**Exploration of our environment through epistemic programming**

**Epistemic programming as a means of data analysis and exploration of personally meaningful contexts using the example of environmental data**

Note: *This document has been automatically translated using DeepL.com. As a result, some content may not be fully accurate or correctly displayed. Additionally, not all Jupyter Notebooks in the learning environment (*[*https://go.upb.de/ep-umweltanalyse*](https://go.upb.de/ep-umweltanalyse)*) are available in English. All translated Jupyter Notebooks can be found in the subfolder “English”.*

**Information about the teaching module:**

|  |  |
| --- | --- |
| Title: | **Exploration of our environment through epistemic programming - Epistemic programming as a means of data analysis and exploration of personally meaningful contexts using the example of environmental data** |
| Target group: | Computer science courses and interdisciplinary (GeoMINT) courses in grades 8 to 10 (grammar school, comprehensive school) |
| Content areas according to the North Rhine-Westphalian CS curriculum | "Information and Data", "Algorithms", "Computer Science, People and Society", "Computer Science Systems" |
| Time scope: | approx. 16 lessons of 45 minutes each (without own data collection)  approx. 20 lessons of 45 minutes each (with own data collection) |
| Tools: | Data collection: Arduino or Sensebox or other measurement options (optional) Data evaluation: Jupyter notebooks and Python |

# Brief overview

Programming is omnipresent in many different disciplines these days and is often even referred to as the fourth "basic skill" alongside "reading", "writing" and "arithmetic". While programming quickly brings to mind software development processes, programming can also be a means of exploring personally relevant topics or questions, which relate in particular to aspects outside of computer science.

In this teaching module, students are immersed in this knowledge-centered programming practice (also known as epistemic programming) through the context of environmental data and environmental analysis to learn about programming as a means of exploring personal or societal interests in order to be empowered to use programming as such. The teaching module focuses on data-driven environmental analysis as an example. In this context, students can formulate their own initial (research) question, which they then further refine in a heuristic knowledge and programming process or summarize into a concrete research interest, which they pursue by collecting (their own) environmental data and analyzing it. The students record their findings and their programming process during the implementation of their project in a kind of "research diary" - a so-called computational essay - which can be shared with other students or other interested persons at the end of the project.

The teaching module itself is modular in the sense that the data collection part can be designed variably. Students can either collect environmental data independently using a self-developed measuring instrument or collect the data from various data sources. Additional data or data sources (such as weather data from the NASA Langley Research Center (LaRC) POWER project, which is funded by the NASA Earth Science/Applied Science Program (<https://power.larc.nasa.gov/data-access-viewer/>)) can also be used, which may not be listed here.

The teaching module is divided into a total of 5 parts, which are briefly described below. The teaching module should be completed by the pupils in small groups (ideally 3-5 pupils per group).

1. **Development of initial questions/research interests**

At the beginning of the teaching module, the students develop their own research question in the context of the overarching topic of environmental analysis by first learning about criteria for good (statistical) research questions and then working on several questions in small groups, receiving feedback and suggestions for revision from other groups. In order to gain ideas for suitable research questions, the students first examine existing studies and develop their own/related questions based on them. In addition, the teacher can point out possible research questions that may have been investigated in previous lessons. Examples could be: "How well do different textiles (cotton/polyester) protect against UV radiation?" or "How does particulate matter pollution behave in different places (e.g. on a main road compared to a green area)?"

At the end of this phase, each group decides on its own initial research question that it would like to pursue as part of its epistemic programming project

1. **Collect data**

In a second phase, the students collect time series data[[1]](#footnote-2) for their own project. They can either

1. **Collect data yourself using a measuring instrument**

*Note: In this teaching module, we show data collection using the Sensebox as an example. However, there are also many other measurement tools available (e.g. Arduinos or "ready-made" measurement sensors).*

1. **Collect and collate data from various sources**

In this phase, it is important that the students compile data that is suitable for their own project. To this end, students should always keep their initial research question in mind during this phase and critically question how the data can be helpful in answering it. Of course, additional data can also be collected in a second survey.

1. **Data exploration and interwoven knowledge and programming process**

In this phase, the collected data is analyzed with regard to the research interest or research question of the respective student group. The students develop a computational essay to carry out the data analysis using the Python programming language in a Jupyter notebook and at the same time document their programming and knowledge process and, in particular, record their findings (more detailed information on computational essays can be found in the section ,2.39.1.1 and9.4.4 ). They are supported by prepared Python libraries, which enable them to directly call up existing methods for data analysis, and by a so-called worked example (a prepared and documented example of an environmental data analysis, see also sections ,2.49.1.2 and9.4.5 ), which the groups can use for orientation both with regard to the process and with regard to specific program code.

1. **Presentation and exchange of computational essays**

In this phase, the groups share their finished computational essays with each other in order to be able to understand the programming and cognitive processes of the other student groups on the one hand and to exchange feedback and further ideas on the other.

1. **Reflection**

Finally, the students reflect on the programming approach of (data-driven) epistemic programming used in this teaching module. In particular, they develop ideas and perspectives for their own projects in which they can use programming as a means of exploring personally or socially relevant issues

**Content of the document:**

[1 Brief overview 1](#_Toc191131699)

[2 Epistemic programming 3](#_Toc191131700)

[3 Core didactic idea: Teaching epistemic programming as a means of gaining knowledge in the context of data exploration 5](#_Toc191131701)

[4 Learning requirements 6](#_Toc191131702)

[5 Goals 6](#_Toc191131703)

[6 Key questions 7](#_Toc191131704)

[7 Summary overview of the teaching module 9](#_Toc191131705)

[8 Overview of the teaching process 11](#_Toc191131706)

[9 Descriptions of the materials and further background information 32](#_Toc191131707)

[10 Literature 40](#_Toc191131708)

# Epistemic programming

## What does epistemic programming mean?

Nowadays, programming plays an important role not only in terms of software development. Programming can also be used in many areas outside of computer science to gain insights into domain-specific issues. In this way, socially or personally relevant issues can be explored and answers to specific (research) questions can be found. A key area of application for epistemic programming projects is the exploration or evaluation of data in order to gain insights into personally relevant facts from various domains from which the data may originate. In this sense, programming can be used as a tool to understand one's own environment on the basis of data or to be able to pursue one's own interests - to be mature. From this perspective, this programming practice also has a general educational role with regard to our digitalized and data-infused world. With the help of epistemic programming, learners should be able to assume epistemic agency, i.e. influence knowledge and knowledge processes independently and assume cognitive control and responsibility for their own learning (Miller et al., 2018; Odden et al., 2023 , see also the glossary in the section 9.4.3 ).

## Epistemic programming process

The programming process in the context of an epistemic programming project consists of an interaction between the person programming and the computer or the programmed digital artifact. The course of the interaction is cyclical, whereby new insights can be gained in each run, which in turn can influence and possibly change the interest in knowledge. Ideas for impulses for action then arise from the adapted interest in knowledge, on the basis of which the digital artifact is then manipulated in the form of instructions or program code. The processing of these instructions then leads to new (programming) results, which in turn are interpreted into insights.

It is important to note at this point that this cyclical approach in the individual phases does not always have to be followed strictly in reality; sometimes there are also loops or omissions of these individual intermediate products or actions. However, this model provides a good basis for classifying and understanding the processes involved in epistemic programming projects, which are difficult to standardize due to the diversity of the programmer's interests. The decisive aspect here is that knowledge does not arise directly from the person, but is only possible through mutual interaction with the digital artifact.

## Computational Essays as Products of Epistemic Programming Processes

In order to ensure the traceability of the resulting interwoven cognitive and programming process, a suitable documentation medium is required. It should be possible to record all intermediate products of the interaction between the digital artifact and the human being - i.e. findings, cognitive interests, intentions, instructions/program code and (programming) results. Computational essays, in which all these aspects can be recorded, are ideal for this purpose. A computational essay can be described as a formulated essay which, in addition to textual descriptions, interpretations and explanations, also contains manipulable program code that can be executed directly in the medium. This makes it possible to record the intermediate products described above from the individual cycles of the epistemic programming process and also to interact with them directly

## Use of worked examples as support for epistemic programming

As already mentioned, epistemic programming should enable students to experience how they can explore personally or socially relevant issues through programming. In this context, it is therefore of great importance to make this approach accessible to students and programming innovators in general and to provide help in overcoming any hurdles in the programming process. A key idea in this regard is the use of so-called worked examples. In the context of epistemic programming, a worked example is an elaborated and documented programming example in which processes and methods are applied that could be useful for the students' programming project. A computational essay can be used as a form of worked example to record the programming process and the individual steps in particular. At the same time, ready-made programming methods (libraries) in particular can be introduced/explained in the worked example in order to make them directly usable for the students. By using such a worked example, students can thus orient themselves on the programming process or the individual steps from the worked example and at the same time use ready-made program code and adapt it according to their own interest in knowledge. A worked example can thus represent a blueprint for the students' programming processes.

# Core didactic idea: Teaching epistemic programming as a means of gaining knowledge in the context of data exploration

The epistemic programming approach described here represents a knowledge-centered perspective on programming that is to be conveyed to the students in this teaching module. In this way, students should experience programming as a tool that can be used in everyday life to explore personally or socially relevant issues (data-driven). In this respect, students learn how to carry out data exploration projects using programming and how they can pursue their own research interests in this way. On the one hand, this relates to the collection and evaluation of time series data (= data that is repeatedly collected in a certain measurement interval) according to their own goals, but also to the development of computational essays for the comprehensible and reproducible documentation of the knowledge and programming process. In particular, students learn about individual steps in data analysis (such as filtering, creating visualizations or calculating certain statistical parameters) and, in this context, how information can be systematically obtained from data.

The practice of (data-driven) epistemic programming takes place in this teaching module by carrying out their own project in the context of environmental issues. In doing so, the students develop their own epistemic interests, which they pursue as part of their project. In this context, they learn about the process of epistemic programming described above and how they can gain new insights through this interaction with the digital artifact.

In the first part of the teaching module, students learn about criteria for good statistical questions and, in particular, how these can be developed by developing their own questions using a list of criteria and at the same time giving feedback on their questions to other groups of students.

In the second part, the students learn how they can collect suitable data based on their final question or collect it themselves. They can either learn how Arduinos/senseboxes can be used to collect environmental data by programming and connecting appropriate sensors or they can get to know various platforms for (environmental) data and how data can be systematically selected and downloaded here in order to make it usable for their own data evaluation. The systematic approach to searching for data is taught so that students can pursue their own research interests. Alternatively, this second part can also be omitted and specific data can be accessed directly via a Jupyter notebook (name.ipynb) at any location and at any time since 2001. In this case, the data can be loaded directly via a Jupyter notebook and also analyzed in this notebook.

In the third part, the students carry out data analysis in the context of their epistemic programming project in a Jupyter notebook using the Python programming language. In doing so, they learn various steps and actions to read in, process, filter, visualize and otherwise evaluate the collected data. At the same time, by creating their computational essay, they learn how to document their programming and knowledge process in a comprehensible and reproducible way. In order to implement their own project, students acquire basic Python programming skills. This refers in particular to program reading skills, which are required when using the worked examples in order to be able to understand the existing code in the worked example or from the libraries used there and then make it usable for their own project. Basic Python constructs are also taught (especially as part of a short basic Python course), which the students can use in their data analysis.

In the fourth part, the individual groups of students exchange information about the computational essays they have developed so that they can familiarize themselves with the other computational essays and the data exploration carried out in them and understand and evaluate them. Here, too, the students' program reading skills are promoted in particular. They also learn about other examples of epistemic programming projects through the exchange with the other groups.

In the fifth part, the students reflect on the perspective of epistemic programming and the role of data analysis in the context of exploring personal interests. In this phase, the interaction between the person programming and the digital artefact should also be worked out and discussed, with a particular focus on analyzing the roles of these two actors. In this phase, the students also develop further ideas and questions that can be addressed in the context of data-driven epistemic programming projects.

# Learning requirements

This teaching module does not require any special prior knowledge on the part of the students. Basic experience in using a computer and in particular in dealing with files (copying, cutting, deleting, etc.) as well as basic knowledge in the interpretation of statistical key figures (mean value, standard deviation, possibly also correlation) and diagrams (being able to read diagrams and derive information from them) are desirable so that students can manage their collected data or arrive at their own findings through their data evaluations. Initial experience with the Python programming language and Jupyter Notebooks is also helpful, although the necessary basic knowledge is also taught as part of this module.

# Goals

## Overarching objectives of this teaching module:

1. (broad objective) Students learn about data-driven epistemic programming as a means of data exploration in the context of a personally relevant environmental issue.
2. (Detailed objectives) The students carry out a data-driven epistemic programming project as epistemic agents[[2]](#footnote-3) in the context of their own environment-related research question by collecting and evaluating (their own) data, interpreting the evaluations and thus developing findings with regard to a personally or socially relevant question

## Objectives of the subsections of the teaching module

* **Part 1: Development of initial questions/research interests**
  + Students learn about the exploration of data as a method for objectively answering personally or socially relevant questions.
  + The students know criteria for good and answerable statistical questions.
  + Students develop statistical questions for a data-driven epistemic programming project by following the previously learned criteria.
  + The students evaluate research questions developed by other student groups and provide criteria-based feedback and develop suggestions for improvement.
  + The students identify variables to be investigated that are contained in the research questions they have developed.
* **Part 2: Collecting data**

1. **Collect data yourself using a measuring instrument**

* The students develop a plan for collecting data that allows them to explore their previously formulated research question.
* The students develop a measuring instrument to collect data by assembling a Sensebox/Arduino from suitable components and sensors.
* The students design and implement a program for data collection with the Sensebox/an Arduino.

1. **Collect and collate data from various sources**
   * The students develop a plan for collecting data that allows them to explore their previously formulated research question.
   * Students identify suitable data sources and generate/load data sets with regard to their research question

* **Part 3: Data exploration and interwoven knowledge and programming process**
  + The students know important methods for reading, processing, filtering, visualizing and evaluating time series data as well as basic Python commands in order to apply or combine these methods with regard to their own project.
  + The students use the worked example provided by adapting and extending code from it and adopting the sequencing - adapted for their own project.
  + The students carry out the essential steps "Importing data", "Preparing data", "Filtering data", "Visualizing and evaluating data" in the data exploration process.
    - Data preparation consists at least of assigning the correct data types and setting the index.
    - When visualizing and evaluating the data, students can choose between different methods and approaches, which are shown in the worked example. In particular, these include visualization as scatter or box plots (with multiple plots), the calculation of key statistical parameters (min/max/mean), the grouping of data sets and the calculation of correlations.
  + The students know the characteristics and uses of a computational essay.
  + The students communicate their cognitive and programming process in a comprehensible and reproducible way by gradually developing a computational essay and combining code cells, code output cells and text cells.
  + The students carry out epistemic programming processes by deriving new action steps from their findings in a spiral, implementing them in the form of program code and having them executed by the computer.
* **Part 4: Presentation and exchange of computational essays**
  + The students reproduce the cognitive and programming process documented in the computational essays of the other groups.
  + The students evaluate the documentation and program code of the other groups and develop well-founded adaptation and extension proposals.
* **Part 5: Reflection**
  + Students evaluate the role of programming and data in their cognitive processes.
  + The students describe and discuss the interaction between the programmer and the digital artifact in epistemic programming processes.
  + The students discuss the perspective of data-driven epistemic programming and the general educational role of programming with regard to knowledge and exploration processes.
  + Students identify suitable applications for epistemic programming programming projects and develop ideas for their own and personally or socially relevant data-driven epistemic programming projects.

# Key questions

* **Part 1: Development of initial questions/research interests**
  + How can good and answerable statistical questions be formulated for data-driven epistemic programming projects?
* **Part 2: Collecting data**

1. **Collect data yourself using a measuring instrument**

* How can a Sensebox/Arduino be assembled and programmed in order to collect suitable data for your own research question?

1. **Collect and collate data from various sources**
   * How can suitable data be found from existing data sources to suit your own research question?

* **Part 3: Data exploration and interwoven knowledge and programming process**
  + How can different steps in the data exploration process be executed in Python and Jupyter Notebooks (e.g. read in data, define data types, create different visualizations, ...)?
  + How can worked examples be used to support data-driven epistemic programming projects?
  + What aspects must a computational essay contain in order to document epistemic programming processes in a comprehensible and reproducible way?
* **Part 4: Presentation and exchange of computational essays**
  + What are the criteria for comprehensible and reproducible computational essays?
  + How can computational essays be read, understood and adapted/extended?
* **Part 5: Reflection**
  + What role do the programmer and the digital artifact play in epistemic programming processes?
  + What other personally or socially relevant projects are there that can be addressed using data in epistemic programming projects?

# Summary overview of the teaching module

Depending on the learning objectives and the time available, there are various options for carrying out this teaching module. While the focus of the module is on conducting a data exploration, you can choose to collect your own data beforehand, collect data from various (online) sources or use weather data from the NASA Langley Research Center (LaRC) POWER project, which is funded by the NASA Earth Science/Applied Science Program (<https://power.larc.nasa.gov/data-access-viewer/>). In the latter case, you skip the data collection phase and therefore need slightly less time for the entire teaching module.

The teaching module can therefore be roughly divided into the following parts:

Part 3:   
Data exploration and interwoven cognition and programming process

Part 4:  
Presentation and exchange of computational essays

Part 5:  
Reflection

Part 1:   
Development of initial questions/research interests

Part 2a   
Collecting data  
Collect data yourself using a measuring instrument

Part   
Collecting data  
Collect and collate data from various sources

Do not collect/collate data and use data from the NASA Langley Research Center (LaRC) POWER project

The following diagram shows the respective activities, key questions and subject content of the individual parts.

Part 1:   
Development of initial questions/research interests

Criteria-based development of an initial research question; exchange and feedback between the groups

How can good and answerable statistical questions be formulated for data-driven epistemic programming projects?

Criteria for answerable statistical questions and their formulation, basic idea of data exploration as a means of gaining knowledge

Part   
Collecting data  
Collect data yourself using a measuring instrument

Planning your own data collection, developing a measuring instrument for data collection, assembling and programming a Sensebox/Arduino using suitable sensors

How can a Sensebox/Arduino be assembled and programmed in order to collect suitable data for your own research question?

Planning data collection with your own measuring station, basic Arduino programming, hardware components of an Arduino measuring station - especially measuring sensors

Part 2b:   
Collecting data  
Collect and collate data from various sources

Planning your own data collection, identifying suitable data sources, downloading appropriate data sets

How can suitable data be found from existing data sources to suit your own research question?

Plan and carry out data collection using suitable data sources, compile various data sets

Part 3:   
Data exploration and interwoven cognition and programming process

Introduction to the medium "computational essay" and the use of a worked example, implementation of your own epistemic programming project by creating a computational essay with the help of a worked example

How can...  
... read, process, filter, visualize and evaluate data in Python and Jupyter Notebooks  
... Worked Examples can be used as support in data-driven epistemic programming projects?  
What aspects must a computational essay contain in order to document epistemic programming processes in a comprehensible and reproducible way?

Computational essays as a medium for documenting cognitive/programming processes in epistemic programming projects, use of worked examples as orientation for own projects, basic Python programming for exploration of time series data incl. use of libraries; importing, processing, filtering, visualizing and evaluating data, interpretation and presentation of results

Part 4:  
Presentation and exchange of computational essays

Giving feedback, implementing adaptations/extensions with the help of feedback,  
Short presentation, then exchange of the computational essays created, exploration of other computational essays, publication

What are the criteria for comprehensible and reproducible computational essays?  
How can computational essays be read, reproduced and adapted/extended?

Criteria for comprehensible and reproducible computational essays, reading computational essays and interactive data reports, (implicit) basic Python programming and use of libraries

Part 5:  
Reflection

Reflection on one's own cognitive and programming process as well as one's own perspective on (knowledge-centered) programming

What role do humans/digital artifacts play in epistemic programming processes?  
What other personally or socially relevant projects are there that can be addressed through data analysis in ep. programming projects?

Analysis of the role of humans and digital artifacts in the programming project, transfer to other possible applications, awareness of the individual perspective on programming as a tool for gaining knowledge

x

x

Key question

Subject content

Activity

# Overview of the teaching process

The following table describes the course of the lesson. The "Content" column contains descriptions of the course of the lesson with didactic comments. The table also lists the objectives and the materials used for each phase of the teaching module.

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Contents** | **Goals** | **Material** |
| **Part 1: Development of initial questions/research interests** | | | |
| **1.1** | **Introduction to the context and problematization:**  In this first phase of the teaching module, the context - environmental data analysis - will be introduced by discussing and highlighting examples of current issues. In particular, the role of data and data analysis will be discussed.  As a stimulus for the discussion, the students can find out about possible research questions and projects in the context of data-based environmental analyses by reading texts on corresponding studies and discussing the results, data used and findings of these studies.  This can be done using the texts linked in the material "Overview of texts and articles on the topic of "Data-based environmental analysis"". Here, the students discuss one of these texts in pairs using the worksheet "Exchange on environment-based study". The results are then shared in plenary, followed by a discussion on the key question "What role does data play in answering environmental questions and what conclusions can be drawn from it in this context?".  Didactic commentary:  In this phase, students should be sensitized to the topic of environmental data exploration. In particular, they should be shown examples of topics or questions on which data exploration can be carried out. In doing so, they should develop initial ideas for their own questions, which they can then implement later in their own project. It is important that the students already identify initial ideas for (research) interests based on the possibilities presented in this phase, which are either similar to the topics of the studies discussed here or may also address completely different environmental aspects.  In order to introduce the topic and the possibilities of data exploration for answering environment-related questions at this point, the following aspects in particular should be "worked towards" in the concluding discussion:   * Data analyses and, in particular, visualizations can be used to gain insights into environmental issues. * It must be ensured that the data are a) "correct" and b) "representative", i.e. a) the data are not subject to any measurement errors and b) the data adequately reflect the situation, in particular not only special cases are considered. * You can verify or expand the results by analyzing your own data.   To work towards these aspects, you could, for example, choose the following introductory question: "What should you pay attention to when carrying out a data exploration on a specific topic?" | * Developing interest and motivation to carry out environmental data exploration * Exploration of the role of data and data analysis/visualization for the assessment of environmental issues/issues * Demonstrate examples of environmental issues that can be addressed/answered through data exploration | Overview of texts and articles on the topic of "Data-based environmental analysis" (Overview\_news\_environmental\_questions)  WS1\_Exchange\_study |
| **1.2** | **Development of a personal question**  In this phase, the students develop a personal question that they would like to answer in their project. They can either use the studies considered in phase 1.1 as a guide and conduct a replication study, for example, or set a completely different environmental focus. With regard to the research question, it is important that the students take into account the possibilities of data collection or the data already available insofar as the research question is to be answered with this data. Suitable data for the research question must therefore be collected/found. Since the subsequent guided data analysis is based on so-called time series data (= data that is collected repeatedly in a certain measurement interval and provided with a time stamp), the students should plan their research question in such a way that they can answer it with time series data.  First, the students get together in small groups (at best 4 students per group) and then work out their own question for their own environmental exploration project using the worksheet "Developing a question".  At the beginning, they are given 6 criteria/questions that they should consider with regard to their question. The student groups then develop one or two initial ideas for a question, for which they should then briefly explain why this is/are important to them. They should also already record which variables need to be collected to answer the question.  Two groups then exchange their worksheets and receive feedback from the other group, whereby the other group checks the question(s) with regard to the criteria described above on the worksheet and also gives the group written feedback. Both groups then revise their own questions based on the feedback from the other group and record these questions on their worksheet.  Didactic commentary:  If the student groups have difficulties finding a research question, they could also use the following initial/rough example questions as a guide:   * How does the particulate matter content in my city differ between busy roads and traffic-calmed zones? * How does the CO2 content in our classroom behave and how does it behave under different ventilation strategies? * How does the weather differ in different places around the world? * How has the weather changed in the last 20 years in my city/at a certain location | * Development of an own research question for the project * Identification of variables or data to be collected to answer the research question * Familiarization with criteria and procedures for developing a statistically verifiable research question | WS2\_Development\_Question |
| **Part 2: Collecting data**  *Note: In this part of the teaching module, you can decide to collect time series data yourself using your own measuring instrument - 2a - or to compile existing time series data from various sources - 2b. It is also possible to read certain data (temperature, humidity, precipitation, wind speed) for any location and any time period since 2001 directly into a Jupyter notebook. In this case, you can continue directly with part 3.1 and let the student groups work in the subfolder "Environment NASA Data" and there in the Jupyter Notebook Computational\_Notebook\_NASA-Data.ipynb after completing the tutorial.* | | | |
| **Part 2a: Collecting data - collecting data yourself using a measuring instrument**  *Note: In the following, the teaching module is described with regard to the use of the Sensebox and, in this respect, how the student groups can collect their own environmental data using the Sensebox. This description focuses on the use of a Sensebox kit consisting of the board, (various) sensors, SD card module, display and possibly other components. An example kit is the SenseboxMini* [*(https://sensebox.de/de/products-mini*](https://sensebox.de/de/products-mini)*). However, you can of course also use other measuring instruments or program them yourself, such as an Arduino* [*(https://docs.arduino.cc/*](https://docs.arduino.cc/)*) or a MicroBit (*[*https://microbit.org/get-started/user-guide/introduction/*](https://microbit.org/get-started/user-guide/introduction/)*).* | | | |
| **2a.1** | **Getting to know the Sensebox as a programmable measuring instrument**  In this phase, the students get to know the Sensebox and develop their first programs using the block programming environment for the Sensebox.  Each group initially receives its own sense box (including sensors and other accessories), which they can later use for data collection. With the help of the flashcards for the Sensebox (in particular the flashcards SB01-SB07, SB12, GI01, GI02, GI03), the groups first learn which components the Sensebox contains, how it can be assembled and how it can be programmed using the block programming environment.  The groups should then assemble their sense box and program it in the Blockly environment in such a way that the current temperature measured by the temperature sensor is shown on the display. An example solution for this can be found in the file "Sol\_Measurement\_Temperature".  Didactic commentary:  The aim of this phase is for the students to be able to explore and get to know the Sensebox freely and to create a first small project at the end so that they can understand how to assemble and program the Sensebox.  Below you will find a few tips on setting up the Blockly environment and transferring the finished Blockly program to the Sensebox. These are also relevant for the subsequent phases. More detailed information can also be found in the Blockly documentation.  *Important for programming in the Blockly environment:*  In the block programming environment, the correct board to be used in the Sensebox must first be selected. This should be communicated to the students once at the beginning of this phase.  *Important for transferring the program code:*  Once the program has been written in the Blockly environment, it still needs to be transferred to the Sensebox. If you are working on a computer, you need to download the program and then "drag" it to the Sensebox. This is explained in the instructions at the following URL: <https://docs.sensebox.de/en/docs/boards/mcus2/mcus2-kompilieren/#kompilieren>  The program created in Blockly can also be transferred to the Sensebox using the tablet. How this works is explained in the following instructions: <https://sensebox.de/en/app> | * Getting to know the Sensebox (and the measuring sensors) as a measuring instrument for environmental data * Getting to know the Blockly programming environment for programming the Sensebox | Flashcards for the Sensebox (Link: <https://sensebox.de/de/lernkarten-mcu>)  Block programming environment Blockly (Link: <https://blockly.sensebox.de/>)  Documentation from Blockly (Link: <https://docs.sensebox.de/category/blockly/>)  Example solution for the first temperature measurement (Sol\_Measurement\_Temperature) |
| **2a.2** | **Development of a first measuring station**  In this phase, the groups develop an initial environmental station with the Sensebox. They can either think about assembling and programming it themselves or use the project description for the "Mini environmental station" as a guide. The aim is for the groups to end up with a measuring station that measures certain values (e.g. temperature data) and saves them in a csv file, indicating the time elapsed since the start of the experiment.  *Note on the "Mini Environmental Station" project description:*  In the finished program of the project description, it is specified that a line is inserted before each line with the current values in the created file, in which it is specified which values are measured - e.g:  Time; temperature; humidity; air pressure  6100; 24.1; 44.2; 1013.25  Time; temperature; humidity; air pressure  6101; 24.4; 42.7; 1015.42  Time; temperature; humidity; air pressure  6103; 24.2; 41.9; 1011.51  ...  For subsequent data evaluation, however, it is sufficient if the "heading" is only recorded once at the top of the file - e.g:  Time; temperature; humidity; air pressure  6100; 24.1; 44.2; 1013.25  6101; 24.4; 42.7; 1015.42  6103; 24.2; 41.9; 1011.51  ...  In addition, in contrast to the project description, the data should be saved in a csv file (and not in a txt file) for later data exploration. A corresponding program that implements these two aspects and is otherwise identical to the one in the project description is the program Mini-Umweltmessstation\_fuer\_Datenexploration.xml. This can be opened in the Blockly environment by clicking on the "Open project" button (top right in the button bar) and then transferred to the Sensebox.  If the BME680 environmental sensor is not available, you can of course also measure other data. In the program Mini-Umweltmessstation\_mitHDC1080Sensor.xml, for example, temperature and humidity data are measured and saved with the HDC1080 sensor.  *Further information when using the senseBox MCU as a board (instead of the senseBox MCU:mini or the senseBox MCU-S2):*  To transfer the data to the SD card, the XBEE slot 2 must be used for the SD card reader.  Didactic commentary:  In this phase, the groups prepare an initial template for programming the Sensebox into a measuring station. The focus here is on ensuring that they can implement the essential functionalities (displaying current data on the display and transferring data to an SD card) so that they can create their own measuring instrument for their own research question based on their results in this phase. The project description of the "Mini Environmental Station" can serve as orientation and support. Of course, they can also contribute other ideas at this stage. | * Development of a first measuring station for measuring and storing environmental data | Block programming environment Blockly (Link: <https://blockly.sensebox.de/>)  Documentation from Blockly (Link: <https://docs.sensebox.de/category/blockly/>)  Project description "Mini Environmental Station" (Link:  <https://sensebox.de/projects/de/2022-12-19-messstation-mini>)  XML files for data collection for later data exploration:   * Mini-Monitoring\_Station * Mini-Monitoring\_Station\_HDC1080sensor |
| **2a.3** | **Programming the Sensebox for data collection**  Based on the measuring stations and programs developed in the previous phase, the student groups develop a measuring instrument tailored to their own research question (from phase 1) in this phase. In order to be able to plan the data collection in a first step, the groups should again refer to AB2 Developing a research question. In phase 1.2, they recorded their final research question on the one hand and also specifically recorded the variables required to answer the research question(s). Depending on the research question, it may be necessary to carry out several data collections (e.g. at different locations).  The groups should record the planning of their data collection on the AB3a\_Planning\_Data\_Collection. They should then use this later to document the data collection process as well as their data exploration process. After completing questions 1-4 of the AB, the groups should imagine their planned data collection on the basis of these. Based on this, they should identify whether there are other groups with similar interests so that data could be exchanged if necessary. This should then be recorded in question 5 on the AB. For example, if several groups want to collect data on a main road, they can exchange their data with each other so that only one sense box has to collect data there. The sense box of the other group can then record other required data at the same time.  After completing the AB, the groups start to create their measuring stations. They can first develop a measuring instrument for the first data collection and then set it up and collect data while they program the measuring instrument for a possible second and/or third data collection.  When the preparations for programming all the measuring stations have been completed and while the individual measuring stations are collecting the data, the students can already start getting to know Python with the help of the Python crash course (see phase 3.1).  Once data collection is complete, the data collected by the Sensebox still needs to be compiled and saved.  Didactic commentary:  In this phase, the groups should independently develop a measuring instrument for collecting data on their research question. They can use the program from the previous phase as an anchor point from which they can continue working. Of course, the groups can also support each other - for example with regard to the use of certain sensors.  *Notes on programming the Sensebox:*   * Depending on the learning objectives   1. the students can program the Sensebox independently in their groups   2. the teacher can provide a "code base" which the students can then expand on   3. the teacher can already provide the complete code so that the students can transfer it to their assembled Sensebox * Information on programming the Sensebox can be found on the documentation website: <https://docs.sensebox.de/en/> * Programming can be carried out either using the block-based language  or with the Arduino IDE (text-based programming).   1. Information about Blockly: <https://docs.sensebox.de/en/docs/boards/mcu/mcu-erster-sketch/>   2. Information about the Arduino IDE: <https://docs.sensebox.de/en/docs/category/arduino/> * It is important that the time is also saved when programming the sense box so that suitable diagrams can be created later in the data exploration in which the time can be entered on the x-axis. As in the previous phase, you can either save the time since the start of the measurement in the csv file or save the current time using an (external) clock. Information on this can be found here: <https://docs.sensebox.de/en/docs/hardware/accessoires/rtc-modul/>  or  <https://edu.books.sensebox.de/de/grundlagen/uhr.html> * The data collected by the Sensebox can either be stored on an SD card - as explained in the previous phase - or sent to the OpenSenseMap via Wifi or the LoRaWan network and downloaded from there later. Information on sending data via Wifi can be found here:  <https://docs.sensebox.de/en/docs/hardware/wifi/> or <https://sensebox.de/projects/de/2024-01-10-iotmesstation_s2> Information on sending via LoRaWan can be found here: <https://docs.sensebox.de/en/docs/hardware/bee/lora-bee/> or <https://sensebox.de/projects/de/2021-02-19-ttnv3> or <https://sensebox.de/projects/de/2019-05-10-lora-osem-tago> or <https://sensebox.de/docs/informatikunterricht_lorawan.pdf> * In general, the Sensebox can again be programmed in the Blockly environment. Alternatively, or based on an initial version of the program, can be programmed in Arduino code if required (if necessary with a prior introduction). This requires the Arduino IDE, which must first be prepared for use with the Sensebox. Information about this can be found here: <https://docs.sensebox.de/en/docs/category/arduino/>   You can then copy the equivalent code to a Blockly program directly from the Blockly environment (on the right) and then further adapt or extend it in the Arduino IDE.  *Instructions for compiling the collected data with the Sensebox:*   * There are various options for how the data collected by the Sensebox is stored. You specify this when programming the Sensebox:   1. on an SD card on the Sensebox as a csv file   2. on the OpenSenseMap  1. If you save the data locally on an SD card, which is attached to the Sensebox via a corresponding module, you can then copy the corresponding csv file from the SD card to the computer. 2. If the data has been saved on the OpenSenseMap (e.g. if you have implemented the transmission via WiFi or LoRaWan), you can download the data of your own station (and possibly other stations as well) via the OpenSenseMap. Information on this can be found here: <https://tutorials.opensensemap.org/analysis/analysis-download/>  * *Important*: At the end of this phase, the students should have saved all relevant data as a csv file so that they can read it into the data exploration environment and explore it in the next phase. | * Development of your own measuring instrument for data collection with regard to your own research question | Blockly (Link: <https://docs.sensebox.de/category/blockly/>)  WS3a\_Planning\_data\_collection |
| **Part 2b: Collecting data - collecting and collating data from various sources** | | | |
| **2b.1** | **Organization of data collection**  In this phase, the groups should structure their data collection process by recording which time series data (= data that is collected repeatedly in a specific measurement interval) they want to search for. They should use the worksheet AB3b\_Planning\_Data Collection (initially only questions 1-4) to record the type, time periods and locations of the data they will use in their data exploration and then present this to the other groups in plenary.  The groups can then consider which other group has similar interests - particularly with regard to data - so that it might be worthwhile sharing data at a later date. This can then be recorded using question 5 on the worksheet. | * Structuring the data collection process | WS3b\_Planning\_data\_collection (page 1) |
| **2b.2** | **Research on various data sources**  In this phase, the groups research suitable data sources for data exploration for their research question recorded on AB2. The sources in the *overview of possible data sources* can be used as initial ideas. It is important that as many possible data sources as possible are first identified in this phase, which are then recorded on the second page of AB3b. The groups can then exchange their sources with each other, which means that all groups may receive further data sources and can add them to the worksheet.  Didactic commentary:  Of course, data sources other than those listed in the overview can also be suggested to the groups. The groups can also use a search engine such as Ecosia or DuckDuckGo to search for datasets. It is recommended that search queries are made in English, as there are many datasets on English-language sites. | * Identification of suitable data sources for your own research question | WS3b\_Planning\_data\_collection (page 2)  Overview of possible data sources: Overview\_Data\_Sources |
| **2b.3** | **Download and save the relevant data**  In this phase, the student groups use the data sources previously identified and recorded on AB3b to download suitable data for data exploration with regard to their research question. Initially, several data sets for the same characteristic (e.g. temperature) can be downloaded. A selection can then be made later during data exploration. In order to record the data source for the respective data (also for later documentation and thus for the transparency of the data exploration process), the groups should record the respective downloaded data sets on the back of AB3b and also record the source (including link and access date).  At the end of this phase, the groups can discuss the downloaded data sets once again and exchange data sets if necessary.  Didactic commentary:  When downloading the data, make sure that it is in csv format or in a format that can be easily converted to csv format. For example, txt files can be converted quite easily by changing the file extension. To briefly check the data after downloading, it is worth briefly opening the data records. The first line should then contain a header in which all variables of the data set are listed. The data should then follow in rows, with each row representing a point in time at which the data contained in the row was recorded. It is important to ensure that a timestamp or a point in time or similar is also recorded. | * Collecting and collating data sets with regard to your own research question | WS3b\_Planning\_data\_collection (page 2) |
| **Part 3: Data exploration and interwoven knowledge and programming process** | | | |
| **3.1** | **Getting to know the programming environment:**  Before the students start their own data exploration processes, it is helpful that they first get to know the programming environment (Jupyter Notebooks) and the programming language (Python) (see also glossary, section9.1 ). To do this, they independently complete an introductory course in Jupyter Notebooks. This course consists of a total of seven Jupyter Notebooks in which the students deal with various essential aspects and constructs (e.g. variables, case distinctions, repetitions, functions, use of DataFrames) of Jupyter Notebooks and the Python programming language.  *Instructions for working in the Jupyter Notebook environment:*  When accessing the Jupyter Notebook environment for the first time (via <https://go.upb.de/ep-umweltanalyse>), you can freely assign a user name and password in the login screen. The edited and, if necessary, newly created Jupyter Notebooks are then saved in the account when you click on the save symbol in the bar in the individual Jupyter Notebooks or on "File -> Save and Checkpoint". After logging in again with the specified user name and password, you can then access your own documents again later  Further information on the structure and content of the Jupyter Notebook environment can be found at9.1.4 .  Didactic commentary:  With the Jupyter Notebooks described here, the students acquire basic knowledge about the use of Jupyter Notebooks and Python as a programming language. The Jupyter Notebooks included are intended to build up basic knowledge to such an extent that the students can use, adapt and combine existing program code from the worked examples in the next phase. The other Juypter Notebooks 5-8 are particularly helpful for in-depth treatment of the Python programming language and are optional in the context of environmental analysis.  is important - especially for phases 3.2 and 3.3 - that the students remember the access data they have created and, if necessary, write it down somewhere, as otherwise they will no longer have access to their work results. They should also make sure that they exchange different work results within their small groups so that work can continue within the group even if one student is absent. To do this, students can download the Jupyter notebooks as an ipynb file via "File -> Download as -> Notebook (ipynb)" in the menu bar. An ipynb file that has already been downloaded can - just like a csv file - be uploaded again via the upload button in the folder view of the Jupyter Notebook environment (e.g. also under a different account). More detailed information on this can be found in the glossary under section .9.1.5 | * Getting to know the Jupyter Notebooks programming environment * Building a basic knowledge of the Python programming language and the Jupyter Notebooks environment | Jupyter Notebook environment  <https://go.upb.de/ep-umweltanalyse>  In it: Jupyter Notebooks in the subfolder Tutorial\_JupyterNotebooks\_Python |
| **3.2** | **Data exploration process**  In this phase, the students carry out their own data explorations in their small groups in order to find answers to the questions they have recorded on AB2. The aim at the end of phase 3.3 is to develop a computational essay (see also sections ,2.39.1.1 and9.4.4 ), i.e. a document that maps the data exploration process and presents the programming results together with the program code and the documentation of the knowledge and programming process. In a Jupyter notebook, this can best be achieved by dividing the program code into individual knowledge and programming steps, which are then represented in individual or possibly several code cells and between which explanations of the code or the current knowledge steps as well as interpretations of the programming results and knowledge can be recorded. A good example of a computational essay are the two worked examples, which can be found in the "Worked Examples" subfolder and are also explained in more detail below in this phase.  While this phase initially focuses on the exploration process itself, phase 3.3 is about finalizing and, if necessary, supplementing your own Jupyter notebook to turn it into a computational essay.  For their programming and discovery processes, the students can again work in the same Jupyter notebook environment as in phase 3.1 (see also glossary, section9.1.4 ). Here there are two different work environments (or subfolders) in which the student groups can find the notebooks for carrying out their data exploration:   1. *Environment NASA data*   If you have not collected your own data or compiled data from various sources, you can use the notebooks in the *Environment NASA Data* subfolder. There, the students can use the Jupyter Notebook *Computational\_Notebook \_NASA-Data.ipynb* to load various weather data for any location and for any time period directly into the Jupyter Notebook and analyze it there. The weather data originates from the POWER project of the NASA Langley Research Center (LaRC), which is funded by the NASA Earth Science/Applied Science Program and is always available almost daily [(](https://www.telekom-stiftung.de/en)<https://power.larc.nasa.gov/data-access-viewer/>). It is usually possible to load data for all locations from 2001 up to three days before the current time directly into the notebook. How this works is explained directly in the Jupyter Notebook. In addition, students can use the Jupyter Notebook *Computational\_Notebook \_NASA-Data\_Example\_visualizations* and how they can be created with the "NASA data". In contrast to your own data - as in b) - the "NASA data" does not have to be prepared additionally, as this has already been done "in the background". In particular, the data types do not have to be determined or the index defined, as shown in the worked examples, for example. The students therefore do not have to pay attention to these sections of the worked examples.   1. *Environment own data*   If the students have collected or compiled their own data and saved them as csv file(s), they can read these csv files into a DataFrame in the Jupyter Notebook *Computational\_Notebook.ipynb* and then explore them. How to read in the data is explained in more detail in the Jupyter Notebook  *Tutorial\_Daten\_einlesen* explained in more detail. Subsequently, data preparation is often necessary, within which the data types must be communicated and the index of the DataFrame must be defined. Students can find out how this works in the Worked Example *ComputationalEssay\_CO2.ipynb*, for example, which is located in the subfolder *Worked Examples -> CO2 measuring station*. The *Worked Examples* folder is located at the top level of the Jupyter Notebook environment.  To carry out their own exploration and programming processes, the students use a worked example as a guide and at the same time as a code base (see also the entry in the glossary under 9.1.2). On the one hand, the students can adopt the external structure of the worked example for their analysis or adapt it to their own needs. On the other hand, they can also adopt specific "code snippets" or entire code cells from the worked example for their own project  For this purpose, the Worked Example "WorkedExampleCO2.ipynb" is located in the Jupyter Notebook environment in the subfolder "Worked Examples/Worked\_Example\_CO2-Messstation". This worked example shows a data exploration process along the PPDAC model in the context of the topic "CO2 indoor air analysis" (see glossary - section9.4.2 ) and deals in particular with the topics "Importing data", "Filtering data", "Grouping data" and the creation of various visualization forms. Alternatively, a shorter worked example on the topic of weather analysis in London and Paderborn is available, which shows (possibly with less explanatory text) how data can be read in, filtered and visualized in a scatter or box plot  One possible approach for the students could be first copy the code snippets for reading in the data from the worked example, insert them into their own Jupyter Notebook document and adapt them to their own data. You could then select an evaluation or visualization from the worked example that you hope will give you an initial overview of your own data set. They can also copy and adapt the code accordingly. Following the creation of an initial overview and for a more in-depth analysis (for example by filtering/grouping the data), they can then consider adjustments/extensions against the background of their own interest in knowledge or their own research question, using the worked example as a "quarry" for relevant program code.  For the analysis, the student groups can now proceed freely and explore the data in "their" Jupyter Notebook by applying the data operations and data transformations "Filter", "Visualize", "Group", "Calculate statistical parameters". How which of these operations and transformations is used always depends heavily on the respective interest in knowledge of the individual projects of the groups. The worked example can provide guidance by helping students to discover suitable operations and transformations that they can use in their own project. The teacher can support the groups with questions during this phase, but should hold back as much as possible in terms of guiding the projects so that the students can proceed independently and gain new insights, which is an essential goal of the entire teaching module in terms of developing epistemic agency (see section ).9.4.3  The time frame for data exploration can be designed very flexibly. At a minimum, the student groups should be given two double lessons/learning units (2\*90 minutes) to complete their exploration.  It can be helpful to make a short interim presentation of the individual projects in plenary at regular intervals, in which the individual student groups briefly report on their current state of knowledge and process, where questions or problems may lie and what they have already solved/achieved. This may provide opportunities for individual groups to exchange ideas or help each other by coming together briefly after these interim presentations and consulting with each other.  Didactic commentary:  In order to support the students in carrying out their data project and developing their computational essay, the following aspects should be given to them as a procedure:  In this phase, it is important to give the student groups time and space for free exploration of the data. It is advisable to start with an overview at the beginning of the exploration - e.g. with the help of a simple scatter or box plot - and then to delve deeper and deeper into the data exploration with regard to your own question, whereby the interest in knowledge can also continue to change/be more specific. In doing so, you can build on previous findings.  In this respect, it could be helpful for the students to record their respective findings directly in the Jupyter Notebook and, building on this, to plan and record their next steps directly there. Documentation of the program code or the individual code cells can also take place at this stage. This is already a helpful step for the creation of a computational essay (see sections ,2.39.1.1 and9.4.4 ), so that the work in phase 3.3 can already be relieved here.  In this phase, the students should be instructed to use the selected worked example as an aid. Not only can they use the external structure as a guide, but they can also copy, adapt and combine individual pieces of code from the worked example. Students should be encouraged to use existing program code in particular.  It is also important to mention that the student groups should make sure that they save their edited Jupyter Notebooks every now and then - especially before they log out of the Jupyter Notebook environment. This can be done with an open Jupyter Notebook by clicking on "File" and then "Save and Checkpoint" in the bar at the top. | * Exploration of the environmental data with regard to your own research question * Performing data operations and data transformations "Filtering", "Visualizing", "Grouping", "Calculating statistical parameters" * Getting to know worked examples and how to work with them for your own programming processes | Jupyter Notebook environment  <https://go.upb.de/ep-umweltanalyse>  In it: Jupyter Notebooks in the subfolder *Umgebung NASA-Daten*  or  *Umgebung eigene Daten*  and Worked Examples in the subfolder *Worked Examples* |
| **3.3** | **Finalizing the computational essay**  **PPDAC**  After the groups have already been able to carry out various data operations and data transformations in the previous phase, they are introduced to the format of a computational essay in this phase (see section9.1.1 ). Based on this, the individual groups finalize their knowledge and programming processes, with the aim of producing a computational essay at the end.  This phase therefore consists of an input on computational essays and a programming phase in which the individual groups finalize their computational essays as Jupyter notebooks.  Input on the topic of computational essays can be provided using the PowerPoint slides with the name "Input\_Computational\_Essays". The file also contains specific notes on the individual slides, which can support the explanation in class. The slides explain the PPDAC cycle (see also section9.4.2 ), which is a "typical" process for data projects and which the groups of students in this teaching module followed - albeit unconsciously. The term "computational essay" is then clarified before explaining how a computational essay can be structured and what it can contain if it is structured according to the PPDAC cycle.  In addition, the worked example for the CO2 data can also be run through again - either together in plenary or individually by the students (in the subfolder "Worked Examples => Worked\_Example\_CO2-Messstation => WorkedExample\_CO2.ipynb"). The Worked Example is also a computational essay, which is structured according to the PPDAC cycle and can therefore provide good orientation for the students with regard to their own Jupyter notebooks.  The table on Checklist 2 (Computational Essay) serves as support for the creation and finalization of the computational essay. This table shows what should be included in a computational essay for each step of the PPDAC cycle. The group can use this table and tick off in the "Done" column which aspects they have already considered for their own computational essay.  Didactic commentary:  This phase or the input for the computational essays can also take place at a different point in the cognitive and programming process and does not necessarily have to take place at the very end as shown here. The advantage of addressing the topic earlier would be that the students could prepare their Jupyter notebooks according to the format of a computational essay right from the start. This would ensure that the Jupyter notebook does not get "mixed up" in the many exploration steps. However, the students would then no longer be able to go through the exploration process as freely as they could if they were initially able to explore the data completely freely, without having to take the external form of the Jupyter notebook into consideration at the same time[[3]](#footnote-4) . This is probably why a procedure in which the students first proceed freely and can then structure the Jupyter notebook according to a computational essay makes sense.  The aim at the end of this phase should be for the student groups to have created a structured computational essay that can then be made accessible to other student groups or people outside the course who are interested in the research questions the students have addressed. In the computational essay, the knowledge and programming process should be presented in a comprehensible and reproducible way so that readers of the computational essay can interact with it or extend/modify it. The PPDAC cycle can provide a good structure for the structure of the computational essay. As shown in the slide set on slide 12, different aspects from the problem description to the conclusion can be highlighted in the computational essay, resulting in a coherent structure for the explanation of the cognition and programming process. Of course, a model other than the PPDAC cycle can also be used. However, in the context of this teaching module, it represents a suitable "common thread" that can be applied directly to the data exploration process of the student groups.  The table in checklist 2 (computational essay) provides the student groups with a guide for creating and finalizing the computational essay. Here, they should first check off in the column highlighted in gray if they have considered an aspect for their computational essay. The aspects listed in the table can of course also be adapted or expanded according to the specific programming project. | * Getting to know computational essays as an interactive way of documenting your own cognitive and programming process * Adaptation and completion of a computational essay as a project report | Powerpoint slides for the computational essay and the PPDAC cycle (Input\_Computational\_Essays)  Jupyter Notebook environment  <https://go.upb.de/ep-umweltanalyse>  In it: Jupyter Notebooks in the subfolder *Umgebung NASA-Daten*  or  *Umgebung eigene Daten*  and Worked Examples in the subfolder *Worked Examples*  Computational Essay Checklist:  Checklist\_Computational\_Essay |
| **Part 4: Presentation and exchange of computational essays** | | | |
| **4.1** | **Exchange of computational essays in different groups**  In this phase, two groups of students exchange their computational essays with each other. How this works is explained in the information document "Exchange of Jupyter Notebooks".  Each group should now first read through the computational essay they have received at their leisure and try to understand and comprehend the knowledge and programming process of the other group. Afterwards, the group discusses among themselves and records on the first page of checklist 2 which insights they were able to gain from the computational essay, what was particularly successful or understandable and what they did not (fully) understand or where information was still missing.  The group then fills in the table on the checklist together (pages 2 and 3) in order to be able to give precise feedback on the individual steps of the PPDAC cycle. The group should agree on a rating of +, o or - for each of the aspects listed (roughly + "well done", o "something could be changed here" and - "something is missing here or something needs to be changed here"). If an aspect does not apply to the computational essay due to the specific project (for example, if the data is already available in prepared form - see aspect 3a)), "Not applicable" should be ticked. At the same time, there is space below to add notes on individual aspects. Here, the group should point out specific aspects that the other group should still work on or that are unclear and make constructive suggestions for improvement. In particular, the aspects that were rated with - should be taken up here with notes. | * Perceiving computational essays from the reader's perspective * Giving feedback on another computational essay | Computational Essay Checklist: Checklist\_Computational\_Essay  Jupyter Notebook environment  <https://go.upb.de/ep-umweltanalyse>  Computational essay from another group  Note document "Exchange of Jupyter notebooks":  Note\_ Exchange\_Jupyter\_notebooks |
| **4.2** | **Incorporating feedback for your own computational essay**  In this phase, the groups revise their own computational essays based on the feedback they have received from the other group using Checklist 2 (Computational Essay). The groups proceed in four steps. First, they identify aspects that they would like to revise and can revise (with reasonable effort). Then they consider what needs to be done or adapted/added in order to revise the relevant aspects. The groups record both steps in the table on page 4 of the checklist. Finally, they then carry out the adjustments/extensions from the table and finally check whether the aspects noted could be improved.  In the first step, the groups review the checklist completed by the partner group (p. 1-3) and discuss which aspects they would like to adapt based on this feedback. First, they go through the general feedback on the first page of the checklist and work out as many specific points as possible, which they record in the first column of the table on page 4. They then go through the table on pages 2 and 3, which is based on the PPDAC cycle, and make a note of the aspects marked as negative and the corresponding notes based on those of the other group on the bottom of page 3. They can first mark the aspects they want to revise in the table on page 2/3 by marking them in the last column "Revision", before recording these marked aspects in the table on page 4, also taking into account the notes of the other group.  In the second step, for each of the aspects noted in the first column of the table on page 4, they consider how they would like to improve this aspect or how they would like to adapt/extend the computational essay. They record these suggestions for adaptation/extension, which should be as concrete as possible, in the second column of the table on page 4.  In the third step, they then revise the computational essay (possibly in a division of labor) by implementing the recorded suggestions.  Finally, or in the fourth step, they then critically examine once again whether the aspects noted could be sufficiently improved with the help of the adaptations/extensions of the computational essay. They can mark which aspects this applies to in the "Done" column. The group can then go through a second improvement loop for the remaining aspects.  Didactic commentary:  In this phase, the groups should work as independently as possible and based on the feedback from their partner groups. Of course, they can exchange information with the other group, especially if there are questions of understanding or ambiguities, and, if necessary, ask again in more detail what was not yet optimal about the computational essay or how they can improve a certain aspect.  It is important to note that the group should of course only tackle those aspects that can be improved (with reasonable effort). In particular, no new research questions or completely new evaluation methods should be tackled. Rather, the aim of this phase should be to present the findings and the process by which they were obtained in a clear and comprehensible manner and to bring a "common thread" to the computational essay. | * Process feedback in a structured manner and develop adaptation/extension proposals based on this * Adapt and expand existing computational essay based on feedback | Computational Essay Checklist: Checklist\_Computational\_Essay  Jupyter Notebook environment  <https://go.upb.de/ep-umweltanalyse>  Own computational essays |
| **4.3** | **Presentation of the computational essays**  In this phase, the student groups are given the opportunity to present their computational essays to the entire learning group (and possibly also to other interested persons) and make them available to the public (e.g. via the school website).  For the presentation of the computational essays, the computational essays are presented in a round. For this purpose, one student from each group forms a new group. In order to make the presentation in this new constellation as active as possible for all students, the phase can be divided into three parts for each presentation: Short presentation, individual exploration, questions and answers  For the short presentation, the groups that have created the computational essays together should get together before this tour and think together about how they want to design the short presentation. To do this, they make notes on the first page of worksheet 4 (presentation) on the above-mentioned aspects, which they can then use later to give their short presentation. For the subsequent phase of individual exploration, the group also compiles their own computational essay, including all the necessary resources (especially data, possibly also images) and makes these files available so that other students can access them. How this works is explained in the information document "Exchange of Jupyter Notebooks".  *NOTE: In the future, it should be possible for this step of providing the Jupyter notebooks and resources to be carried out by the teacher. As soon as this is possible, it will be explained here.*  In the first phase, the group leader presents the computational essay in a short presentation (max. 3 minutes) by describing what the group has investigated, how they proceeded, what evaluations and visualizations were created and what insights were gained.  This short presentation by the respective group is then followed by an individual exploration phase (approx. 10 min) in the tour. All students go through the computational essay of the group currently presenting and try to understand the cognitive and programming process. In doing so, they take notes on the AB 4, recording what the group has investigated, what insights they have gained and how they proceeded. The student from the group that is currently presenting can provide support with relevant questions at short notice.  There is then an opportunity to ask questions to the respective group leader. Based on this, the students can add to their notes again. In addition, they should each define a title for the project they are currently looking at on the free line behind "Project presentation". They then continue with the short presentation of the next group.  This three-step phase is run through here for all groups, so that at the end the pupils have made notes on each group on their sheet 4 and have also added notes on their own group on the first page.  *Optional, if publication of the computational essays is desired:*  Following this presentation, the computational essays can be exported as an HTML file for presentation to the public. How this works can be found in section9.1.6 . Using the HTML file, the computational essay can then be published as a page on the school website, for example, or in another suitable place. It is important to note that the program code in the HTML file or after it has been published on the website cannot be changed. Nevertheless, the evaluations and visualizations in this file can be viewed interactively, i.e. in particular you can "zoom in" on the visualizations as usual or display additional information by "hovering" the mouse over them. In this way, the computational notebooks and the underlying cognitive processes can be made accessible to the general public.  Didactic commentary:  Other interested people from the school or the public can also take part in the tour. In particular, if issues of importance to the local area have been addressed, it may be interesting to invite relevant members of the public (e.g. local politicians, the press, etc.) to present the results to suitable decision-makers, who can then take appropriate action if necessary.  The tour with the individual exploration phases is particularly suitable in that it allows all pupils to form their own impression of the other projects and actively engage with their cognitive and programming processes. They also present their own cognitive processes and thus reflect on their own process once again.  Alternatively, a presentation can be given in plenary instead of the tour, for example to save time. During a presentation, the students in each group can go through the computational essay step by step and explain the individual steps in more detail.  Pages 2 and 3 of worksheet 4, which contain the work instructions including placeholders for the notes for the other groups, can be copied by the teacher as often as necessary depending on the number of groups.  By taking notes on worksheet 4, the students record their findings and the processes behind them for all groups. Thus, in addition to the computational essays themselves, these notes serve to secure the knowledge and programming processes of their own and the other groups.  Publication on the school website is not mandatory and can also be omitted or shifted to another website or medium if this is more appropriate in the respective setting. | * Compiling and presenting own group results * Actively run through the computational essays of other groups and understand the respective cognitive and programming processes * Exchange on the knowledge gained and the programming processes in the context of which this knowledge was gained * Publication of the group's own computational essays | WS4\_Praesentation  Jupyter Notebook environment  <https://go.upb.de/ep-umweltanalyse>  Computational essays of all groups that are provided  Note "Exchange of Jupyter notebooks":  Note \_Exchange\_Jupyter\_Notebooks |
| **Part 5: Reflection** | | | |
| **5.1** | **Reflection on the cognitive and programming process**  In this phase, the students become aware of their individual perception of the programming process in this teaching module and of the role of programming. To do this, they reflect on their own cognitive and programming process and perspectives on the role of programming using worksheet 5 - Reflection on the programming process. This is divided into two phases. In the first phase, the students work individually on tasks 1-5 of the worksheet, which are divided into two parts (own project or cognitive and programming process and perspectives on the role of programming). In the first 3 reflection tasks, the focus is initially on examining their own group's cognitive and programming process once again and recording how it went and what role the computer or programming played. In the subsequent tasks 4 and 5, the students then take a meta-perspective and evaluate the perception of programming as a means of gaining knowledge on a personal or social level and, from there, develop further ideas for programming projects that are also about gaining knowledge about personally or socially relevant topics. It is important that it is made clear to the students that there is no "right" or "wrong" in these questions, but that it is a matter of the students individually becoming aware of their perception of programming as a means of gaining knowledge or its role in knowledge processes, in order to be able to use it later as such a tool in accordance with this perception.  After the students have completed these first 5 tasks, there is a final exchange and discussion phase in which the students can share their perceptions of programming and, if necessary, take on board further perspectives/ideas. The discussion phase can either follow the 5 tasks or alternatively be more open. A possible starting point could be the statement from task 4, which the students can discuss. They could then also contribute experiences from their own cognitive and programming processes that they have previously recorded in the tasks. They could also exchange further ideas from task 5 afterwards.  As a backup, the students then work on task 6 on the worksheet, in which they record individually relevant characteristics of the programming concept discussed for themselves, so that these can serve as a mental anchor in later (own) knowledge-centered programming projects, so that the students can immerse themselves in a similar process again.  Didactic commentary:  In this phase, the students should become aware of their perception of the knowledge-centered perspective on programming taught in this teaching module. This has 2 main goals: Firstly, the students should become aware that programming is not "only" suitable for the development of professional software, but can be a helpful tool for exploring personally or socially relevant issues or facts. On the other hand, the students should consider for themselves where they can use this type of programming - whether in everyday life or in other contexts. During the discussion phase, they should also be made aware (if necessary by the teacher's explanation) that there are freely accessible libraries and documentation for a large number of project ideas similar to the context of environmental data analysis discussed here, as well as ready-made and commented partial solutions (similar or exactly like worked examples).  In this phase, the students should be encouraged to adopt programming - figuratively speaking - as a tool for their "toolbox" in order to pursue their own questions or interests - possibly also data-based. As part of this project, they have been given an initial "basic ability" to do this based on worked examples and with the help of ready-made libraries.  *A small request:*  In our research as part of the ProDaBi team, we are working on the development of epistemic programming as a knowledge-centered programming approach. An essential part of this relates to collecting and evaluating the students' perspectives on programming after the implementation of this teaching module. At this point, there is therefore the opportunity for the students to become part of this research by collecting and evaluating their work on worksheet 5. If you and your students are interested in supporting us in our research, we would be very pleased if you would send an e-mail to Sven Hüsing ([sven.huesing@upb.de](mailto:sven.huesing@upb.de) ), who is responsible for this module. You will then receive a consent form that the students and their legal guardians must fill out in order to participate in the study as part of the research described. We would be very grateful if you and your students would support us in this regard. | * Awareness of the individually perceived role of programming * Reflection on your own programming processes and the roles of humans and computers in them * Working out typical characteristics of knowledge-centered programming processes | WS5\_Reflection |

# Descriptions of the materials and further background information

In this section, individual materials - and in particular the Jupyter notebook environment in which the students work during the teaching module - will be explained in more detail.

## Working with Jupyter notebooks and Jupyter notebook environment

In this teaching module, students evaluate previously collected or gathered data in Jupyter Notebooks. The Jupyter Notebook environment is a web-based, interactive platform where users can create programming-based narratives that can include live code, explanatory text, interactive elements and other media (see Perez & Granger, 2015, p. 2 ). The documents that users can create, known as Jupyter Notebooks, consist of code and Markdown cells arranged one below the other. Users can write code in the code cells, the output of which can be displayed directly below the code cell when the code is executed. By using several code cells, the entire program code can be split up and intermediate results can be displayed. You can also add comments in the Markdown cells, for example to document the program code.

An example of a Jupyter notebook can be found here: <https://jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb>(accessed on 20.11.2024; Note: After accessing the link, you must first click in the empty cell at the top of the page to activate the notebook).

One advantage of using Jupyter Notebooks for work in schools is that teachers can provide ready-made Jupyter Notebooks that already contain program code (from various libraries), for example. Students can then build on this program code and expand it to suit their own interests. A Jupyter Notebook can also be used as a digital worksheet by teachers writing tasks in the Jupyter Notebook (as Markdown cells). The students can then work on the tasks in (possibly prepared or already partially completed) code cells.

Further information about Jupyter notebooks can be found in the documentation at <https://docs.jupyter.org/en/latest/>(accessed on 20.11.2024)

### Computational essays as programming products

A computational essay is an interactive document that contains explanatory text, small programs or program snippets and outputs to present the results of an investigation or express one's own ideas (see Atkinson et al., 2000; Odden & Malthe-Sørenssen, 2021; Wolfram, 2017 ). A computational essay can therefore be used to present the programming process in an understandable way so that readers can understand how the results and conclusions were arrived at (see also Hüsing & Podworny, 2022 ). At the same time, readers of a computational essay can interact with the content or the code and in this way explore the programming process further/deeper on their own.

In the context of data exploration, computational essays are a suitable means of recording and documenting the interwoven data exploration and programming process. In a computational essay, the evaluation can be explained step by step and at the same time the programming results such as visualizations or the calculated characteristic values can be described and interpreted. The use of Jupyter Notebooks and the division into many programming and markdown cells is very well suited for this purpose.

Below you can see an excerpt from a sample computational essay in which a comparison of the temperature in London and Paderborn is shown. The green cells contain an explanation of the program code used and a (brief) interpretation of the two graphs shown.

Ein Bild, das Text, Screenshot, Webseite, Software enthält.

Automatisch generierte Beschreibung

### Use worked examples as support for programming

A worked example is a fully developed and documented example of a project similar to the one being considered and tackled (see Atkinson et al., 2000; Muldner et al., 2023 ). It can be used as a source of inspiration and orientation - especially if you have little experience in the context of the project. In the context of programming, worked examples can also serve as a "quarry" for program code, in that learners first copy the program code from a worked example, then adapt it to their own needs or their own project and then extend it (see also "Use-Modify-Make" in Lee et al., 2011 and Hüsing et al., 2024 ). Computational essays are particularly suitable here as a format for a worked example in order to make the programming process transparent for learners and to show how the programming snippets can be used.

There are two worked examples in this teaching module that can help students to carry out their own explorations of environmental data. The worked example on CO2 analysis shows how CO2 data can be analyzed in connection with other variables. In addition, the worked example on weather analysis shows how weather data from different locations can be explored. Learners can use these worked examples as a guide or "code quarry" for their own environmental data projects.

### Access to the Jupyter Notebook environment for this series of lessons (creating an account and logging in)

A prepared Jupyter Notebook environment is available for the data analysis phase of this teaching module. This can be accessed via the following link: <https://go.upb.de/ep-umweltexploration>

Ein Bild, das Text, Schrift, Screenshot, Design enthält.

Automatisch generierte BeschreibungWhen opening this link for the first time, students must create their own account. To do this, they enter a user name and password in the login screen. They can choose either of these freely. The only important thing is that they remember both so that they can always access their work progress in their own account. To do this, you can log in again later using the same link with the data you entered when you first logged in.

### Structure and contents of the Jupyter Notebook environment

The Jupyter Notebook environment contains various Jupyter notebooks that can be used by students in the teaching module. After logging into the environment, a folder overview is displayed. Each folder (except for the res folder) represents a separate environment, which is briefly described below.

* Cheatsheets
  + Cheatsheet data exploration in Jupyter Notebooks with Python.ipynb
    - *Cheatsheet, which shows and briefly explains essential Python commands and code snippets for data exploration in Jupyter Notebooks; students can use this cheatsheet as an aid during their own data explorations*
* Tutorial\_JupyterNotebooks\_Python

*This folder contains several tutorial Jupyter notebooks that introduce the use of Jupyter notebooks with the Python programming language. The Jupyter notebooks can be run through and edited individually.*

* + 0\_Start.ipynb
  + 1\_First\_steps.ipynb
  + 2\_Python\_Basics. ipynb
  + 3\_Case distinctions.ipynb
  + 4\_Repeats.ipynb
  + 5\_Functions.ipynb
  + 6\_DataFrames.ipynb
* Environment own data

*This folder contains the environment for exploring your own data, which you have either collected yourself (see phase 2a of the lesson series) or gathered and compiled from various sources (see phase 2b of the lesson series).*

* + Computational\_Notebook.ipynb
    - *Jupyter Notebook, in which the students can carry out their own data analysis; already contains the import of the essential libraries as well as the task/introduction*
  + Own\_basic\_information.ipynb
    - *Jupyter Notebook, in which basic information about the data (e.g. measurement methods/origin etc.) can be recorded.*
  + Measurement protocol.ipynb
    - *Jupyter notebook, in which information about the measurement and the measurement protocol can be recorded if the students have collected their own data*
  + Tutorial\_Data\_import.ipynb
    - *Short tutorial explaining how to load your own data into the Jupyter Notebook environment*
* Environment NASA data

*This folder contains the environment for the exploration of data from the NASA Langley Research Center (LaRC) POWER project, which is funded by the NASA Earth Science/Applied Science Program*

* + Computational\_Notebook\_NASA-Data.ipynb
    - *Notebook, in which students can load various weather data for any location and for any period of time directly into the Jupyter Notebook and analyze it there; The following weather data can be loaded into the Jupyter Notebook:* 
      * *Humidity in %*
      * *Precipitation in mm/hour*
      * *Temperature in °C*
      * *Wind speed in m/s*
  + Computational\_Notebook\_NASA-Data\_Example\_Visualizations.ipynb
    - *Jupyter Notebook, which uses examples to show how certain visualizations can be created with the "NASA data"*
* Worked Examples

*This folder contains 2 different worked example projects for exploring environmental data. The project is presented as a worked example, which learners can use to orient themselves or to use/adapt/extend individual code snippets.*

* + Worked\_Example\_CO2 measuring station
    - Basic information.ipynb
      * *Jupyter Notebook, which contains important information on data measurement*
    - WorkedExample\_CO2.ipynb
      * *Jupyter Notebook, in which an exploration of CO2 data in the classroom is shown as an example*
  + Worked\_Example\_Weather\_Data
    - WorkedExample\_Weather\_data.
      * *Jupyter Notebook, in which an exploration of weather data (from London and Paderborn) is shown as an example*

*Note: The folders with the names "res" or "data" contain files (RESsources) such as images and data that are used in the Jupyter notebooks, but which do not need to be accessed directly via the folder structure.*

### Downloading your own Jupyter notebooks and uploading other Jupyter notebooks (e.g. from other groups)

Here in the teaching module, the groups should exchange the Jupyter notebooks they have created in phase 4 and give each other feedback on them. It can also be useful in other situations to exchange the Jupyter Notebooks with each other, for example if the Jupyter Notebooks are to be presented and interested parties are looking at the Jupyter Notebooks that have been created. In these cases, the Jupyter Notebooks must be downloaded from the Jupyter Notebook environment. They can then be uploaded again later in another person's account.

Download a Jupyter notebook

There are several ways to download a Jupyter Notebook from the Jupyter Notebook environment. Either you download the Jupyter Notebook when it is open or you download it from the folder view.

If you want to download the currently opened Jupyter Notebook, click on "File" at the top of the list and then on "Download". The Jupyter Notebook will then be downloaded as an ipynb file.

Ein Bild, das Text, Screenshot, Software, Webseite enthält.

Automatisch generierte BeschreibungAlternatively, you can also download the Jupyter Notebook from the folder view. To do this, click on the tick next to the Jupyter Notebook you want to download and then click on "Download" at the top of the bar. You can also download several files at once by activating the checkmark next to several files and then clicking on "Download".

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

***Important note:***

*To ensure that the Jupyter Notebook can also be opened in another account, it is important to download all data and resources, such as images, and upload them again later in the same folder structure. For example, if a "data.csv" file is located in a "Data" subfolder, such a subfolder must also be created in the new account, into which the "data.csv" file is then uploaded.*

Uploading a Jupyter notebook

*Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung*To upload a downloaded Jupyter Notebook to another account, click on the "Upload" button in the folder view. Then select all the files you want to upload. The files should then appear directly in the folder. You can also copy or move them to other folders using drag & drop or by right-clicking on one or more selected files.

***Important note:***

*In order for the uploaded Jupyter Notebook to function correctly, all resources to which the Jupyter Notebook must have access (such as data or images) must also be uploaded in the same folder structure as before. For example, if a file "data.csv" is located in a subfolder "data", such a subfolder must also be created in the new account, into which the file "data.csv" is then uploaded.*

### Export of a Jupyter notebook as HTML file

To make a Jupyter notebook accessible to a wider public, it can be exported as an HTML file. This offers the advantage that although the program code can no longer be changed, the programming results are still interactive. In particular, it is possible to zoom in and out of developed visualizations. The HTML file can be used, for example, to publish the learners' computational essay on your own homepage and thus provide comprehensible and transparent information about the learners' findings, which may also be relevant to local society.

To export a Jupyter Notebook as an HTML file, open the Jupyter Notebook and then click on "File => Save and Export Notebook As => HTML". The downloaded file can then be opened in any browser. The Jupyter Notebook will then be displayed accordingly and you can interact with the programming results. At the same time, you can use the code from the HTML file to display the computational essay on the school homepage, for example.

Ein Bild, das Text, Screenshot, Reihe, Diagramm enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Working with the Sensebox and the OpenSenseMap

### The Sensebox

The Sensebox is a programmable measuring instrument that can be used to collect and store your own data. It consists of a microcomputer to which various measurement sensors can be connected to collect data. The data can be transmitted via an SD card or via the OpenSenseMap (using a WiFi or LoRaWan module).

The Sensebox can be programmed either using the block-based language   
or with the Arduino IDE (text-based programming).

* Information about Blockly: <https://docs.sensebox.de/en/docs/boards/mcu/mcu-erster-sketch/>
* Information about the Arduino IDE: <https://docs.sensebox.de/en/docs/category/arduino/>

You can find more information about the Sensebox here: <https://docs.sensebox.de/en/>

### The OpenSenseMap

The OpenSenseMap is a platform on which you can register your own Sensebox in order to send data to the platform (via WiFi or LoRaWan module on the Sensbox), collect it and publish it. At the same time, you have access to the data of all registered Senseboxes worldwide via the platform. In this way, data records can be downloaded as csv files from different locations around the world and used for your own analysis.

Further information on the OpenSenseMap: <https://sensebox.de/en/opensensemap>

## Working with data from the POWER Project of the NASA Langley Research Center (LaRC)

In the "NASA Data Environment", students can load weather data from all over the world directly into their Jupyter notebooks. The data comes from the NASA Langley Research Center (LaRC) POWER project, which is funded by the NASA Earth Science/Applied Science Program [(](https://power.larc.nasa.gov/data-access-viewer/)<https://power.larc.nasa.gov/data-access-viewer/>). In addition to the data on temperature, humidity, precipitation and wind speed, many other data can be downloaded from the website in various file formats. These can also be uploaded to the "Environment own data" section, for example.

Here you can find more information about the POWER project: <https://power.larc.nasa.gov/docs/>

Note: The POWER project kindly requests a reference, web link and/or reprint of published papers or reports or a brief description of other uses (e.g. posters, oral presentations, etc.) of data products they have provided/facilitated. This helps them determine the use of the data, which is helpful for optimizing product development. It also helps them to assess the value to the community. For more information on submitting reference material, please contact the POWER Project Team: [larc-power-project@mail.nasa.gov](mailto:larc-power-project@mail.nasa.gov)

## Glossary

### Epistemic programming

Epistemic programming (Hüsing et al., 2023) refers to an epistemic programming practice in which programming is used as a means of gaining knowledge about personally or socially relevant phenomena or issues. A typical example of an epistemic programming project is the execution of a data exploration to investigate a context that can be represented by data. In order to learn epistemic programming as a practice, it is advisable to carry out project-based lessons in which learners use programming in a prepared environment to approach their own questions in a tinkering-like process. Computational essays (see also sections ,2.39.1.1 and9.4.4 ) are suitable here both as intended products and as so-called worked examples (see also sections ,2.49.1.2 and9.4.5 ) in order to support the cognitive and programming process, especially for programming innovators.

### PPDAC cycle

The PPDAC cycle (Wild & Pfannkuch, 1999) is a model for the implementation of data projects and is divided into the phases "Problem", "Plan", "Data", "Analysis" and "Conclusions". These phases can be run through and repeated one after the other in cycles.

* In the *problem phase,* the aim is to narrow down and define the problem more precisely.
* In the *plan phase*, the data collection and evaluation is planned.
* *The data phase* involves data collection, data management and data cleansing.
* In the *analysis phase*, the data is explored and results and hypotheses are generated on this basis.
* In the *Conclusions phase*, the results are interpreted and conclusions and new ideas are generated.

### Epistemic Agency

The construct of epistemic agency refers to the ability to co-determine knowledge and knowledge practices in a community (Miller et al., 2018; Stroupe, 2014) or to take cognitive control and responsibility for one's own learning (Odden et al., 2023) . Consequently, learners can achieve epistemic agency by being actively involved in knowledge and cognitive processes and by being given the freedom to independently explore personally or socially relevant interests. Miller et al. (2018) provide four ways to promote epistemic agency in learners:

* Using students' knowledge as a resource for learning and on it
* Allow learners to actively develop knowledge
* Build knowledge that is useful for the students (e.g. through authentic questions)
* Using learning and knowledge to initiate changes in structures or actions

### Computational Essay

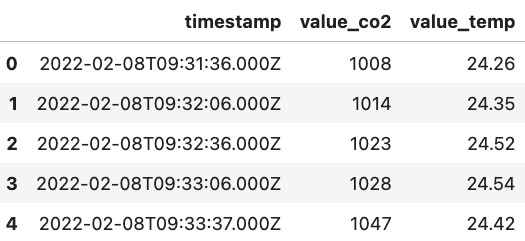
A computational essay can be described as a formulated essay which, in addition to textual descriptions, interpretations and explanations, also contains manipulable program code that can be executed directly in the medium (DiSessa, 2000; Odden et al., 2023; Wolfram, 2017) .

### Worked Examples

In the context of epistemic programming, a worked example refers to an elaborated and documented programming example in which processes and methods are applied that could be useful for the students' programming project (Atkinson et al., 2000; Hüsing et al., 2024b; Muldner et al., 2023) . Learners can use a worked example both as an orientation for their own programming process and as a "quarry" from which they can use, adapt and extend code (see also Lee et al. (2011) and Sentance et al. (2019) ).

### Time series data

Time series data refers to data that has been collected repeatedly over a certain period of time. The time of collection is also recorded for each value collected. Time series data thus makes it possible to analyze the course of the collected values over time. For example, the data can be displayed in a scatter plot:



# 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. https://doi.org/10/csm67w

DiSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. MIT Press.

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. https://doi.org/10/gqv58h

Hüsing, S., Schulte, C., Sparmann, S., & Bolte, M. (2024a). Using Worked Examples for Engaging in Epistemic Programming Projects. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. Technical Symposium on Computer Science Education (SIGCSE TS), New York, NY, USA. https://doi.org/10.1145/3626252.3630961

Hüsing, S., Schulte, C., Sparmann, S., & Bolte, M. (2024b). Using Worked Examples for Engaging in Epistemic Programming Projects. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, 443-449. https://doi.org/10.1145/3626252.3630961

Hüsing, S., Schulte, C., & Winkelnkemper, F. (2023). Epistemic programming. In S. Sentance, E. Barendsen, N. R. Howard, & C. Schulte (Eds.), *Computer Science Education: Perspectives on Teaching and Learning in School* (2nd ed., pp. 291-304). Bloomsbury Academic; Bloomsbury Collections. http://dx.doi.org/10.5040/9781350296947.ch-022

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*,*2* (1), 32-37. https://doi.org/10/ggdnzr

Miller, E., Manz, E., Russ, R., Stroupe, D., & Berland, L. (2018). Addressing the epistemic elephant in the room: Epistemic agency and the next generation science standards. *Journal of Research in Science Teaching*,*55* (7), 1053-1075. https://doi.org/10.1002/tea.21459

Muldner, K., Jennings, J., & Chiarelli, V. (2023). A Review of Worked Examples in Programming Activities. *ACM Transactions on Computing Education*,*23* (1), 1-35. https://doi.org/10.1145/3560266

Odden, T. O. B., & Malthe-Sørenssen, A. (2021). Using computational essays to scaffold professional physics practice. *European Journal of Physics*,*42* (1). https://doi.org/10/gjf7nq

Odden, T. O. B., Silvia, D. W., & Malthe-Sørenssen, A. (2023). Using computational essays to foster disciplinary epistemic agency in undergraduate science. *Journal of Research in Science Teaching*,*60* (5), 937-977. https://doi.org/10.1002/tea.21821

Perez, F., & Granger, B. E. (2015). Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science. *Jupyter Blog.* https://blog.jupyter.org/project-jupyter-computational-narratives-as-the-engine-of-collaborative-data-science-2b5fb94c3c58

Rule, A., Tabard, A., & Hollan, J. D. (2018). Exploration and Explanation in Computational Notebooks. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1-12. https://doi.org/10/gfw4vk

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: A sociocultural perspective. *Computer Science Education*,*29* (2-3), Article 2-3. https://doi.org/10/ggdpcg

Stroupe, D. (2014). Examining Classroom Science Practice Communities: How Teachers and Students Negotiate Epistemic Agency and Learn Science-as-Practice. *Science Education*,*98* (3), 487-516. https://doi.org/10.1002/sce.21112

Wild, C. J., & Pfannkuch, M. (1999). Statistical Thinking in Empirical Enquiry. *International Statistical Review*,*67* (3), 223-248. https://doi.org/10/fdhtmj

Wolfram, S. (2017, November 14). What Is a Computational Essay? *Stephan Wolfram Writings.* https://writings.stephenwolfram.com/2017/11/what-is-a-computational-essay/

1. Time series data = data that is collected repeatedly in a specific measurement interval and is provided with a time, see also section 9.4.6 [↑](#footnote-ref-2)
2. Developing epistemic agency means influencing knowledge and knowledge processes independently and taking cognitive control and responsibility for one's own learning (Miller et al., 2018; Odden et al., 2023) ; see also the glossary in the section 9.4.3 [↑](#footnote-ref-3)
3. There is an interesting article on this debate by Rule et al. (2018) , which deals with different perspectives on the creation of Jupyter Notebooks with regard to the basic ideas of "exploring" and "explaining" in Jupyter Notebooks. [↑](#footnote-ref-4)