Based on this: tinkering with the code, which means trying different methods, libraries or programming snippets, before executing the adapted code again



Example: Environmental Analysis through Epistemic Programming

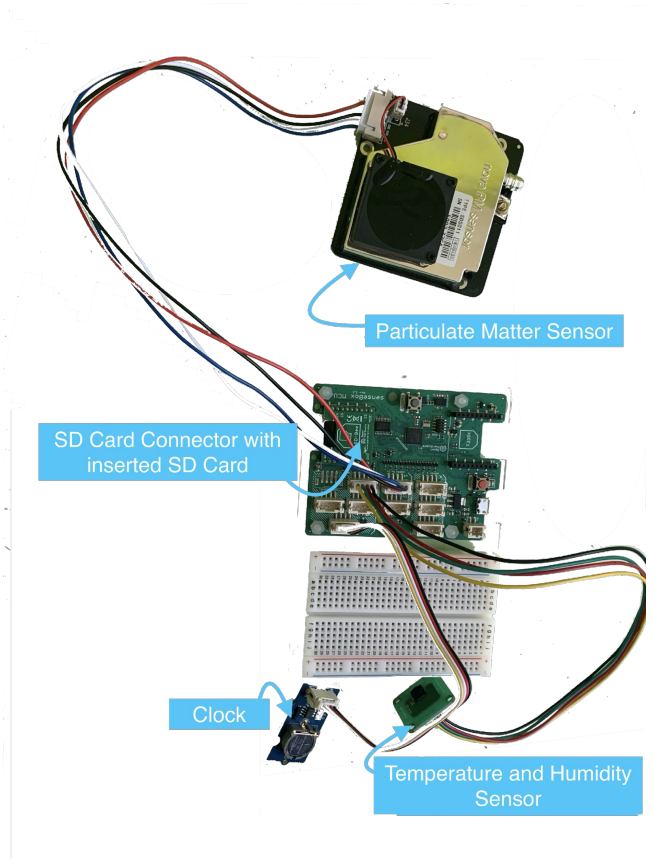


Hüsing & Podworny (2022); Podworny et al. (2022)

Project Example: Fine dust pollution

(Measuring Instrument: Sensebox)

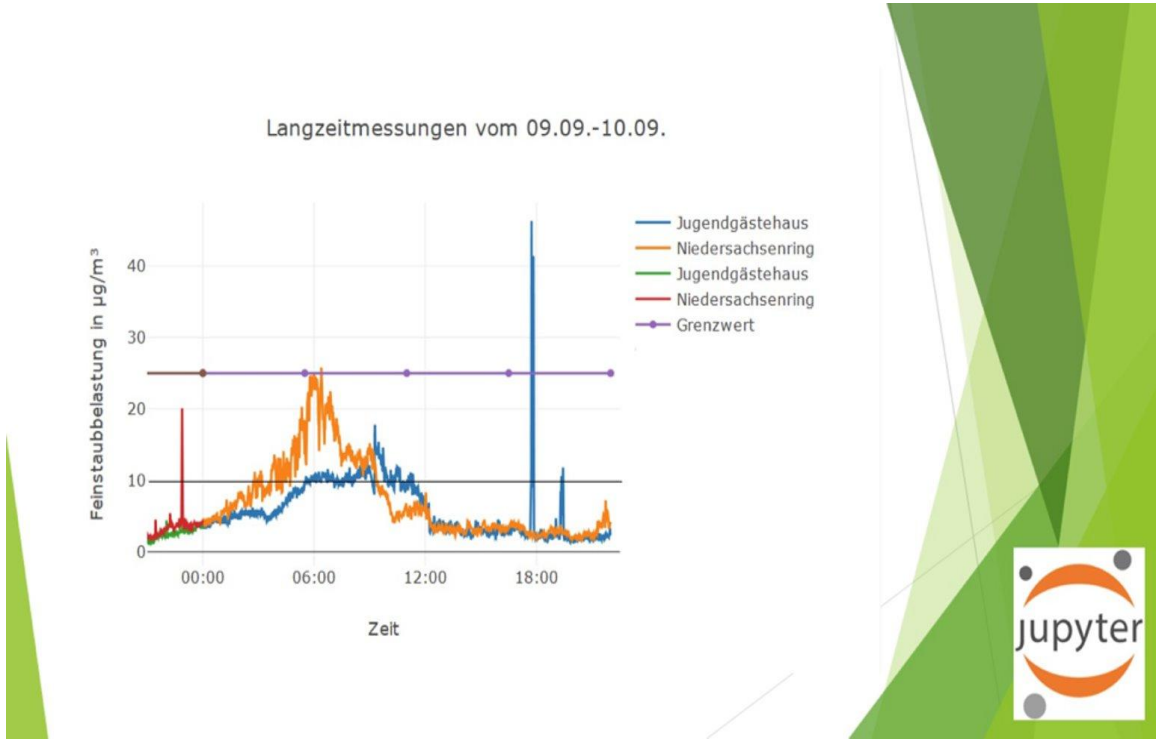
- Programming the SenseBox:
- Arduino-Software
- With Blockly for the Sensebox
  - block-based programming tool
  - Especially for students without programming-experience
  - Link: >click me<



Data analysis with Python in Jupyter Notebooks

The following steps are executed by the students …

- importing libraries, reading in data and displaying data
- data cleansing
- evaluation and visualizing data
  - developing research questions
  - finding suitable visualizations
  - trying things out
  - “code-tuning”
  - Documenting
    - in the Jupyter Notebook itself
    - creation of a Computational Essay showing the code,
      the programming results,
      the interpretation and description of both



possible result (fine dust pollution and legal limit)

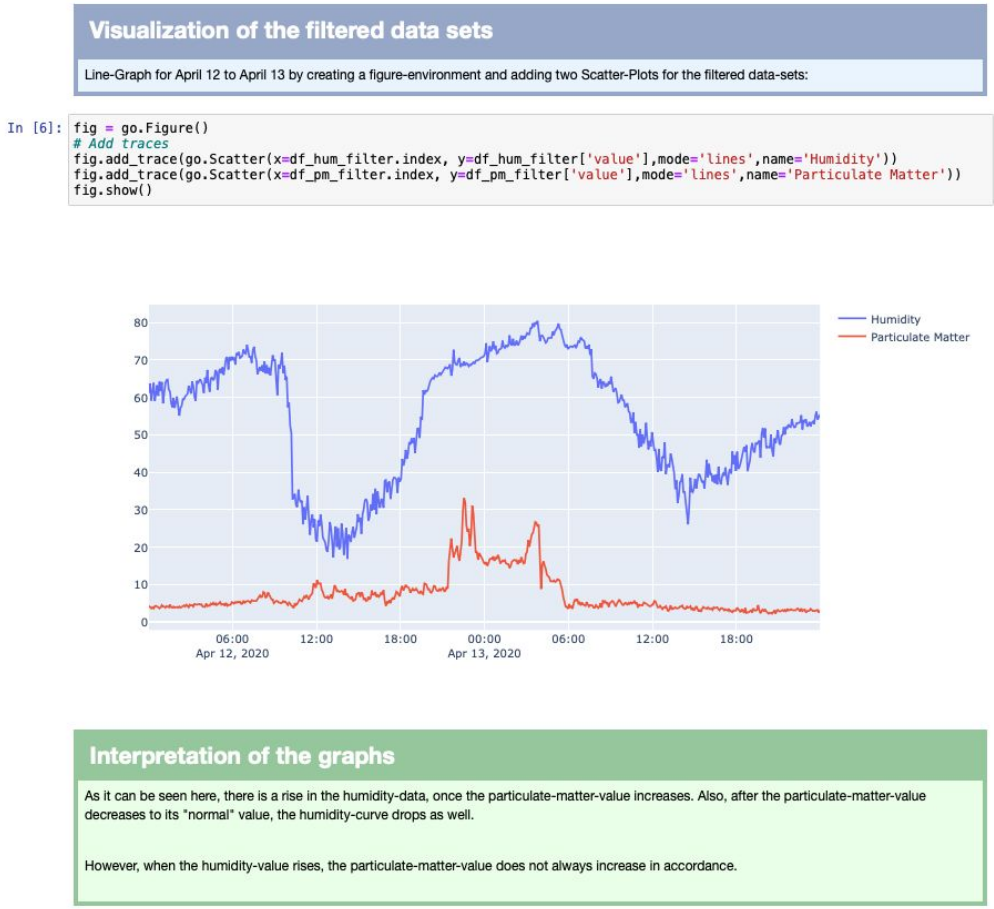
Data analysis with Python in Jupyter Notebooks

… with the help of already prepared Jupyter Notebooks (= Worked Example)

- to support the individual programming steps
- by prescribing code that students can use/combine/adapt
- to show the students how a Computational Essay can be structured and what needs to be included

(Atkinson et al., 2000; Caspersen & Bennedsen, 2007; Merrienboer & Krammer, 1987)

- see PDP_Epistemic Programming_Worked_Example.ipynb



Hands-On-Activity: Environmental Analysis

… with the help of the prepared Jupyter Notebook (Worked Example) in the material-folder

- Build groups of 5 and examine the Jupyter Notebook from the material-folder.

- Create an own Computational Essay for your Environmental Analysis by adapting and combining Code from the Worked Example and by adding Markdown-Cells, in which you should document your code and interpret your results.

- You do not have to use all code cells from the examined Jupyter Notebook, but you can come up with your own evaluation including your own visualizations. Your Computational Essay should contain at least two visualizations including documentation/interpretation.

In summary:
“Epistemic Programming”

- Goal: Programming in order to express oneself, get a differentiated perspective on the world and thus be able to act in a self-determined way.

Regarding Data-Driven Programming:

- Data science with the goal of deriving new insights by analyzing data;
- Gaining knowledge through data as a digital skill or digital competency
  - Highlights the role of digital artefacts and their ubiquity in data mining, collection, and storage, as well as analysis and interpretation
  - clarifies the role of digital data and the ability to gain insights on your own with new approaches and tools

This approach emphasizes:

- Digitization changes the world - If I program epistemically, then I can change the world (in small ways)

A few more project ideas

- Air pressure and the correlations with the weather
- Development of national/regional Covid-19 cases (or any other disease)
- Success from sports team and its money spent
- …

What Ideas do you have?

Reflection

Discuss the following questions:

- What have you learnt during the activity?
- Did you use mathematics? When? Examples?
- What did you do very well? Why?
- What went wrong? Why?
- What will you do differently next time?

Literature

- Atkinson, R. K., Derry, S. J., Renkl, A., & Wortham, D. (2000). Learning from Examples: Instructional Principles from the Worked Examples Research. Review of Educational Research, 70(2), 181–214.
- Caspersen, M. E., & Bennedsen, J. (2007). Instructional Design of a Programming Course: A Learning Theoretic Approach. Proceedings of the Third International Workshop on Computing Education Research, 111–122.
- DiSessa, A. A. (2000). Changing minds: Computers, learning, and literacy. MIT Press.
- de Ridder, G.J. (2007). Reconstructing Design, Explaining Artifacts: Philosophical Reflections on the Design and Explanation of Technical Artifacts. Ph.D. Dissertation. Vrije Universiteit Amsterdam.
- Granger, B. E., & Perez, F. (2021). Jupyter: Thinking and Storytelling With Code and Data. Computing in Science & Engineering, 23(2), 7–14.
- Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed cognition: Toward a new foundation for human-computer interaction research. ACM Transactions on Computer-Human Interaction, 7(2), 174–196.
- Hüsing, S., & Podworny, S. (2022). Computational Essays as an Approach for Reproducible Data Analysis in lower Secondary School. Proceedings of the IASE 2021 Satellite Conference. IASE 2021 Satellite Conference: Statistics Education in the Era of Data Science.
- Kroes, P., & Meijers, A. (2006). The dual nature of technical artefacts. Studies in History and Philosophy of Science Part A, 37(1), 1–4.
- McNamara, A. (2019). Key Attributes of a Modern Statistical Computing Tool. The American Statistician, 73(4), 375–384.
- Merrienboer, J. J. G., & Krammer, H. P. M. (1987). Instructional strategies and tactics for the design of introductory computer programming courses in high school. Instructional Science, 16(3), 251–285.
- Odden, T. O. B., & Malthe-Sørenssen, A. (2021). Using computational essays to scaffold professional physics practice. European Journal of Physics, 42(1), 015701.
- Podworny, S., Hüsing, S., & Schulte, C. (2022). A place for a data science project in school: Between statistics and epistemic programming. Statistics Education Research Journal, 21(2), 6.
- Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten Simple Rules for Reproducible Computational Research. PLoS Computational Biology, 9(10), e1003285.
- Schulte, C. (2013). Reflections on the Role of Programming in Primary and Secondary Computing Education. Proceedings of the 8th Workshop in Primary and Secondary Computing Education, 17–24.
- Schulte, C., & Budde, L. (2018). A Framework for Computing Education: Hybrid Interaction System: The need for a bigger picture in computing education. 18th Koli Calling International Conference on Computing Education Research (Koli Calling ’18), 18, 10.
- Wolfram, S. (2017, November 14). What Is a Computational Essay? Stephan Wolfram Writings. https://writings.stephenwolfram.com/2017/11/what-is-a-computational-essay/