**Exploration unserer Umwelt durch Epistemisches Programmieren**

**Epistemisches Programmieren als Mittel zur Datenanalyse und Exploration persönlich bedeutsamer Kontexte am Beispiel von Umweltdaten**

**Steckbriefinformationen des Unterrichtsmoduls:**

|  |  |
| --- | --- |
| Titel: | **Exploration unserer Umwelt durch Epistemisches Programmieren - Epistemisches Programmieren als Mittel zur Datenanalyse und Exploration persönlich bedeutsamer Kontexte am Beispiel von Umweltdaten** |
| Zielgruppe: | Informatikkurse sowie fächerübergreifende (GeoMINT) Kurse in Klasse 8 bis 10 (Gymnasium, Gesamtschule) |
| Inhaltsfelder nach dem Kernlehrplan WP Informatik NRW (24.05.23) | „Information und Daten“, „Algorithmen“, „Informatik, Mensch und Gesellschaft“, „Informatiksysteme“ |
| Zeitlicher Umfang: | ca. 16 Unterrichtsstunden je 45 Minuten (ohne eigene Datenerhebung)  ca. 20 Unterrichtsstunden je 45 Minuten (mit eigener Datenerhebung) |
| Tools: | Datenerhebung: Arduino bzw. Sensebox oder andere Messmöglichkeiten (optional) Datenauswertung: JupyterNotebooks und Python |

# Kurzüberblick

Programmieren ist in der heutigen Zeit in vielen verschiedenen Disziplinen allgegenwärtig und wird oft sogar als vierte „Grundfähigkeit“ neben „Lesen“, „Schreiben“ und „Rechnen“ bezeichnet. Während man beim Programmieren schnell an Softwareentwicklungsprozesse denkt, kann Programmieren auch ein Mittel zur Exploration persönlich relevanter Themen oder Fragestellungen sein, welche sich insbesondere auch auf Aspekte außerhalb der Informatik beziehen.

In diesem Unterrichtsmodul tauchen die Schüler\*innen anhand des Kontexts von Umweltdaten und Umweltanalysen in diese erkenntniszentrierte Programmierpraktik (auch Epistemisches Programmieren genannt) ein, um Programmieren als ein Mittel zur Exploration persönlicher oder gesellschaftlicher Interessen kennenzulernen, um dazu befähigt zu werden, Programmieren als solches nutzen zu können. Das Unterrichtsmodul fokussiert dabei beispielhaft datengetriebene Umweltanalysen. Schüler\*innen können in diesem Kontext eine eigene initiale (Forschungs-)Frage formulieren, die sie dann in einem heuristischen Erkenntnis- und Programmierprozess immer weiter ausschärfen bzw. zu einem konkreten Forschungsinteresse zusammenfassen, dem sie dabei durch das Sammeln (eigener) Umweltdaten sowie deren Analyse nachgehen. Ihre Erkenntnisse sowie ihren Programmierprozess halten die Schüler\*innen während der Durchführung ihres Projektes in einer Art „Forschungstagebuch“ – einem sogenannten Computational Essay – fest, welches am Ende des Projektes mit anderen Schüler\*innen oder auch weiteren interessierten Personen geteilt werden kann.

Das Unterrichtsmodul in sich ist insofern modular, als dass der Teil zur Datenerhebung variabel gestaltet werden kann. Entweder können die Schüler\*innen eigenständig Umweltdaten mithilfe eines selbst entwickelten Messinstruments erheben oder die Daten aus verschiedenen Datenquellen zusammensuchen. Dabei können auch zusätzliche Daten oder Datenquellen (wie beispielsweise Wetterdaten aus dem POWER-Projekt des NASA Langley Research Center (LaRC), das durch das NASA Earth Science/Applied Science Program finanziert wird ([https://power.larc.nasa.gov/data-access-viewer/](https://www.telekom-stiftung.de/en))) verwendet werden, die hier u.U. nicht aufgeführt werden.

Das Unterrichtsmodul gliedert sich in insgesamt 5 Teile, welche im Folgenden kurz beschrieben werden. Das Unterrichtsmodul sollte dabei von den Schüler\*innen in Kleingruppen (optimalerweise 3-5 Schüler\*innen pro Gruppe) durchlaufen werden.

1. **Erarbeitung initialer Fragestellungen/Forschungsinteressen**

Zu Beginn des Unterrichtsmoduls erarbeiten die Schüler\*innen eine eigene Forschungsfrage im Kontext des übergeordneten Themas der Umweltanalyse, indem sie zunächst Kriterien für gute (statistische) Forschungsfragen kennenlernen und anschließend in Kleingruppen mehrere Fragen erarbeiten und dabei von anderen Gruppen Feedback und Überarbeitungsvorschläge erhalten. Um Ideen hinsichtlich passender Forschungsfragen zu erlangen, untersuchen die Schüler\*innen zunächst existierende Studien und entwickeln aufbauend darauf eigene/verwandte Fragestellungen. Zusätzlich kann die Lehrkraft auf mögliche Forschungsfragen hinweisen, die ggf. in früheren Unterrichtsdurchführungen untersucht wurden. Beispiele könnten sein: „Wie gut schützen unterschiedliche Textilien (Baumwolle/Polyester) vor UV-Strahlung?“ oder „Wie verhält sich die Feinstaubbelastung an unterschiedlichen Orten (z.B. an einer Hauptstraße im Vergleich zu einer Grünfläche)?“

Am Ende dieser Phase entscheidet sich jede Gruppe für eine eigene initiale Forschungsfrage, der sie im Rahmen ihres epistemischen Programmierprojektes nachgehen möchte.

1. **Daten sammeln**

In einer zweiten Phase sammeln die Schüler\*innen Zeitreihendaten[[1]](#footnote-2) für Ihr eigenes Projekt. Dabei können sie entweder

1. **Daten mithilfe eines Messinstrumentes selbst erheben**

*Hinweis: In diesem Unterrichtsmodul zeigen wir die Datenerhebung exemplarisch mit der Sensebox. Es gibt aber auch noch viele weitere Möglichkeiten an Messwerkzeugen (z.B. auch Arduinos oder „fertige“ Messsensoren).*

1. **Daten aus verschiedenen Quellen sammeln und zusammentragen**

Wichtig ist in dieser Phase, dass die Schüler\*innen Daten zusammenstellen, die für das eigene Vorhaben passend sind. Hierzu sollten die Schüler\*innen während dieser Phase stets ihre initiale Forschungsfrage im Blick haben und kritisch hinterfragen, wie die Daten bei der Beantwortung hilfreich sein können. Natürlich können zusätzlich auch weitere Daten in einer zweiten Erhebung gesammelt werden.

1. **Datenexploration und verwobener Erkenntnis- und Programmierprozess**

Die erhobenen bzw. gesammelten Daten werden in dieser Phase hinsichtlich des Erkenntnisinteresses bzw. der Forschungsfrage der jeweiligen Schüler\*innengruppe ausgewertet. Dabei entwickeln die Schüler\*innen ein Computational Essay, um die Datenauswertung mithilfe der Programmiersprache Python in einem Jupyter Notebook durchzuführen und gleichzeitig Ihren Programmier- und Erkenntnisprozess zu dokumentieren und insbesondere ihre Erkenntnisse festzuhalten (genauere Informationen zu Computational Essays sind in Abschnitt 2.3, 9.1.1 und 9.4.4 zu finden). Unterstützung erhalten sie einerseits durch bereits vorbereitete Python-Bibliotheken, die es ihnen ermöglichen, bereits vorhandene Methoden zur Datenauswertung direkt aufzurufen sowie durch ein sogenanntes Worked Example (ein vorbereitetes und dokumentiertes Beispiel einer Umweltdatenanalyse, siehe auch Abschnitte 2.4, 9.1.2 und 9.4.5), an dem sich die Gruppen sowohl hinsichtlich des Prozesses als auch in Bezug auf konkreten Programmcode orientieren können.

1. **Vorstellung und Austausch der Computational Essays**

In dieser Phase teilen die Gruppen ihre fertigen Computational Essays miteinander, um einerseits die Programmier- und Erkenntnisprozesse der anderen Schüler\*innengruppen nachvollziehen zu können und andererseits Feedback und weiterführende Ideen auszutauschen.

1. **Reflexion**

Zum Abschluss reflektieren die Schüler\*innen den in diesem Unterrichtsmodul genutzten Programmieransatz des (datengetriebenen) Epistemischen Programmierens. Dabei entwickeln sie insbesondere Ideen und Perspektiven für eigene Projekte, in denen sie Programmieren als Mittel zur Exploration persönlich oder gesellschaftlich relevanter Sachverhalte nutzen können.

**Inhalt des Dokuments:**

[1 Kurzüberblick 1](#_Toc191126986)

[2 Epistemisches Programmieren 3](#_Toc191126987)

[3 Didaktische Kernidee: Vermittlung von Epistemischem Programmieren als Mittel zum Erkenntnisgewinn im Kontext von Datenexplorationen 5](#_Toc191126988)

[4 Lernvoraussetzungen 6](#_Toc191126989)

[5 Ziele 6](#_Toc191126990)

[6 Zentrale Leitfragen 8](#_Toc191126991)

[7 Zusammenfassender Überblick über das Unterrichtsmodul 9](#_Toc191126992)

[8 Überblick über den Unterrichtsverlauf 11](#_Toc191126993)

[9 Beschreibungen der Materialien und weiterführende Hintergrundinformationen 34](#_Toc191126994)

[10 Literatur 42](#_Toc191126995)

# Epistemisches Programmieren

## Was bedeutet Epistemisches Programmieren?

Programmieren spielt heutzutage nicht nur im Sinne der Softwareentwicklung eine wichtige Rolle. Auch in vielen Bereichen außerhalb der Informatik kann Programmieren genutzt werden, um Erkenntnisse über domänenspezifische Sachverhalte zu gewinnen. Auf diese Weise können gesellschaftlich oder auch persönlich relevante Sachverhalte erkundet und Antworten zu bestimmten (Forschungs-)Fragen gefunden werden. Ein wesentliches Anwendungsgebiet für epistemische Programmierprojekte stellt die Exploration oder Auswertung von Daten dar, um aus diesen Daten Erkenntnisse über persönlich relevante Sachverhalte aus verschiedenen Domänen, aus denen die Daten stammen können, zu gewinnen. Programmieren lässt sich in diesem Sinne als Werkzeug einsetzen, um die eigene Umwelt anhand von Daten zu verstehen bzw. eigenen Interessen nachgehen zu können – mündig zu sein. Unter dieser Perspektive wird dieser Programmierpraktik insbesondere auch eine allgemeinbildende Rolle hinsichtlich unserer digitalisierten und datafizierten Welt zuteil. Mithilfe des epistemischen Programmierens sollen Lernende Epistemic Agency übernehmen können, d.h. Wissen und Wissensprozesse eigenständig beeinflussen und kognitive Kontrolle und Verantwortung für das eigene Lernen übernehmen (Miller et al., 2018; Odden et al., 2023, siehe auch den Glossar im Abschnitt 9.4.3).

## Epistemischer Programmierprozess

Der Programmierprozess im Rahmen eines epistemischen Programmierprojektes besteht aus einer Interaktion zwischen der programmierenden Person und dem Computer bzw. dem programmierten digitalen Artefakt. Der Ablauf der Interaktion verläuft zyklisch, wobei in jedem Durchlauf neue Erkenntnisse gewonnen werden können, die wiederum das Erkenntnisinteresse beeinflussen und ggf. auch verändern können. Aus dem adaptierten Erkenntnisinteresse entstehen dann Ideen für Handlungsimpulse, worauf aufbauend dann das digitale Artefakt in Form von Handlungsanweisungen bzw. Programmcode manipuliert wird. Die Verarbeitung dieser Handlungsanweisungen führt dann zu neuen (Programmier-)Ergebnissen, die dann wiederum zu Erkenntnissen interpretiert werden.

Wichtig ist an dieser Stelle zu sagen, dass dieses zyklische Vorgehen in den einzelnen Phasen in der Realität nicht immer strikt durchlaufen werden muss; Manchmal gibt es auch Schleifen oder Auslassungen dieser einzelnen Zwischenprodukte bzw. Aktionen. Allerdings bietet dieses Modell eine gute Grundlage, um die Prozesse im Rahmen epistemischer Programmierprojekte, die aufgrund der Diversität der Interessen der programmierenden Person schwierig zu vereinheitlichen sind, einzuteilen und zu verstehen. Entscheidend ist hier der Aspekt, dass die Erkenntnis nicht direkt aus dem Menschen heraus geschieht, sondern nur durch die gegenseitige Interaktion mit dem digitalen Artefakt möglich wird.

## Computational Essays als Produkte Epistemischer Programmierprozesse

Um eine Nachvollziehbarkeit des resultierenden verwobenen Erkenntnis- und Programmierprozesses gewährleisten zu können, bedarf es eines passenden Dokumentationsmediums. In diesem sollten alle Zwischenprodukte der Interaktion des digitalen Artefaktes und des Menschen aufgezeichnet werden können – also Erkenntnisse, Erkenntnisinteressen, Handlungsabsichten, Handlungsanweisungen/Programmcode und (Programmier-)Ergebnisse. Hierzu bieten sich sogenannte Computational Essays an, in denen all diese Aspekte festgehalten werden können. Ein Computational Essay lässt sich als ein ausformuliertes Essay beschreiben, welches neben textuellen Beschreibungen, Interpretationen und Erläuterungen insbesondere auch manipulierbaren Programmcode enthält, der direkt im Medium ausgeführt werden kann. Dies ermöglicht es, die oben beschriebenen Zwischenprodukte aus den einzelnen Zyklen des epistemischen Programmierprozesses festzuhalten und ebenfalls direkt damit zu interagieren.

## Verwendung von Worked Examples als Unterstützung beim epistemischen Programmieren

Wie bereits erwähnt, sollen Schüler\*innen durch Epistemisches Programmieren erfahren, wie sie durch Programmieren persönlich oder gesellschaftlich relevante Sachverhalte erkunden können. In diesem Kontext ist es folglich von großer Bedeutung, diesen Ansatz für Schüler\*innen und Programmiernoviz\*innen im Allgemeinen zugänglich zu machen und Hilfen zum Bewältigen eventueller Hürden im Programmierprozess bereitzustellen. Eine wesentliche Idee in diesem Bezug stellt die Verwendung von sogenannten Worked Examples dar. Im Kontext des Epistemischen Programmierens bezeichnet ein Worked Example ein ausgearbeitetes und dokumentiertes Programmier-Beispiel, in welchem Prozesse und Methoden angewandt werden, die für das Programmierprojektes der Schüler\*innen nützlich sein könnten. Hierbei bietet sich ein Computational Essay als Form des Worked Examples an, um insbesondere auch den Programmierprozess und die einzelnen Schritte festzuhalten. Gleichzeitig können im Worked Example insbesondere vorgefertigte Progammiermethoden (Bibliotheken) eingeführt/erläutert werden, um sie für die Schüler\*innen direkt nutzbar zu machen. Durch die Verwendung eines solchen Worked Examples können sich Schüler\*innen somit an dem Programmierprozess bzw. den einzelnen Schritten aus dem Worked Example orientieren und gleichzeitig fertigen Programmcode nutzen und entsprechend des eigenen Erkenntnisinteresses anpassen. Ein Worked Example kann somit eine Blaupause für die Programmierprozesse der Schüler\*innen darstellen.

# Didaktische Kernidee: Vermittlung von Epistemischem Programmieren als Mittel zum Erkenntnisgewinn im Kontext von Datenexplorationen

Der hier beschriebene Ansatz des Epistemischen Programmierens stellt eine erkenntniszentrierte Perspektive auf das Programmieren dar, die den Schüler\*innen in diesem Unterrichtsmodul vermittelt werden soll. Auf diese Weise sollen Schüler\*innen Programmieren als ein Mittel erfahren, das man im Alltag nutzen kann, um persönlich oder gesellschaftlich relevante Sachverhalte (datengetrieben) erkunden zu können. In dieser Hinsicht lernen die Schüler\*innen, wie sie Datenexplorationsprojekte mittels Programmieren durchführen und auf diese Weise eigenen Erkenntnisinteressen nachgehen können. Dies bezieht sich einerseits auf die Sammlung und Auswertung von Zeitreihendaten (= Daten, die in einem bestimmten Messintervall wiederholt erhoben werden) entsprechend der eigenen Ziele, andererseits aber auch auf die Entwicklung von Computational Essays zur nachvollziehbaren und reproduzierbaren Dokumentation des Erkenntnis- und Programmierprozesses. Insbesondere lernen die Schüler\*innen einzelne Schritte in der Datenanalyse kennen (wie Filtern, Erstellen von Visualisierungen oder Berechnen bestimmter statistischer Kenngrößen) und in diesem Kontext, wie sich systematisch Informationen aus Daten gewinnen lassen.

Die Praktik des (datengetriebenen) epistemischen Programmierens erfolgt in diesem Unterrichtsmodul anhand der Durchführung eines eigenen Projektes im Kontext umweltbezogener Fragestellungen. Dabei entwickeln die Schüler\*innen eigene Erkenntnisinteressen, denen sie im Rahmen ihres Projektes nachgehen. In diesem Rahmen lernen sie den oben beschriebenen Prozess des Epistemischen Programmierens kennen und wie sie durch diese Interaktion mit dem digitalen Artefakt zu neuen Erkenntnissen gelangen können.

Im ersten Teil des Unterrichtsmoduls lernen die Schüler\*innen Kriterien für gute statistische Fragestellungen kennen und dabei besonders auch, wie diese entwickelt werden können, indem sie eigene Fragestellungen unter der Verwendung eines Kriterienkatalogs entwickeln und gleichzeitig anderen Schüler\*innengruppen Rückmeldung zu ihren Fragestellungen geben.

Im zweiten Teil lernen die Schüler\*innen, wie sie basierend auf ihrer final entwickelten Fragestellung geeignete Daten sammeln oder selbst erheben können. Dabei können sie entweder kennenlernen, wie Arduinos/Senseboxen durch Programmierung und Anschluss entsprechender Sensoren zur Erhebung von Umweltdaten genutzt werden können oder aber verschiedene Plattformen für (Umwelt-)Daten kennenlernen und wie hier Daten systematisch ausgewählt und heruntergeladen werden können, um sie für die eigene Datenauswertung nutzbar zu machen. Die systematische Herangehensweise an die Datensuche wird dabei vermittelt, um den eigenen Erkenntnisinteressen nachgehen zu können. Alternativ kann dieser zweite Teil auch ausgelassen werden und es kann direkt über ein Jupyter Notebook (name.ipynb) auf bestimmte Daten zu beliebigen Orten und beliebigen Zeitpunkten seit 2001 zugegriffen werden. In diesem Fall können die Daten direkt über ein Jupyter Notebook geladen und auch in diesem analysiert werden.

Im dritten Teil führen die Schüler\*innen die Datenanalyse im Kontext ihres epistemischen Programmierprojektes in einem Jupyter Notebook mit der Programmiersprache Python durch. Dabei lernen sie verschiedene Schritte und Aktionen kennen, um die gesammelten Daten einzulesen, aufzubereiten, zu filtern, zu visualisieren und anderweitig auszuwerten. Gleichzeitig lernen sie durch das Erstellen ihres Computational Essays kennen, ihren Programmier- und Erkenntnisprozess nachvollziehbar und reproduzierbar zu dokumentieren. Zur Umsetzung hinsichtlich ihres eigenen Projektes erwerben die Schüler\*innen dabei grundlegende Fähigkeiten in der Python-Programmierung. Dies bezieht sich insbesondere auch auf Programmlesekompetenzen, die bei der Nutzung der Worked Examples benötigt werden, um den bereits bestehenden Code im Worked Example bzw. aus den dort verwendeten Bibliotheken nachvollziehen zu können und anschließend für das eigene Projekt nutzbar zu machen. Ebenfalls werden grundlegende Python-Konstrukte (insbesondere im Rahmen eines kurzen Python-Grundkurses) vermittelt, welche die Schüler\*innen in ihrer Datenauswertung anwenden können.

Im vierten Teil tauschen sich die einzelnen Schüler\*innengruppen hinsichtlich ihrer entwickelten Computational Essays aus, sodass sie sich in die jeweils anderen Computational Essays und die darin durchgeführte Datenexploration einarbeiten und diese nachvollziehen und bewerten. Auch hier werden insbesondere auch die Programmlesekompetenzen der Schüler\*innen gefördert. Außerdem lernen sie durch den Austausch mit den anderen Gruppen weitere Beispiele für epistemische Programmierprojekte kennen.

Im fünften Teil reflektieren die Schüler\*innen die Perspektive des Epistemischen Programmierens und die Rolle von Datenanalysen im Kontext der Exploration persönlicher Interessen. In dieser Phase soll dabei auch die Interaktion zwischen programmierender Person und digitalem Artefakt herausgearbeitet und diskutiert werden, wobei besonders die Rollen dieser beiden Akteur\*innen analysiert werden sollen. Zudem entwickeln die Schüler\*innen in dieser Phase weitere Ideen und Fragestellungen, die im Rahmen von datengetriebenen epistemischen Programmierprojekten adressiert werden können.

# Lernvoraussetzungen

Dieses Unterrichtsmodul setzt keine besonderen Vorkenntnisse der Schüler\*innen voraus. Grundlegende Erfahrungen im Umgang mit dem Computer und insbesondere hinsichtlich des Umgangs mit Dateien (kopieren, ausschneiden, löschen etc.) sowie Grundkenntnisse in der Interpretation von statistischen Kennzahlen (Mittelwert, Standardabweichung, ggf. auch Korrelation) und Diagrammen (Diagramme lesen können und Informationen daraus ableiten können) sind wünschenswert, sodass Schüler\*innen ihre gesammelten Daten verwalten bzw. durch ihre Datenauswertungen zu eigenen Erkenntnissen gelangen können. Zudem sind erste Erfahrungen mit der Programmiersprache Python und Jupyter Notebooks hilfreich, allerdings werden benötigte Grundkenntnisse auch im Rahmen dieses Moduls vermittelt.

# Ziele

## Übergeordnete Ziele dieses Unterrichtsmoduls:

1. (Grobziel) Die Schüler\*innen lernen datengetriebenes epistemisches Programmieren als Mittel zur Datenexploration im Kontext eines persönlich relevanten umweltbezogenen Sachverhalts kennen.
2. (Feinziele) Die Schüler\*innen führen im Kontext einer eigenen umweltbezogenen Forschungsfrage ein datengetriebenes epistemisches Programmierprojekt als Epistemic Agents[[2]](#footnote-3) durch, indem sie (eigene) Daten sammeln und auswerten sowie die Auswertungen interpretieren und damit Erkenntnisse hinsichtlich einer persönlich oder gesellschaftlich relevanten Fragestellung entwickeln.

## Ziele der Teilabschnitte des Unterrichtsmoduls

* **Teil 1: Erarbeitung initialer Fragestellungen/Forschungsinteressen**
  + Die Schüler\*innen lernen die Exploration von Daten als Methode zur objektiven Beantwortung von persönlich oder gesellschaftlich relevanten Fragestellungen kennen.
  + Die Schüler\*innen kennen Kriterien guter und beantwortbarer statistischer Fragestellungen.
  + Die Schüler\*innen entwickeln statistische Fragestellungen für ein datengetriebenes epistemisches Programmierprojekt, indem sie die zuvor kennengelernten Kriterien befolgen.
  + Die Schüler\*innen bewerten von anderen Schüler\*innengruppen entwickelte Forschungsfragen und geben kriteriengeleitetes Feedback und entwickeln Verbesserungsvorschläge.
  + Die Schüler\*innen identifizieren zu untersuchende Variablen, die in ihren entwickelten Forschungsfragen enthalten sind.
* **Teil 2: Daten sammeln**

1. **Daten mithilfe eines Messinstrumentes selbst erheben**

* Die Schüler\*innen entwickeln einen Plan zur Erhebung von Daten, die ihnen eine Exploration in Bezug zu ihrer zuvor formulierten Forschungsfrage ermöglichen.
* Die Schüler\*innen entwickeln ein Messinstrument zur Erhebung von Daten, indem sie eine Sensebox/einen Arduino entsprechend aus passenden Komponenten und Sensoren zusammenbauen.
* Die Schüler\*innen entwerfen und implementieren ein Programm zur Datenerhebung mit der Sensebox/einem Arduino.

1. **Daten aus verschiedenen Quellen sammeln und zusammentragen**
   * Die Schüler\*innen entwickeln einen Plan zur Sammlung von Daten, die ihnen eine Exploration in Bezug zu ihrer zuvor formulierten Forschungsfrage ermöglichen.
   * Die Schüler\*innen identifizieren geeignete Datenquellen und generieren/laden Datensätze im Hinblick auf ihre Forschungsfrage

* **Teil 3: Datenexploration und verwobener Erkenntnis- und Programmierprozess**
  + Die Schüler\*innen kennen wichtige Methoden zum Einlesen, Aufbereiten Filtern, Visualisieren und Auswerten von Zeitreihendaten sowie grundlegende Python-Befehle, um diese Methoden hinsichtlich ihres eigenen Projektes anzuwenden bzw. zu kombinieren.
  + Die Schüler\*innen nutzen das zur Verfügung gestellte Worked Example, indem sie Code daraus adaptieren und erweitern und die Sequenzierung - angepasst für ihr eigenes Projekt – übernehmen.
  + Die Schüler\*innen führen die wesentlichen Schritte „Daten einlesen“, „Daten aufbereiten“, „Daten filtern“, „Daten visualisieren und auswerten“ im Datenexplorationsprozess aus.
    - Die Datenaufbereitung besteht dabei mindestens aus der Vergabe der richtigen Datentypen sowie aus dem Setzen des Index.
    - Bei der Visualisierung und Auswertung der Daten können die Schüler\*innen zwischen verschiedenen Methoden und Herangehensweisen wählen, die im Worked Example gezeigt werden. Dies sind insbesondere die Visualisierung als Scatter- oder Boxplots (mit mehreren Plots), die Berechnung wesentlicher statistischer Kennwerte (min/max/mean), die Gruppierung von Datensätzen sowie die Berechnung von Korrelationen.
  + Die Schüler\*innen kennen Eigenschaften und Nutzungsweisen eines Computational Essays.
  + Die Schüler\*innen kommunizieren ihren Erkenntnis- und Programmierprozess auf nachvollziehbare und reproduzierbare Weise, indem sie schrittweise ein Computational Essay entwickeln und darin Codezellen, Code-Output-Zellen und Textzellen kombinieren.
  + Die Schüler\*innen führen epistemische Programmierprozesse aus, indem sie spiralartig neue Handlungsschritte aus ihren Erkenntnissen ableiten, diese in Form von Programmcode implementieren und vom Computer ausführen lassen.
* **Teil 4: Vorstellung und Austausch der Computational Essays**
  + Die Schüler\*innen vollziehen die in den Computational Essays der anderen Gruppen dokumentierten Erkenntnis- und Programmierprozess nach.
  + Die Schüler\*innen bewerten die Dokumentation und den Programmcode der anderen Gruppen und entwickeln begründete Anpassungs- und Erweiterungsvorschläge.
* **Teil 5: Reflexion**
  + Die Schüler\*innen bewerten die Rolle des Programmierens und der Daten in ihren Erkenntnisprozessen.
  + Die Schüler\*innen beschreiben und diskutieren die Interaktion zwischen programmierender Person und digitalem Artefakt in epistemischen Programmierprozessen.
  + Die Schüler\*innen diskutieren die Perspektive des datengetriebenen epistemischen Programmierens und die allgemeinbildende Rolle des Programmierens im Hinblick auf Erkenntnis- und Explorationsprozesse.
  + Die Schüler\*innen identifizieren geeignete Anwendungsmöglichkeiten für epistemisches Programmieren Programmierprojekte und entwickeln Ideen für eigene und persönlich oder gesellschaftlich relevante datengetriebene epistemische Programmierprojekte.

# Zentrale Leitfragen

* **Teil 1: Erarbeitung initialer Fragestellungen/Forschungsinteressen**
  + Wie lassen sich gute und beantwortbare statistische Fragestellungen für datengetriebene epistemische Programmierprojekte formulieren?
* **Teil 2: Daten sammeln**

1. **Daten mithilfe eines Messinstrumentes selbst erheben**

* Wie kann eine Sensebox/ein Arduino zusammengebaut und programmiert werden, um geeignete Daten hinsichtlich der eigenen Forschungsfrage erheben zu können?

1. **Daten aus verschiedenen Quellen sammeln und zusammentragen**
   * Wie können geeignete Daten aus vorhandenen Datenquellen passend zur eigenen Forschungsfrage gefunden werden?

* **Teil 3: Datenexploration und verwobener Erkenntnis- und Programmierprozess**
  + Wie können in Python und Jupyter Notebooks verschiedene Schritte im Datenexplorationsprozess ausgeführt werden (z.B. Daten einlesen, Datentypen festlegen, verschiedene Visualisierungen erstellen, …)?
  + Wie können Worked Examples als Unterstützung in datengetriebenen epistemischen Programmierprojekten genutzt werden?
  + Welche Aspekte muss ein Computational Essay enthalten, um epistemische Programmierprozesse nachvollziehbar und reproduzierbar zu dokumentieren?
* **Teil 4: Vorstellung und Austausch der Computational Essays**
  + Was sind Kriterien für nachvollziehbare und reproduzierbare Computational Essays?
  + Wie können Computational Essays gelesen, nachvollzogen und adaptiert/erweitert werden?
* **Teil 5: Reflexion**
  + Welche Rolle nehmen die programmierende Person und das digitale Artefakt im Rahmen von epistemischen Programmierprozessen ein?
  + Welche weiteren persönlich oder gesellschaftlich relevanten Projekte gibt es, die unter der Verwendung von Daten in epistemischen Programmierprojekten adressiert werden können?

# Zusammenfassender Überblick über das Unterrichtsmodul

Abhängig von der Lernzielausrichtung sowie der zur Verfügung stehenden Zeit gibt es verschiedene Möglichkeiten, dieses Unterrichtsmodul durchzuführen. Während der Fokus im Modul auf der Durchführung einer Datenexploration liegt, kann man zuvor wahlweise eigene Daten sammeln, Daten aus verschiedenen (Online-)Quellen zusammentragen oder Wetterdaten aus dem POWER-Projekt des NASA Langley Research Center (LaRC), das durch das NASA Earth Science/Applied Science Program finanziert wird, nutzen ([https://power.larc.nasa.gov/data-access-viewer/](https://www.telekom-stiftung.de/en)). Im letzteren Fall überspringt man dann die Datensammlungsphase und benötigt daher etwas weniger Zeit für das gesamte Unterrichtsmodul.

Grob lässt sich das Unterrichtsmodul somit in folgende Teile unterteilen:

Teil 3:   
Datenexploration und verwobener Erkenntnis- und Programmierprozess

Teil 4:  
Vorstellung und Austausch der Computational Essays

Teil 5:  
Reflexion

Teil 1:   
Erarbeitung initialer Fragestellungen/Forschungsinteressen

Teil 2a   
Daten sammeln  
Daten mithilfe eines Messinstrumentes selbst erheben

Teil 2b   
Daten sammeln  
Daten aus verschiedenen Quellen sammeln und zusammentragen

Keine Daten sammeln/ zusammentragen und Daten aus dem POWER- Projekt des NASA Langley Research Center (LaRC) nutzen

In der folgenden Graphik sieht man die jeweiligen Aktivitäten, Leitfragen und Fachinhalte der einzelnen Teile.

Teil 1:   
Erarbeitung initialer Fragestellungen/Forschungsinteressen

Kriterienbasierte Entwicklung einer initialen Forschungsfrage; Austausch und Feedback zwischen den Gruppen

Wie lassen sich gute und beantwortbare statistische Fragestellungen für datengetriebene epistemische Programmierprojekte formulieren?

Kriterien für beantwortbare statistische Fragestellungen, sowie deren Formulierung, Grundlegende Idee von Datenexplorationen als Erkenntnismittel

Teil 2a   
Daten sammeln  
Daten mithilfe eines Messinstrumentes selbst erheben

Planung einer eigenen Datenerhebung, Entwicklung eines Messinstruments zur Datenerhebung, Zusammenbau und Programmierung einer Sensebox/eines Arduinos unter Verwendung geeigneter Sensoren

Wie kann eine Sensebox/ein Arduino zusammengebaut und programmiert werden, um geeignete Daten hinsichtlich der eigenen Forschungsfrage erheben zu können?

Datenerhebung mit eigener Messstation planen, Grundlegende Arduino-Programmierung, Hardware-Komponenten einer Arduino-Messstation – insbesondere Messsensoren

Teil 2b:   
Daten sammeln  
Daten aus verschiedenen Quellen sammeln und zusammentragen

Planung einer eigenen Datensammlung, Identifikation geeigneter Datenquellen, Herunterladen entsprechender Datensätze

Wie können geeignete Daten aus vorhandenen Datenquellen passend zur eigenen Forschungsfrage gefunden werden?

Datensammlung anhand von geeigneten Datenquellen planen und durchführen, Zusammentragen verschiedener Datensätze

Teil 3:   
Datenexploration und verwobener Erkenntnis- und Programmierprozess

Einführung in das Medium „Computational Essay“ und in Nutzungsweisen eines Worked Examples, Durchführung des eigenen epistemischen Programmierprojektes durch Erstellung eines Computational Essays unter Zuhilfenahme eines Worked Examples

Wie können…  
… in Python und Jupyter Notebooks Daten eingelesen, aufbereitet, gefiltert, visualisiert und ausgewertet werden?   
… Worked Examples als Unterstützung in datengetriebenen epistemischen Programmierprojekten genutzt werden?  
Welche Aspekte muss ein Computational Essay enthalten, um epistemische Programmierprozesse nachvollziehbar und reproduzierbar zu dokumentieren?

Computational Essays als Medium zur Dokumentation von Erkenntnis- /Programmierprozessen in Epistemischen Programmierprojekten, Nutzung von Worked Examples als Orientierung für eigene Projekte, grundl. Python-Programmierung zur Exploration von Zeitreihendaten inkl. Verw. von Bibliotheken; Einlesen Aufbereiten, Filtern, Visualisieren und Auswerten von Daten, Interpretation und Darstellung von Ergebnissen

Teil 4:  
Vorstellung und Austausch der Computational Essays

Geben von Feedback, Umsetzen von Adaptionen/Erweiterungen mithilfe von Feedback,  
Kurze Vorstellung, dann Austausch der erstellten Computational Essays, Exploration anderer Computational Essays, Veröffentlichung

Was sind Kriterien für nachvollziehbare und reproduzierbare Computational Essays?  
Wie können Computational Essays gelesen, nachvollzogen und adaptiert/erweitert werden?

Kriterien für nachvollziehbare und reproduzierbare Computational Essays, Lesen von Computational Essays und interaktiven Datenreports, (implizit) grundlegende Python-Programmierung und Verwendung von Bibliotheken

Teil 5:  
Reflexion

Reflexion des eigenen Erkenntnis- und Programmierprozesses sowie der eigenen Perspektive auf (erkenntniszentriertes) Programmieren

Welche Rolle nehmen Mensch/das digitale Artefakt im Rahmen von epistemischen Programmierprozessen ein?  
Welche weiteren persönlich oder gesellschaftlich relevanten Projekte gibt es, die durch Datenauswertungen in ep. Programmierprojekten adressiert werden können?

Analyse der Rolle von Mensch und digitalem Artefakt im Programmierprojekt, Übertragungauf andere Anwendungsmöglichkeiten, Bewusstmachen der individuellen Perspektive auf Programmieren als ein Werkzeug zum Erkenntnisgewinn

x

x

Leitfrage

Fachinhalt

Aktivität

# Überblick über den Unterrichtsverlauf

In der folgenden Tabelle wird der Unterrichtsverlauf beschrieben. In der Spalte „Inhalt“ finden sich Beschreibungen des Unterrichtsverlaufs mit didaktischen Bemerkungen. Zu den jeweiligen Phasen des Unterrichtsmoduls werden in der Tabelle außerdem die Ziele sowie die verwendeten Materialien angegeben.

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Inhalt** | **Ziele** | **Material** |
| **Teil 1: Erarbeitung initialer Fragestellungen/Forschungsinteressen** | | | |
| **1.1** | **Einführung in den Kontext und Problematisierung:**  In dieser ersten Phase des Unterrichtsmoduls soll in den Kontext – umweltbezogene Datenanalysen – eingeführt werden, indem beispielhafte aktuelle Fragestellungen diskutiert und aufgezeigt werden. Dabei soll insbesondere auch auf die Rolle von Daten und Datenanalysen diskutiert werden.  Als Anregung für die Diskussion können die Schüler\*innen sich über mögliche Forschungsfragen und -vorhaben im Rahmen von datenbasierten Umweltanalysen informieren, indem sie Texte zu entsprechenden Studien lesen und sich über Ergebnisse, verwendete Daten und Erkenntnisse dieser Studien austauschen.  Dies kann anhand der Texte, die im Material „Übersicht über Texte und Artikel zum Thema „Datenbasierte Umweltanalyse““ verlinkt sind, gemacht werden. Hier tauschen sich die Schüler\*innen zu zweit anhand des Arbeitsblattes „Austausch über umweltbasierte Studie“ über einen dieser Texte aus. Die Ergebnisse werden dann im Plenum ausgetauscht, woran sich eine Diskussion zur Leitfrage „Welche Rolle spielen Daten bei der Beantwortung umweltbezogener Fragestellungen und woraus kann man in diesem Kontext Schlüsse ziehen?“ anschließt.  Didaktischer Kommentar:  In dieser Phase sollen Schüler\*innen für das Thema der umweltbezogenen Datenexploration sensibilisiert werden. Insbesondere sollen ihnen Beispiele aufgezeigt werden, zu welchen Themen oder Fragestellungen Datenexplorationen durchgeführt werden können. Dabei sollen sie bereits erste Ideen für eigene Fragestellungen entwickeln, die sie dann später im eigenen Projekt umsetzen. Wichtig ist dabei, dass die Schüler\*innen basierend auf den in dieser Phase vorgestellten Möglichkeiten bereits erste Ideen für (Forschungs-)Interessen identifizieren, die entweder ähnlich zu den Themen der hier behandelten Studien sind oder ggf. auch ganz andere umweltbezogene Aspekte thematisieren.  Um in das Thema und die Möglichkeiten von Datenexplorationen zur Beantwortung umweltbezogener Fragestellungen an dieser Stelle einzuführen, sollte in der abschließenden Diskussion besonders auf die folgenden Aspekte „hingearbeitet“ werden:   * Aus Datenanalysen und insbesondere Visualisierungen lassen sich Erkenntnisse zu umweltbezogenen Fragestellungen gewinnen. * Man muss sicherstellen, dass die Daten a) „korrekt“ und b) „repräsentativ“ sind, das heißt, a) die Daten unterliegen keinen Messfehlern und b) die Daten bilden die Situation ausreichend ab, insbesondere werden nicht nur Spezialfälle betrachtet. * Durch eigene Datenauswertungen kann man die Ergebnisse verifizieren oder auch erweitern.   Um auf diese Aspekte hinzuarbeiten, könnte man beispielsweise folgende Einstiegsfrage wählen: „Worauf sollte man bei der Durchführung einer Datenexploration zu einem bestimmten Thema achten?“ | * Entwicklung von Interesse und Motivation für die Durchführung umweltbezogener Datenexplorationen * Exploration der Rolle von Daten und Datenanalysen/-visualisierungen für die Einschätzung umweltbezogener Fragestellungen/Sachverhalte * Aufzeigen von Beispielen für umweltbezogene Fragestellungen, die durch Datenexplorationen behandelt/beantwortet werden können | Übersicht über Texte und Artikel zum Thema „Datenbasierte Umweltanalyse“ (Uebersicht\_Zeitungsartikel\_Umweltfragen)  AB1\_Austausch\_Studie |
| **1.2** | **Entwicklung einer persönlichen Fragestellung**  In dieser Phase entwickeln die Schüler\*innen eine persönliche Fragestellung, die sie in ihrem Projekt beantworten möchten. Dabei können sie sich entweder an den in Phase 1.1 betrachteten Studien orientieren und beispielsweise eine Replikationsstudie durchführen oder aber einen ganz anderen umweltbezogenen Fokus setzen. Wichtig ist bei der Forschungsfrage, dass die Schüler\*innen die Möglichkeiten der Datenerhebung oder der bereits zur Verfügung stehenden Daten insofern berücksichtigen, als dass die Forschungsfrage mit diesen Daten beantwortet werden soll. Es müssen also passende Daten für die Forschungsfrage gesammelt/gefunden werden. Da die spätere angeleitete Datenauswertung auf sogenannten Zeitreihendaten (= Daten, die in einem bestimmten Messintervall wiederholt erhoben und mit einem Zeitstempel versehen werden) aufbaut, sollten die Schüler\*innen Ihre Forschungsfrage so planen, dass sie sie mit Zeitreihendaten beantworten können.  Zunächst finden sich die Schüler\*innen in Kleingruppen (bestenfalls 4 Schüler\*innen pro Gruppe) zusammen und erarbeiten dann unter Verwendung des Arbeitsblattes „Entwicklung einer Fragestellung“ eine eigene Fragestellung für ihr eigenes Umweltexplorationsprojekt.  Dabei werden ihnen zu Beginn 6 Kriterien/Fragen genannt, die sie hinsichtlich ihrer Fragestellung beachten sollen. Anschließend entwickeln die Schüler\*innengruppen ein bis zwei erste Ideen für eine Fragestellung, zu der sie anschließend kurz erläutern sollen, wieso diese für sie bedeutsam ist/sind. Ebenfalls sollen sie bereits festhalten, welche Größen zur Beantwortung der Frage erhoben werden müssen.  Anschließend tauschen immer jeweils zwei Gruppen ihre Arbeitsblätter aus und erhalten von der jeweils anderen Gruppe eine Rückmeldung, wobei die andere Gruppe die Fragestellung(en) hinsichtlich der oben auf dem Arbeitsblatt beschriebenen Kriterien prüft und der Gruppe zusätzlich schriftlich Feedback gibt. Anschließend überarbeiten beide Gruppen jeweils ihre eigenen Fragestellungen basierend auf dem Feedback der anderen Gruppe und halten diese Fragestellung auf ihrem Arbeitsblatt fest.  Didaktischer Kommentar:  Sollten die Schüler\*innengruppen Schwierigkeiten beim Finden einer Forschungsfrage haben, könnten sie sich zusätzlich an folgenden erste/grobe Beispielfragen orientieren:   * Wie unterscheidet sich der Feinstaubgehalt in meiner Stadt an stark befahrenen Straßen und verkehrsberuhigten Zonen? * Wie verhält sich der CO2-Gehalt in unserem Klassenzimmer und wie verhält er sich unter verschiedenen Lüftungsstrategien? * Wie unterscheidet sich das Wetter an verschiedenen Orten auf der Welt? * Wie hat sich das Wetter in den letzten 20 Jahren in meiner Stadt/an einem bestimmten Ort verändert? | * Entwicklung einer eigenen Fragestellung für das Projekt * Identifikation von zu erhebenden Variablen bzw. Daten zur Beantwortung der Fragestellung * Kennenlernen von Kriterien und Vorgehensweisen zur Entwicklung einer statistisch überprüfbaren Fragestellung | AB2\_EntwicklungeinerFragestellung |
| **Teil 2: Daten sammeln**  *Hinweis: In diesem Teil des Unterrichtsmoduls kann man sich entscheiden, Zeitreihendaten selbst mithilfe eines eigenen Messinstrumentes zu erheben – 2a – oder bereits vorhandene Zeitreihendaten aus verschiedenen Quellen zusammenzutragen – 2b. Ebenfalls ist es möglich, bestimmte Daten (Temperatur, Luftfeuchtigkeit, Niederschlag, Windgeschwindigkeit) zu beliebigen Orten und beliebigen Zeiträumen seit 2001 direkt in ein Jupyter Notebook einzulesen. In diesem Fall kann man direkt mit Teil 3.1 weitermachen und die Schüler\*innengruppen nach dem Durchlaufen des Tutorials im Unterordner „Umgebung NASA-Daten“ und dort im Jupyter Notebook Computational\_Notebook\_NASA-Data.ipynb arbeiten lassen.* | | | |
| **Teil 2a: Daten sammeln – Daten mithilfe eines Messinstrumentes selbst erheben**  *Hinweis: Im Folgenden wird das Unterrichtsmodul im Hinblick auf die Nutzung der Sensebox beschrieben und in dieser Hinsicht, wie die Schüler\*innengruppen mithilfe der Sensebox eigene umweltbezogene Daten erheben können. Dabei fokussiert diese Beschreibung die Verwendung eines Sensebox-Bausatzes bestehend aus dem Board, (verschiedenen) Sensoren, SD-Karten-Modul, Display und ggf. anderen Komponenten. Einen Beispielbausatz stellt die SenseboxMini (*[*https://sensebox.de/de/products-mini*](https://sensebox.de/de/products-mini)*) dar. Allerdings können natürlich auch andere Messinstrumente verwendet oder selbst programmiert werden, wie beispielsweise ein Arduino (*[*https://docs.arduino.cc/*](https://docs.arduino.cc/)*) oder ein MicroBit (*[*https://microbit.org/get-started/user-guide/introduction/*](https://microbit.org/get-started/user-guide/introduction/)*).* | | | |
| **2a.1** | **Kennenlernen der Sensebox als programmierbares Messinstrument**  In dieser Phase lernen die Schüler\*innen die Sensebox kennen und entwickeln erste eigene Programme, indem sie die Blockprogrammierumgebung für die Sensebox nutzen.  Dabei erhält jede Gruppe zunächst eine eigene Sensebox (samt Sensoren und anderem Zubehör), die sie später auch für die Datenerhebung nutzen können. Mithilfe der Lernkarten für die Sensebox (insbesondere die Lernkarten SB01-SB07, SB12, GI01, GI02, GI03) erfahren die Gruppen zunächst, welche Komponenten die Sensebox enthält, wie sie zusammengebaut werden kann und wie sie mit der Blockprogrammierumgebung programmiert werden kann.  Anschließend sollen die Gruppen ihre Sensebox so zusammenbauen und in der Blockly-Umgebung so programmieren, dass auf dem Display die aktuelle Temperatur ausgegeben wird, die vom Temperatursensor gemessen wird. Eine Beispiellösung dafür ist in der Datei „Lsg\_Temperaturmessung“ zu finden.  Didaktischer Kommentar:  Das Ziel dieser Phase ist es, dass die Schüler\*innen die Sensebox zunächst erst einmal frei erkunden und kennenlernen können sowie am Ende ein erstes kleineres Projekt erstellen, um nachvollziehen zu können, wie man die Sensebox zusammenbauen und programmieren kann.  Nachfolgend sind ein paar Hinweise zur Einrichtung der Blockly-Umgebung und zum Übertragen des fertigen Blockly-Programms auf die Sensebox zu finden. Diese sind auch für die nachfolgenden Phasen relevant. Detailliertere Informationen findet man zudem in der Blockly-Dokumentation.  *Wichtig für die Programmierung in der Blockly-Umgebung:*  In der Blockprogrammierumgebung muss zunächst das richtige Board ausgewählt werden, welches in der Sensebox verwendet wird. Dies sollte den Schüler\*innen zu Beginn dieser Phase einmal mitgeteilt werden.  *Wichtig für das Übertragen des Programmcodes:*  Sobald das Programm in der Blockly-Umgebung fertig geschrieben wurde, muss es noch auf die Sensebox übertragen werden. Wenn man am Computer arbeitet, muss man dazu das Programm herunterladen und anschließend auf die Sensebox „ziehen“. Dies wird in der Anleitung unter der folgenden URL erläutert: <https://docs.sensebox.de/docs/boards/mcus2/mcus2-kompilieren/#kompilieren>  Auch mit dem Tablet kann das in Blockly erstellte Programm auf die Sensebox übertragen werden. Wie das funktioniert, wird in der folgenden Anleitung erklärt: <https://sensebox.de/de/app.html> | * Kennenlernen der Sensebox (und der Messsensoren) als Messinstrument für Umweltdaten * Kennenlernen der Programmierumgebung Blockly zur Programmierung der Sensebox | Lernkarten für die Sensebox (Link: <https://sensebox.de/de/lernkarten-mcu>)  Blockprogrammierumgebung Blockly (Link: <https://blockly.sensebox.de/>)  Dokumentation von Blockly (Link: <https://docs.sensebox.de/category/blockly/>)  Beispiellösung für die erste Temperaturmessung (Lsg\_Temperaturmessung) |
| **2a.2** | **Entwicklung einer ersten Messstation**  In dieser Phase entwickeln die Gruppen eine erste Umweltstation mit der Sensebox. Dabei können sie sich entweder selbstständig einen Zusammenbau und eine Programmierung überlegen, oder aber sich an der Projektbeschreibung zur „Umweltstation Mini“ orientieren. Das Ziel ist es, dass die Gruppen am Ende eine Messstation haben, die bestimmte Werte (z.B. Temperaturdaten) misst und unter Angabe der verstrichenen Zeit seit Versuchsbeginn in einer csv-Datei abspeichert.  *Hinweis zur Projektbeschreibung „Umweltstation Mini“:*  Im fertigen Programm der Projektbeschreibung ist festgelegt, dass in der erstellten Datei vor jeder Zeile mit den aktuellen Werten eine Zeile eingefügt wird, in der festgehalten ist, welche Werte gemessen werden – also z.B.:  Zeit; Temperatur; Luftfeuchte; Luftdruck  6100; 24.1; 44.2; 1013.25  Zeit; Temperatur; Luftfeuchte; Luftdruck  6101; 24.4; 42.7; 1015.42  Zeit; Temperatur; Luftfeuchte; Luftdruck  6103; 24.2; 41.9; 1011.51  …  Für die spätere Datenauswertung reicht es allerdings aus, wenn die „Überschrift“ nur einmal ganz oben in der Datei festgehalten ist – also z.B.:  Zeit; Temperatur; Luftfeuchte; Luftdruck  6100; 24.1; 44.2; 1013.25  6101; 24.4; 42.7; 1015.42  6103; 24.2; 41.9; 1011.51  …  Zudem sollen die Daten im Gegensatz zur Projektbeschreibung für die spätere Datenexploration in einer csv-Datei (und nicht in einer txt-Datei) gespeichert werden. Ein entsprechendes Programm, welches diese beiden Aspekte umsetzt und ansonsten gleich zu dem in der Projektbeschreibung ist, ist das Programm Mini-Umweltmessstation\_fuer\_Datenexploration.xml. Dieses kann man in der Blockly-Umgebung durch einen Klick auf den Button „Projekt öffnen“ (rechts oben in der Button-Leiste) öffnen und dann auf die Sensebox übertragen.  Falls der Umweltsensor BME680 nicht vorhanden ist, kann man natürlich auch andere Daten messen. In dem Programm Mini-Umweltmessstation\_mitHDC1080Sensor.xml werden zum Beispiel Temperatur- und Luftfeuchtigkeitsdaten mit dem HDC1080-Sensor gemessen und gespeichert.  *Weiterer Hinweis bei Nutzung der senseBox MCU als Board (anstelle der senseBox MCU:mini oder der senseBox MCU-S2):*  Um die Daten auf die SD-Karte zu übertragen, muss der XBEE-Steckplatz 2 für den SD-Kartenleser verwendet werden.  Didaktischer Kommentar:  In dieser Phase bereiten die Gruppen eine erste Vorlage für die Programmierung der Sensebox zu einer Messstation vor. Dabei steht im Vordergrund, dass sie die wesentlichen Funktionalitäten (aktuelle Daten auf dem Display anzeigen und Daten auf eine SD-Karte übertragen) umsetzen können, sodass sie basierend auf ihrem Ergebnis in dieser Phase ein eigenes Messinstrument für ihre eigene Forschungsfrage erstellen können. Die Projektbeschreibung der „Umweltstation Mini“ kann ihnen dabei als Orientierung und Hilfestellung dienen. Natürlich können sie aber auch andere Ideen bereits in dieser Phase einbringen. | * Entwicklung einer ersten Messstation zur Messung und Speicherung von Umweltdaten | Blockprogrammierumgebung Blockly (Link: <https://blockly.sensebox.de/>)  Dokumentation von Blockly (Link: <https://docs.sensebox.de/category/blockly/>)  Projektbeschreibung „Umweltstation Mini“ (Link:  <https://sensebox.de/projects/de/2022-12-19-messstation-mini>)  XML-Dateien zur Datensammlung für die spätere Datenexploration:   * Mini-Umweltmessstation\_fuer\_Datenexploration * Mini-Umweltmessstation\_mitHDC1080Sensor |
| **2a.3** | **Programmierung der Sensebox für die Datenerhebung**  Basierend auf den in der vorherigen Phase entwickelten Messstationen und Programmen entwickeln die Schüler\*innengruppen in dieser Phase ein auf ihre eigene Forschungsfrage (aus Phase 1) zugeschnittenes Messinstrument. Um die Datenerhebung in einem ersten Schritt planen zu können, sollten die Gruppen erneut das AB2 Entwicklung einer Fragestellung beachten. Dort haben sie in Phase 1.2 einerseits ihre finale Forschungsfrage festgehalten und zudem konkret die Größen festgehalten, die zur Beantwortung der Forschungsfrage(n) benötigt werden. Je nach Forschungsfrage kann es nötig sein, dass mehrere Datenerhebungen (z.B. an unterschiedlichen Orten) durchgeführt werden müssen.  Die Planung ihrer Datenerhebung sollen die Gruppen auf dem AB3a\_Planung\_Datenerhebung festhalten. Dies sollen sie dann später nutzen, um neben ihrem Datenexplorationsprozess auch den Prozess der Datenerhebung zu dokumentieren. Nach dem Ausfüllen der Fragen 1-4 des ABs sollen die Gruppen sich auf Basis dieser ihre geplante Datenerhebung vorstellen. Darauf aufbauend sollen sie identifizieren, ob es andere Gruppen mit ähnlichen Interessen gibt, sodass man ggf. Daten austauschen könnte. Dies soll dann in der Frage 5 auf dem AB festgehalten werden. Wenn beispielsweise mehrere Gruppen Daten an einer Hauptverkehrsstraße aufnehmen wollen, können sie ihre Daten miteinander austauschen, sodass nur eine Sensebox dort Daten erheben muss. Die Sensebox der anderen Gruppe kann dann gleichzeitig weitere benötigte Daten aufzeichnen.  Nach dem Ausfüllen des ABs beginnen die Gruppen mit der Erstellung ihrer Messstationen. Dabei können sie zunächst ein Messinstrument für die erste Datenerhebung entwickeln und dieses dann schon einmal aufstellen und Daten sammeln lassen, während sie das Messinstrument für eine eventuelle zweite und/oder dritte Datenerhebung programmieren.  Wenn die Vorbereitungen für die Programmierung aller Messstationen abgeschlossen ist und während die einzelnen Messstationen die Daten sammeln, können die Schüler\*innen bereits damit beginnen, Python mithilfe des Python-Crashkurses (siehe Phase 3.1) kennenzulernen.  Nach Abschluss der Datensammlung müssen die von der Sensebox gesammelten Daten noch zusammengetragen und abgespeichert werden.  Didaktischer Kommentar:  In dieser Phase sollen die Gruppen eigenständig ein Messinstrument für die Datenerhebung zu ihrer Forschungsfrage entwickeln. Dabei können sie das Programm aus der vorangegangenen Phase als Ankerpunkt nehmen, von dem aus sie weiterarbeiten können. Natürlich können sich die Gruppen auch gegenseitig unterstützen – beispielsweise hinsichtlich der Verwendung bestimmter Sensoren.  *Hinweise zur Programmierung der Sensebox:*   * Abhängig von jeweils angestrebten Lernzielen…   1. können die Schüler\*innen die Sensebox eigenständig in ihren Gruppen programmieren   2. kann die Lehrkraft eine „Codebasis“ vorgeben, die die Schüler\*innen dann erweitern   3. kann die Lehrkraft den Code bereits vollständig vorgeben, sodass die Schüler\*innen diesen auf ihre zusammengebaute Sensebox übertragen können * Informationen zur Programmierung der Sensebox gibt es auf der Dokumentationswebsite: <https://docs.sensebox.de/> * Die Programmierung kann wahlweise mithilfe der blockbasierten Sprache Blockly  oder mit der Arduino IDE (textbasierte Programmierung) erfolgen.   1. Informationen zu Blockly: <https://docs.sensebox.de/docs/boards/mcu/mcu-erster-sketch>   2. Informationen zur Arduino IDE: <https://docs.sensebox.de/docs/category/arduino> * Wichtig ist, dass bei der Programmierung der Sensebox auch die Zeit mit gespeichert wird, damit später in der Datenexploration geeignete Diagramme erstellt werden können, bei denen die Zeit auf der x-Achse eingetragen werden kann. Hierbei kann man entweder – wie auch in der vorangegangenen Phase – die Zeit seit Messbeginn in der csv-Datei speichern oder aber auch mithilfe einer (externen) Clock die aktuelle Uhrzeit abspeichern. Informationen dazu gibt es hier: <https://docs.sensebox.de/docs/hardware/accessoires/rtc-modul/>  bzw.  <https://edu.books.sensebox.de/de/grundlagen/uhr.html> * Die Daten, die von der Sensebox erhoben werden, können entweder – wie in der vorherigen Phase erläutert – auf einer SD Karte gespeichert werden oder über Wifi oder das LoRaWan-Netz an die OpenSenseMap gesendet und von dort später heruntergeladen werden. Informationen zum Senden der Daten via Wifi sind hier zu finden:  <https://docs.sensebox.de/docs/hardware/wifi> bzw. <https://sensebox.de/projects/de/2024-01-10-iotmesstation_s2>  Informationen zum Senden via LoRaWan findet man hier: <https://docs.sensebox.de/docs/hardware/bee/lora-bee/> bzw. <https://sensebox.de/projects/de/2021-02-19-ttnv3> bzw. <https://sensebox.de/projects/de/2019-05-10-lora-osem-tago> bzw. <https://sensebox.de/docs/informatikunterricht_lorawan.pdf> * Im Allgemeinen kann die Programmierung der Sensebox wieder in der Blockly-Umgebung erfolgen. Alternativ oder auch aufbauend auf einer ersten Version des Programms kann bei Bedarf die Programmierung in Arduino-Code erfolgen (ggf. mit vorheriger Einführung). Dazu benötigt man die Arduino IDE, welche zunächst für die Nutzung der Sensebox vorbereitet werden muss. Informationen darüber erhält man hier: <https://docs.sensebox.de/docs/category/arduino>   Man kann dann den äquivalenten Code zu einem Blockly-Programm direkt aus der Blockly-Umgebung (auf der rechten Seite) herauskopieren und anschließend in der Arduino IDE weiter anpassen oder erweitern.  *Hinweise zum Zusammentragen der gesammelten Daten mit der Sensebox:*   * Es gibt verschiedene Optionen, wie die Daten, die von der Sensebox gesammelt werden, gespeichert werden. Diesbezüglich legt man sich bei der Programmierung der Sensebox fest:   1. auf einer SD-Karte an der Sensebox als csv-Datei   2. auf der OpenSenseMap  1. Speichert man die Daten lokal auf einer SD-Karte, die über ein entsprechendes Modul an der Sensebox befestigt ist, kann man die entsprechende csv-Datei anschließend von der SD-Karte auf den Computer kopieren. 2. Wurden die Daten auf der OpenSenseMap gespeichert (z.B. wenn man die Übertragung via WiFi oder LoRaWan umgesetzt hat), kann man die Daten der eigenen Station (und ggf. auch noch weiterer Stationen) über die OpenSenseMap herunterladen. Informationen dazu gibt es hier: <https://tutorials.opensensemap.org/analysis/analysis-download/>  * *Wichtig*: Am Ende dieser Phase sollten die Schüler\*innen alle relevanten Daten als csv-Datei abgespeichert haben, damit sie diese in der nächsten Phase in die Datenexplorationsumgebung einlesen und explorieren können. | * Entwicklung eines eigenen Messinstruments zur Datenerhebung hinsichtlich der eigenen Forschungsfrage | Blockly (Link: <https://docs.sensebox.de/category/blockly/>)  AB3a\_Planung\_Datenerhebung |
| **Teil 2b: Daten sammeln – Daten aus verschiedenen Quellen sammeln und zusammentragen** | | | |
| **2b.1** | **Organisation der Datensammlung**  In dieser Phase sollen die Gruppen ihren Datensammlungsprozess strukturieren, indem sie festhalten, nach welchen Zeitreihendaten (= Daten, die in einem bestimmten Messintervall wiederholt erhoben werden) sie suchen wollen. Dabei sollen sie anhand des Arbeitsblattes AB3b\_Planung\_Datensammlung (zunächst nur Fragen 1-4) die Art, die Zeiträume und Orte der Daten, die sie in ihrer Datenexploration verwenden, festhalten und dies anschließend im Plenum den anderen Gruppen vorstellen.  Anschließend können die Gruppen überlegen, welche andere Gruppe ähnliche Interessen – besonders hinsichtlich der Daten – verfolgt, sodass sich ein späterer Austausch von Daten lohnen könnte. Dies kann dann anhand der Frage 5 auf dem Arbeitsblatt festgehalten werden. | * Strukturierung des Datensammlungsprozesses | AB3b\_Planung\_Datensammlung  (Seite 1) |
| **2b.2** | **Recherche zu verschiedenen Datenquellen**  In dieser Phase recherchieren die Gruppen nach geeigneten Datenquellen für die Datenexploration zu ihrer auf AB2 festgehaltenen Forschungsfrage. Als erste Ideen können dabei die Quellen in der *Übersicht über mögliche Datenquellen* verwendet werden. Wichtig ist, dass in dieser Phase zunächst erst einmal möglichst viele mögliche Datenquellen identifiziert werden, die dann auf der zweiten Seite von AB3b festgehalten werden. Anschließend können die Gruppen ihre Quellen untereinander austauschen, wodurch alle Gruppen ggf. noch weitere Datenquellen erhalten und bei sich auf dem Arbeitsblatt ergänzen können.  Didaktischer Kommentar:  Natürlich können auch weitere als die in der Übersicht festgehaltenen Datenquellen den Gruppen vorgeschlagen werden. Zudem können die Gruppen eine Suchmaschine wie beispielsweise Ecosia oder DuckDuckGo zum Suchen nach Datensätzen verwenden. Hier empfiehlt es sich, Suchanfragen auf Englisch zu stellen, da es viele Datensätze auf englischsprachigen Seiten gibt. | * Identifikation geeigneter Datenquellen für die eigene Forschungsfrage | AB3b\_Planung\_Datensammlung  (Seite 2)  Übersicht über mögliche Datenquellen: Uebersicht\_moegliche\_Datenquellen |
| **2b.3** | **Download und Abspeichern der relevanten Daten**  In dieser Phase nutzen die Schüler\*innengruppen die zuvor identifizierten und auf AB3b festgehaltenen Datenquellen, um geeignete Daten für die Datenexploration hinsichtlich ihrer Forschungsfrage herunterzuladen. Dabei können erst einmal auch mehrere Datensätze zum gleichen Merkmal (z.B. Temperatur) heruntergeladen werden. Eine Auswahl kann dann später bei der Datenexploration getroffen werden. Um die Datenquelle zu den jeweiligen Daten festzuhalten (auch für eine spätere Dokumentation und damit für die Transparenz des Datenexplorationsprozesses), sollen die Gruppen auf der Rückseite von AB3b die jeweiligen heruntergeladenen Datensätze festhalten und zusätzlich die Quelle (samt Link und Zugriffsdatum) festhalten.  Am Ende dieser Phase können sich die Gruppen noch einmal über die heruntergeladenen Datensätze austauschen und ggf. Datensätze austauschen.  Didaktischer Kommentar:  Beim Download der Daten ist darauf zu achten, dass diese im csv-Format vorliegen oder in einem Format, welches unkompliziert in das csv-Format umgewandelt werden kann. Beispielsweise können txt-Dateien recht einfach durch Veränderung der Dateiendung umgewandelt werden. Um die Daten nach dem Download kurz zu überprüfen, lohnt es sich, die Datensätze kurz zu öffnen. In der ersten Zeile sollte dann ein Header stehen, in dem alle Variablen des Datensatzes aufgelistet sind. Anschließend sollten die Daten jeweils in Zeilen folgen, wobei eine Zeile jeweils einen Zeitpunkt darstellt, zu dem die in der Zeile enthaltenen Daten aufgenommen wurden. Hierbei ist darauf zu achten, dass auch ein Zeitstempel bzw. ein Zeitpunkt oder Ähnliches festgehalten ist. | * Sammeln und Zusammentragen von Daten-sätzen hinsichtlich der eigenen Forschungsfrage | AB3b\_Planung\_Datensammlung  (Seite 2) |
| **Teil 3: Datenexploration und verwobener Erkenntnis- und Programmierprozess** | | | |
| **3.1** | **Kennenlernen der Programmierumgebung:**  Bevor die Schüler\*innen ihre eigenen Datenexplorationsprozesse beginnen, ist es hilfreich, dass sie die Programmierumgebung (Juypter Notebooks) und die Programmiersprache (Python) erst einmal kennenlernen (siehe auch Glossar, Abschnitt 9.1). Dazu durchlaufen sie selbstständig einen Einführungskurs in Jupyter Notebooks. Dieser Kurs besteht aus insgesamt sieben Jupyter Notebooks, in denen sich die Schüler\*innen mit verschiedenen wesentlichen Aspekten und Konstrukten (u.A. Variablen, Fallunterscheidungen, Wiederholungen, Funktionen, Verwendung von DataFrames) von Jupyter Notebooks und der Programmiersprache Python beschäftigen.  *Hinweise zur Arbeit in der Jupyter Notebook Umgebung:*  Beim ersten Aufruf der Jupyter Notebook Umgebung (über <https://go.upb.de/ep-umweltanalyse>) kann man in der Anmeldemaske einen Username und ein Passwort frei vergeben. Die bearbeiteten und ggf. neu erstellten Jupyter Notebooks bleiben dann im Account gespeichert, wenn man in den einzelnen Jupyter Notebooks auf das Speichersymbol in der Leiste klickt oder auf „File -> Save and Checkpoint“. Nach erneutem Anmelden mit dem festgelegten Username und Passwort kann man dann später wieder auf diese eigenen Dokumente zugreifen.  Weitere Informationen zum Aufbau und den Inhalten der Jupyter Notebook Umgebung sind unter Abschnitt 9.1.4 zu finden.  Didaktischer Kommentar:  Mit den hier beschriebenen Jupyter Notebooks erarbeiten sich die Schüler\*innen ein Grundwissen über die Verwendung von Jupyter Notebooks und Python als Programmiersprache. Die enthaltenen Jupyter Notebooks sollen Grundkenntnisse so weit aufbauen, dass die Schüler\*innen in der nächsten Phase bereits vorhandenen Programmcode aus den Worked Examples nutzen, adaptieren und kombinieren können. Die anderen Juypter Notebooks 5-8 sind insbesondere zur vertieften Behandlung der Programmiersprache Python hilfreich und im Kontext der Umweltanalyse optional.  Wichtig – insbesondere für die Phase 3.2 und 3.3 – ist es, dass sich die Schüler\*innen ihre erstellten Zugangsdaten merken und ggf. auch irgendwo notieren, da sie sonst keinen Zugriff mehr auf ihre Arbeitsergebnisse haben. Zudem sollten sie innerhalb ihrer Kleingruppen darauf achten, dass sie untereinander ggf. verschiedene Arbeitsergebnisse austauschen, sodass auch bei Fehlen einer\*s Schüler\*in innerhalb der Gruppe weitergearbeitet werden kann. Dazu können die Schüler\*innen die Jupyter Notebooks in der Menüleiste über „File -> Download as -> Notebook (ipynb)“ als ipynb-Datei herunterladen. Eine bereits heruntergeladene ipynb-Datei kann man – genau wie auch eine csv-Datei – über den Upload-Button in der Ordneransicht der Jupyter Notebook Umgebung wieder hochladen (z.B. auch unter einem anderen Account). Genauere Informationen hierzu gibt es im Glossar unter Abschnitt 9.1.5. | * Kennenlernen der Programmierumgebung Jupyter Notebooks * Aufbau eines Grundwissens über die Programmiersprache Python und die Umgebung Jupyter Notebooks | Jupyter Notebook Umgebung  <https://go.upb.de/ep-umweltanalyse>  Darin: Jupyter Notebooks im Unterordner Tutorial\_JupyterNotebooks\_Python |
| **3.2** | **Datenexplorationsprozess**  In dieser Phase führen die Schüler\*innen in ihren Kleingruppen ihre eigenen Datenexplorationen durch, um Antworten auf ihre auf AB2 festgehaltenen Fragestellungen zu finden. Das Ziel am Ende der Phase 3.3 ist die Entwicklung eines Computational Essays (siehe auch Abschnitte 2.3, 9.1.1 und 9.4.4), also eines Dokumentes, welches den Datenexplorationsprozess abbildet und die Programmierergebnisse zusammen mit dem Programmcode sowie der Dokumentation des Erkenntnis- und Programmierprozesses darstellt. In einem Jupyter Notebook lässt sich dies am besten durch die Aufteilung des Programmcodes in einzelne Erkenntnis- und Programmierschritte erreichen, die dann in einzelnen oder ggf. auch mehreren Code-Zellen dargestellt sind und zwischen denen jeweils Erklärungen zum Code oder den aktuellen Erkenntnisschritten sowie Interpretationen der Programmierergebnisse sowie Erkenntnisse festgehalten werden können. Ein gutes Beispiel für ein Computational Essay stellen die beiden Worked Examples dar, die sich im Unterordner „Worked Examples“ befinden und ebenfalls weiter unten in dieser Phase genauer erläutert werden.  Während in dieser Phase nun zunächst der Explorationsprozess an sich im Vordergrund steht, geht es in der Phase 3.3 dann um die Fertigstellung und ggf. um die Ergänzungen, um das eigene Jupyter Notebook zu einem Computational Essay zu machen.  Für ihre Programmier- und Erkenntnisprozesse können die Schüler\*innen wieder in derselben Jupyter Notebook-Umgebung wie in der Phase 3.1 (siehe auch Glossar, Abschnitt 9.1.4) arbeiten. Hier gibt es zwei verschiedene Arbeitsumgebungen (bzw. Unterordner), in denen die Schüler\*innengruppen die Notebooks für die Durchführung ihrer Datenexploration finden:   1. *Umgebung NASA-Daten*   Falls keine eigenen Daten gesammelt bzw. aus verschiedenen Quellen zusammengetragen wurden, kann man die Notebooks im Unterordner *Umgebung NASA-Daten* nutzen. Dort können die Schüler\*innen im Jupyter Notebook *Computational\_Notebook \_NASA-Data.ipynb* verschiedene Wetterdaten zu beliebigen Orten und für beliebige Zeiträume direkt in das Jupyter Notebook laden und dort analysieren. Die Wetterdaten stammen aus dem POWER-Projekt des NASA Langley Research Center (LaRC), das durch das NASA Earth Science/Applied Science Program finanziert wird und sind immer nahezu tagesaktuell abrufbar ([https://power.larc.nasa.gov/data-access-viewer/](https://www.telekom-stiftung.de/en)). Man kann so meistens zu allen Orten Daten von 2001 bis zum Zeitpunkt, drei Tage vor der aktuellen Zeit direkt in das Notebook laden. Wie das funktioniert wird direkt im Jupyter Notebook erklärt. Zusätzlich können sich die Schüler\*innen im Jupyter Notebook *Computational\_Notebook \_NASA-Data\_Beispielvisualisierungen* Beispielvisualisierungen anschauen und wie diese mit den „NASA-Daten“ erstellt werden können. Im Gegensatz zu den eigenen Daten – wie in b) – müssen die „NASA-Daten“ nicht zusätzlich aufbereitet werden, da das schon „im Hintergrund“ geschehen ist. Insbesondere müssen also nicht die Datentypen festgesetzt bzw. der Index festgelegt werden, wie es z.B. in den Worked Examples gezeigt wird. Diese Abschnitte der Worked Examples müssen die Schüler\*innen also nicht beachten.   1. *Umgebung eigene Daten*   Wenn die Schüler\*innen eigene Daten gesammelt oder zusammengetragen haben und diese als csv-Datei(en) abgespeichert haben, können sie diese csv-Dateien im Jupyter Notebook *Computational\_Notebook.ipynb* in ein DataFrame einlesen und dann explorieren. Wie das Einlesen der Daten funktioniert, wird im Jupyter Notebook  *Tutorial\_Daten\_einlesen* genauer erläutert. Anschließend ist oft noch eine Datenaufbereitung nötig, innerhalb derer die Datentypen mitgeteilt und der Index des DataFrames festgelegt werden muss. Wie das funktioniert, können die Schüler\*innen beispielsweise im Worked Example *ComputationalEssay\_CO2.ipynb* nachlesen, welches sich im Unterordner *Worked Examples -> CO2-Messstation* befindet. Der Ordner *Worked Examples* liegt dabei auf der obersten Ebene der Jupyter Notebook-Umgebung.  Für die Durchführung ihrer eigenen Explorations- und Programmierprozesse nutzen die Schüler\*innen ein Worked Example als Orientierung und gleichzeitig auch als Codebasis (siehe auch Eintrag im Glossar unter 9.1.2). Die Schüler\*innen können für ihre Analyse dabei einerseits die äußere Struktur des Worked Examples übernehmen bzw. hinsichtlich der eigenen Bedürfnisse adaptieren. Andererseits können sie auch konkrete „Code-Schnipsel“ oder ganze Code-Zellen aus dem Worked Example für ihr eigenes Projekt übernehmen.  Hierzu befindet sich in der Jupyter Notebook Umgebung das Worked Example „WorkedExampleCO2.ipynb“ im Unterordner „Worked Examples/Worked\_Example\_CO2-Messstation“. Dieses Worked Example zeigt im Kontext des Themas „CO2-Raumluftanalyse“ einen Datenexplorationsprozess entlang des PPDAC-Modells (siehe Glossar – Abschnitt 9.4.2) und geht dabei insbesondere auf die Themen „Daten einlesen“, „Daten filtern“, „Daten gruppieren“ sowie auf die Erstellung verschiedener Visualisierungsformen ein. Alternativ steht ein kürzeres Worked Example zum Thema der Wetteranalyse in London und Paderborn zur Verfügung, welches (ggf. mit weniger erklärenden Texten) aufzeigt, wie Daten eingelesen, gefiltert und in einem Scatter- oder Boxplot visualisiert werden können.  Ein mögliches Vorgehen für die Schüler\*innen könnte es sein, zunächst einmal die Codeschnipsel zum Einlesen der Daten aus dem Worked Example zu kopieren, in das eigene Jupyter Notebook Dokument einzufügen und an die eigenen Daten anzupassen. Anschließend könnten sie sich eine Auswertung oder eine Visualisierung aus dem Worked Example heraussuchen, von der sie sich einen ersten Überblick über den eigenen Datensatz versprechen. Auch hierzu könnten sie den Code entsprechend kopieren und anpassen. Im Anschluss an die Erstellung eines ersten Überblicks und für eine tiefergehende Analyse (zum Beispiel durch Filtern/Gruppieren der Daten) können sie sich dann vor dem Hintergrund des eigenen Erkenntnisinteresses bzw. der eigenen Forschungsfrage Anpassungen/Erweiterungen überlegen und dabei das Worked Example als „Steinbruch“ für relevanten Programmcode verwenden.  Für die Analyse können die Schüler\*innengruppen nun frei vorgehen und die Daten in „ihrem“ Jupyter Notebook explorieren, indem sie die Datenoperationen und Datentransformationen „Filtern“, „Visualisieren“, „Gruppieren“, „Berechnen von statistischen Kennwerten“ anwenden. Wie welche dieser Operationen und Transformationen verwendet wird, hängt immer stark vom jeweiligen Erkenntnisinteresse der einzelnen Vorhaben der Gruppen ab. Das Worked Example kann dabei eine Orientierung bieten, indem die Schüler\*innen geeignete Operationen und Transformationen darin entdecken, die sie in ihrem eigenen Projekt verwenden können. Die Lehrkraft kann während dieser Phase die Gruppen bei Fragen unterstützen, sollte sich allerdings im Sinne des Lenkens der Projekte weitestgehend zurückhalten, damit die Schüler\*innen eigenverantwortlich vorgehen und neue Erkenntnisse gewinnen können, was im Sinne der Entwicklung einer Epistemic Agency (siehe Abschnitt 9.4.3) ein wesentliches Ziel des gesamten Unterrichtsmoduls darstellt.  Der zeitliche Rahmen für die Datenexploration ist sehr flexibel gestaltbar. Mindestens sollte man den Schüler\*innengruppen zwei Doppelstunden/Lerneinheiten (2\*90 Minuten) Zeit gewähren, um ihre Exploration abzuschließen.  Es kann hilfreich sein, in regelmäßigen zeitlichen Abständen im Plenum eine kurze Zwischenpräsentation der einzelnen Projekte zu machen, in denen die einzelnen Schüler\*innengruppen kurz berichten, was ihr aktueller Erkenntnis- und Prozessstand ist, wo ggf. Fragen oder Probleme liegen und was sie bereits gelöst/geschafft haben. Ggf. ergeben sich auf diese Weise Möglichkeiten, dass sich einzelne Gruppen untereinander austauschen oder weiterhelfen, indem sie im Anschluss an diese Zwischenpräsentationen kurz zusammenkommen und sich beraten.  Didaktischer Kommentar:  Um die Schüler\*innen für die Durchführung ihres Datenprojektes und die Entwicklung ihres Computational Essays zu unterstützen, sollten ihnen folgende Aspekte als Vorgehensweise mit an die Hand gegeben werden:  In dieser Phase ist es wichtig, dass man den Schüler\*innengruppen Zeit und Raum für eine freie Exploration der Daten gibt. Es empfiehlt sich, zu Beginn der Exploration zunächst überblicksartig zu starten – z.B. mithilfe eines einfachen Scatter- oder Boxplots – und sich dann immer genauer hinsichtlich der eigenen Fragestellung in die Datenexploration zu vertiefen, wobei sich auch das Erkenntnisinteresse immer weiter verändern/konkretisieren kann. Dabei kann man jeweils auf vorausgehende Erkenntnisse aufbauen.  In dieser Hinsicht könnte es hilfreich sein, dass die Schüler\*innen ihre jeweiligen Erkenntnisse direkt im Jupyter Notebook festhalten und darauf aufbauend auch ihre weiteren Schritte direkt dort planen und festhalten. Auch eine Dokumentation des Programmcodes bzw. der einzelnen Code-Zellen kann bereits in dieser Phase erfolgen. Für die Erstellung eines Computational Essays (siehe Abschnitte 2.3, 9.1.1 und 9.4.4) ist dies bereits ein hilfreicher Schritt, sodass die Arbeit in der Phase 3.3 bereits hier entlastet werden kann.  Die Schüler\*innen sollten in dieser Phase darauf hingewiesen werden, das ausgewählte Worked Example zur Hilfe zu nehmen. Nicht nur können sie sich dabei an der äußeren Struktur orientieren, sondern können auch einzelne Code-Teile aus dem Worked Example kopieren, adaptieren und kombinieren. Insbesondere hinsichtlich der Nutzung bereits vorhandenen Programmcodes sollten Schüler\*innen hier ermutigt werden.  Wichtig zu erwähnen ist noch, dass die Schüler\*innengruppen darauf achten sollten, dass sie ihre bearbeiteten Jupyter Notebooks hin und wieder zwischenspeichern – insbesondere bevor sie sich aus der Jupyter Notebook Umgebung abmelden. Dies funktioniert bei einem geöffneten Jupyter Notebook, indem man in der Leiste oben auf „File“ und dann „Save and Checkpoint“ klickt. | * Exploration der Umweltdaten hinsichtlich der eigenen Forschungsfrage * Ausführen der Datenoperationen und Datentransformationen „Filtern“, „Visualisieren“, „Gruppieren“, „Berechnen von statistischen Kennwerten“ * Kennenlernen von Worked Examples und der Arbeitsweise mit diesen für eigene Programmierprozesse | Jupyter Notebook Umgebung  <https://go.upb.de/ep-umweltanalyse>  Darin: Jupyter Notebooks im Unterordner *Umgebung NASA-Daten*  bzw.  *Umgebung eigene Daten*  sowie Worked Examples im Unterordner *Worked Examples* |
| **3.3** | **Finalisieren des Computational Essays**  **PPDAC**  Nachdem die Gruppen in der vorausgehenden Phase bereits verschiedene Datenoperationen und Datentransformationen ausführen konnten, wird ihnen in dieser Phase das Format eines Computational Essays nähergebracht (siehe Abschnitt 9.1.1). Basierend darauf finalisieren die einzelnen Gruppen ihre Erkenntnis- und Programmierprozesse, wobei am Ende ein Computational Essay entstehen soll.  Diese Phase besteht folglich aus einem Input zu Computational Essays sowie einer Programmier-Phase, in der die einzelnen Gruppen ihre Computational Essays als Jupyter Notebooks finalisieren.  Der Input zum Thema Computational Essays kann dabei anhand der Powerpoint-Folien mit dem Namen „Input\_Computational\_Essays“ erfolgen. In der Datei findet man dabei auch konkrete Notizen zu den einzelnen Folien, die bei der Erläuterung im Unterricht unterstützen können. In den Folien wird einerseits der PPDAC-Zyklus (siehe auch Abschnitt 9.4.2) erklärt, der einen „typischen“ Ablauf von Datenprojekten darstellt und dem die Schüler\*innengruppen in diesem Unterrichtsmodul – wenn auch unbewusst – gefolgt sind. Anschließend wird der Begriff des Computational Essays geklärt, bevor dann erläutert wird, wie ein Computational Essay aufgebaut sein kann und was es beinhalten kann, wenn es nach dem PPDAC-Zyklus strukturiert ist.  Zusätzlich kann auch noch einmal das Worked Example zu den CO2-Daten durchlaufen werden – entweder gemeinsam im Plenum oder individuell von den Schüler\*innen (im Unterordner „Worked Examples => Worked\_Example\_CO2-Messstation => WorkedExample\_CO2.ipynb“). Das Worked Example stellt ebenfalls ein Computational Essay dar, welches nach dem PPDAC-Zyklus aufgebaut ist und kann folglich eine gute Orientierung für die Schüler\*innen hinsichtlich ihrer eigenen Jupyter Notebooks bieten.  Als Unterstützung bei der Erstellung und Finalisierung des Computational Essays dient die Tabelle auf der Checkliste 2 (Computational Essay). In dieser Tabelle ist für die einzelnen Schritte des PPDAC-Zyklus‘ eingetragen, was in einem entsprechenden Computational Essay integriert werden sollte. Die Gruppe kann diese Tabelle nutzen und jeweils in der Spalte „Erledigt“ abhaken, welche Aspekte sie bereits beachtet hat für ihr eigenes Computational Essay.  Didaktischer Kommentar:  Diese Phase bzw. der Input zu den Computational Essays kann auch an einer anderen Stelle im Erkenntnis- und Programmierprozess erfolgen und muss nicht zwangsläufig wie hier dargestellt ganz am Ende erfolgen. Eine frühere Thematisierung hätte den Vorteil, dass die Schüler\*innen bereits von Beginn an ihre Jupyter Notebooks entsprechend des Formats eines Computational Essays aufbereiten könnten. Auf diese Weise wäre sichergestellt, dass das Jupyter Notebook in den vielen Explorationsschritten nicht „durcheinander“ gerät. Allerdings könnten die Schüler\*innen dann den Explorationsprozess nicht mehr so frei durchlaufen, wie wenn sie zunächst ganz frei die Daten explorieren könnten, ohne gleichzeitig Rücksicht auf die äußere Form des Jupyter Notebooks nehmen zu müssen[[3]](#footnote-4). Vermutlich ist deswegen ein Vorgehen, bei dem die Schüler\*innen erst frei vorgehen und anschließend das Jupyter Notebook entsprechend eines Computational Essays strukturieren können, sinnvoll.  Das Ziel am Ende dieser Phase sollte sein, dass die Schüler\*innengruppen ein strukturiertes Computational Essay erstellt haben, welches dann anderen Schüler\*innengruppen oder auch Personen außerhalb des Kurses, die sich für die behandelten Forschungsfragen der Schüler\*innen interessieren, zugänglich gemacht werden kann. Im Computational Essay sollte der Erkenntnis- und Programmierprozess nachvollziehbar und reproduzierbar dargestellt werden, sodass Lesende des Computational Essays damit interagieren oder es erweitern/verändern können. Dazu kann der PPDAC-Zyklus eine gute Struktur für den Aufbau des Computational Essays darstellen. Wie auch im Foliensatz auf Folie 12 gezeigt, können so verschiedene Aspekte von der Problembeschreibung bis hin zur Schlussfolgerung im Computational Essay beleuchtet werden, sodass sich eine kohärente Struktur für die Erläuterung des Erkenntnis- und Programmierprozess ergibt. Natürlich kann auch ein anderes Modell als der PPDAC-Zyklus genutzt werden. Allerdings stellt dieser im Kontext dieses Unterrichtsmoduls einen passenden „roten Faden“ dar, der direkt auf den Ablauf des Datenexplorationsprozesses der Schüler\*innengruppen angewandt werden kann.  Die Tabelle der Checkliste 2 (Computational Essay) bietet für die Schüler\*innengruppen eine Orientierungshilfe für die Erstellung und Finalisierung des Computational Essays. Hier sollen sie zunächst in der grau unterlegten Spalte abhaken, wenn sie einen Aspekt für ihr Computational Essay beachtet haben. Die in der Tabelle aufgelisteten Aspekte können natürlich auch entsprechend des konkreten Programmierprojektes adaptiert oder erweitert werden. | * Kennenlernen von Computational Essays als interaktive Dokumentationsmöglichkeit des eigenen Erkenntnis- und Programmierprozesses * Adaption und Fertigstellung eines Computational Essays als Projektbericht | Powerpointfolien zum Computational Essay und zum PPDAC-Zyklus (Input\_Computational\_Essays)  Jupyter Notebook Umgebung  <https://go.upb.de/ep-umweltanalyse>  Darin: Jupyter Notebooks im Unterordner *Umgebung NASA-Daten*  bzw.  *Umgebung eigene Daten*  sowie Worked Examples im Unterordner *Worked Examples*  Checkliste Computational Essay:  Checkliste\_Computational\_Essay |
| **Teil 4: Vorstellung und Austausch der Computational Essays** | | | |
| **4.1** | **Austausch der Computational Essays in verschiedenen Gruppen**  In dieser Phase tauschen jeweils zwei Schüler\*innengruppen ihre Computational Essays untereinander aus. Wie das funktioniert, wird im Hinweisdokument „Austausch von Jupyter Notebooks“ erläutert.  Jede Gruppe soll nun zunächst erst einmal das erhaltene Computational Essay in Ruhe durchlesen und versuchen, den Erkenntnis- und Programmierprozess der anderen Gruppe zu verstehen und nachzuvollziehen. Im Anschluss daran spricht sich die Gruppe untereinander ab und hält auf der ersten Seite der Checkliste 2 fest, welche Erkenntnisse sie dem Computational Essay entnehmen konnte, was besonders gut gelungen bzw. verständlich ist und was sie nicht (vollständig) verstanden hat bzw. an welchen Stellen noch Informationen fehlten.  Anschließend füllt die Gruppe gemeinsam die Tabelle auf der Checkliste aus (Seite 2 und 3), um für die einzelnen Schritte des PPDAC-Zyklus‘ genaue Rückmeldung geben zu können. Dabei soll sich die Gruppe für jeden der aufgelisteten Aspekte auf eine Bewertung von +, o oder - einigen (in etwa + „gut gemacht“, o „Hier könnte noch ein bisschen etwas verändert werden“ und - „Hier fehlt etwas oder hier muss etwas verändert werden“). Falls ein Aspekt aufgrund des konkreten Projektes nicht auf das Computational Essay zutrifft (beispielsweise wenn die Daten bereits in aufbereiteter Form vorliegen – siehe Aspekt 3a)), soll „Nicht zutreffend“ angekreuzt werden. Zeitgleich gibt es unten Platz, um Notizen zu einzelnen Aspekten hinzuzufügen. Hier sollte die Gruppe auf konkrete Aspekte hinweisen, die von der anderen Gruppe noch bearbeitet werden sollten bzw. die unklar sind und konstruktive Vorschläge zur Verbesserung geben. Insbesondere die Aspekte, die mit - bewertet wurden, sollten hier durch Notizen aufgegriffen werden. | * Computational Essays aus Lesendenperspektive wahrnehmen * Feedback zu einem anderen Computational Essay geben | Checkliste Computational Essay:  Checkliste\_Computational\_Essay  Jupyter Notebook Umgebung  <https://go.upb.de/ep-umweltanalyse>  Computational Essay einer anderen Gruppe  Hinweisdokument „Austausch von Jupyter Notebooks“:  Hinweisdokument\_Austausch\_Jupyter\_Notebooks |
| **4.2** | **Einarbeiten von Feedback für das eigene Computational Essay**  In dieser Phase überarbeiten die Gruppen ihre eigenen Computational Essays aufbauend auf dem Feedback, das sie von der anderen Gruppe anhand der Checkliste 2 (Computational Essay) erhalten haben. Dabei gehen die Gruppen in vier Schritten vor. Zunächst einmal identifizieren sie Aspekte, die sie überarbeiten möchten und (mit zumutbarem Aufwand) überarbeiten können. Anschließend überlegen sie sich, was zu tun bzw. zu adaptieren/ergänzen ist, um die entsprechenden Aspekte zu überarbeiten. Beide Schritte halten die Gruppen in der Tabelle auf Seite 4 der Checkliste fest. Zuletzt führen sie dann die Anpassungen/Erweiterungen aus der Tabelle durch und prüfen abschließend, ob damit die notierten Aspekte verbessert werden konnten.  Im ersten Schritt sichten die Gruppen die von der Partner\*innengruppe ausgefüllte Checkliste (S.1-3) und besprechen, welche Aspekte sie basierend auf diesem Feedback anpassen möchten. Zunächst gehen sie dabei das allgemeine Feedback auf der ersten Seite der Checkliste durch und arbeiten möglichst konkrete Punkte heraus, die sie in der ersten Spalte der Tabelle auf Seite 4 festhalten. Anschließend gehen sie die am PPDAC-Zyklus orientierte Tabelle auf den Seiten 2 und 3 durch und notieren sich hierzu die als negativ markierten Aspekte sowie zugehörige Notizen aufbauend auf denen der anderen Gruppe auf Seite 3 unten. Dabei können sie zunächst in der Tabelle auf Seite 2/3 die Aspekte markieren, die sie überarbeiten wollen, indem sie diese in der letzten Spalte „Überarbeitung“ markieren, bevor sie für diese markierten Aspekte dann in der Tabelle auf Seite 4 festhalten und dabei auch die Notizen der anderen Gruppe berücksichtigen.  Im zweiten Schritt überlegen sie sich für jeden der notierten Aspekte in der ersten Spalte der Tabelle auf Seite 4, wie sie diesen Aspekt verbessern möchten bzw. wie sie das Computational Essay anpassen/erweitern möchten. Diese möglichst konkreten Anpassungs-/Erweiterungsvorschläge halten sie dabei in der zweiten Spalte der Tabelle auf Seite 4 fest.  Im dritten Schritt überarbeiten sie dann das Computational Essay (ggf. arbeitsteilig), indem sie die festgehaltenen Vorschläge umsetzen.  Abschließend bzw. im vierten Schritt prüfen sie dann noch einmal kritisch, ob die notierten Aspekte mithilfe der Adaptionen/Erweiterungen des Computational Essays hinreichend verbessert werden konnten. Dabei können sie jeweils in der Spalte „Erledigt“ markieren, auf welche Aspekte dies zutrifft. Für die dann noch verbliebenen Aspekte kann die Gruppe dann noch einmal eine zweite Verbesserungsschleife durchlaufen.  Didaktischer Kommentar:  Die Gruppen sollen in dieser Phase möglichst eigenständig und basierend auf dem Feedback ihrer Partner\*innengruppen arbeiten. Dabei können sie natürlich besonders bei Verständisfragen oder Unklarheiten mit der anderen Gruppe austauschen und ggf. noch einmal genauer nachfragen, was an dem Computational Essay noch nicht optimal war bzw. wie sie einen bestimmten Aspekt noch verbessern können.  Wichtig zu beachten ist dabei, dass die Gruppe natürlich nur solche Aspekte angehen sollte, die auch (mit zumutbarem Aufwand) verbessert werden können. Insbesondere sollten keine neuen Forschungsfragen oder gänzlich neue Auswertungsmethoden angegangen werden. Vielmehr sollte es in dieser Phase darum gehen, die erlangten Erkenntnisse sowie den Prozess, in dem diese gewonnen werden konnten, übersichtlich und nachvollziehbar darzustellen sowie einen „roten Faden“ in das Computational Essay zu bringen. | * Feedback strukturiert aufbereiten und darauf aufbauend Adaptions-/ Erweiterungsvorschläge entwickeln * Vorhandenes Computational Essay feedbackgeleitet anpassen und erweitern | Checkliste Computational Essay:  Checkliste\_Computational\_Essay  Jupyter Notebook Umgebung  <https://go.upb.de/ep-umweltanalyse>  Eigene Computational Essays |
| **4.3** | **Vorstellung der Computational Essays**  In dieser Phase erhalten die Schüler\*innengruppen die Möglichkeit, ihre Computational Essays der gesamten Lerngruppe (und ggf. auch anderen interessierten Personen) vorzustellen und diese für die Öffentlichkeit verfügbar zu machen (beispielsweise über die Schulhomepage).  Für die Vorstellung der Computational werden die Computational Essays in einem Rundgang gestellt. Dazu finden sich aus jeder Gruppe jeweils ein\*e Schüler\*in zu einer neuen Gruppe zusammen. Um die Präsentation in dieser neuen Konstellation möglichst aktiv für alle Schüler\*innen zu gestalten kann man die Phase an für jede Präsentation in drei Teile untergliedern: Kurzpräsentation, individuelle Exploration, Fragen und Antworten  Für die Kurzpräsentation sollen sich die Gruppen, die gemeinsam die Computational Essays erstellt haben, vor diesem Rundgang zusammensetzen und gemeinsam überlegen, wie sie die Kurzpräsentation gestalten wollen. Dazu halten sie auf der ersten Seite des Arbeitsblatts 4 (Präsentation) Notizen zu den o.g. Aspekten fest, mithilfe derer sie dann später ihre Kurzpräsentation halten können. Für die anschließende Phase der individuellen Exploration stellt die Gruppe zudem das eigene Computational Essay samt aller benötigten Ressourcen (vor allem Daten, ggf. auch Bilder) zusammen und stellt diese Dateien zur Verfügung, sodass andere Schüler\*innen darauf zugreifen können. Wie das funktioniert, wird im Hinweisdokument „Austausch von Jupyter Notebooks“ erläutert.  *HINWEIS: In Zukunft soll es möglich sein, dass dieser Schritt des zur Verfügung Stellens der Jupyter Notebooks und Ressourcen durch die Lehrkraft erfolgen kann. Sobald dies möglich ist, wird dies hier an dieser Stelle erläutert.*  In der ersten Phase stellt der\*die Gruppenverantwortliche in einer Kurzpräsentation (max. 3min) das Computational Essay vor, indem sie\*er beschreibt, was die Gruppe untersucht hat, wie sie vorgegangen ist, welche Auswertungen und Visualisierungen erstellt wurden und welche Erkenntnisse gewonnen werden konnten.  Im Rundgang folgt dann eine individuelle Explorationsphase (ca. 10 min) auf diese Kurzpräsentation der jeweiligen Gruppe. Dazu durchlaufen alle Schüler\*innen das Computational Essay der jeweiligen gerade präsentierenden Gruppe und versuchen, den Erkenntnis- und Programmierprozess nachzuvollziehen. Dabei machen sie sich auf dem AB 4 Notizen, wobei sie jeweils festhalten, was die Gruppe untersucht hat, welche Erkenntnisse sie gewinnen konnte und wie sie dabei vorgegangen ist. Die\*der Schüler\*in der gerade präsentierenden Gruppe kann dabei bei kurzfristig relevanten Fragen unterstützen.  Im Anschluss gibt es dann die Gelegenheit, Fragen an den\*die jeweilige Gruppenverantwortliche\*n zu stellen. Darauf aufbauend können die Schüler\*innen ihre Notizen noch einmal ergänzen. Zusätzlich sollen sie auf der freien Linie hinter „Projektpräsentation“ jeweils einen Titel für das aktuell betrachtete Projekt festlegen. Anschließend geht es dann mit der Kurzpräsentation der nächsten Gruppe weiter.  Diese dreischrittige Phase wird hier für alle Gruppen durchlaufen, sodass die Schüler\*innen am Ende Notizen zu jeder Gruppe auf ihrem AB 4 gemacht haben und auch zu ihrer eigenen Gruppe Notizen ergänzt haben auf der ersten Seite.  *Optional, falls eine Veröffentlichung der Computational Essays gewünscht ist:*  Im Anschluss an diese Vorstellung können die Computational Essays für die Präsentation in der Öffentlichkeit als HTML-Datei exportiert werden. Wie dies funktioniert, ist in Abschnitt 9.1.6 nachzulesen. Anhand der HTML-Datei kann das Computational Essay dann beispielsweise als Seite auf der Schulhomepage veröffentlicht werden oder aber auch an anderer geeigneter Stelle. Wichtig ist dabei zu erwähnen, dass der Programmcode in der HTML-Datei bzw. nach Veröffentlichung dieser auf der Website nicht verändert werden kann. Dennoch können die Auswertungen und Visualisierungen in dieser Datei interaktiv betrachtet werden, d.h. insbesondere kann man in die Visualisierungen wie gewohnt „hineinzoomen“ oder zusätzliche Informationen durch „Darüberschweben“ der Maus anzeigen lassen. Folglich kann man auf diese Weise die Computational Notebooks und die darunterliegenden Erkenntnisprozesse der breiten Öffentlichkeit zugänglich machen.  Didaktischer Kommentar:  An dem Rundgang können auch weitere interessierte Personen aus der Schule oder der Öffentlichkeit teilnehmen. Insbesondere wenn Fragestellungen mit Bedeutsamkeit für die lokale Umgebung behandelt wurden, kann es interessant sein, auch entsprechende Personen aus der Öffentlichkeit (wie z.B. Lokalpolitiker\*innen, die Presse o.Ä.) einzuladen, um die Ergebnisse geeigneten Entscheidungstragenden vorzustellen, die dann ggf. entsprechende Maßnahmen ergreifen können.  Der Rundgang mit den individuellen Explorationsphasen eignet sich insofern besonders gut, als dass sich dadurch alle Schüler\*innen ein eigenes Bild von den anderen Projekten machen und sich dabei aktiv mit deren Erkenntnis- und Programmierprozesse auseinandersetzen können. Zudem präsentieren sie auch ihre eigenen Erkenntnisprozesse und reflektieren so auch noch einmal ihren eigenen Prozess.  Alternativ kann statt des Rundgangs auch eine Präsentation im Plenum erfolgen, beispielsweise um Zeit zu sparen. Bei einer Präsentation können die Schüler\*innen jeder Gruppe das Computational Essay schrittweise durchlaufen und die einzelnen Schritte genauer erläutern.  Die Seiten 2 und 3 des Arbeitsblattes 4, welche die Arbeitsanweisungen samt Platzhalter für die Notizen zu den anderen Gruppen enthalten können von der Lehrkraft abhängig von der Gruppenanzahl passend häufig kopiert werden.  Durch die Notizen auf dem Arbeitsblatt 4 halten die Schüler\*innen jeweils für alle Gruppen Erkenntnisse sowie die Prozesse dahinter fest. Somit dienen diese Notizen zusätzlich zu den Computational Essays an sich der Sicherung der Erkenntnis- und Programmierprozesse der eigenen und der anderen Gruppen.  Die Veröffentlichung auf der Schulhomepage muss nicht zwingend erfolgen und kann auch weggelassen werden bzw. auf eine andere Webseite oder ein anderes Medium verlagert werden, wenn dies passender ist im jeweiligen Setting. | * Zusammentragen und Präsentieren eigener Gruppenergebnisse * Aktives Durchlaufen der Computational Essays anderer Gruppen und Nachvollziehen der jeweiligen Erkenntnis- und Programmierprozesse * Austausch über die erlangten Erkenntnisse und Programmierprozesse, in deren Kontext diese Erkenntnisse gewonnen werden konnten * Veröffentlichung der gruppeneigenen Computational Essays | AB4\_Praesentation  Jupyter Notebook Umgebung  <https://go.upb.de/ep-umweltanalyse>  Computational Essays aller Gruppen, die zur Verfügung gestellt werden  Hinweisdokument „Austausch von Jupyter Notebooks“:  Hinweisdokument\_Austausch\_Jupyter\_Notebooks |
| **Teil 5: Reflexion** | | | |
| **5.1** | **Reflexion des Erkenntnis- und Programmierprozesses**  In dieser Phase machen sich die Schüler\*innen ihre individuelle Wahrnehmung über den Programmierprozesses in diesem Unterrichtsmodul sowie über die Rolle des Programmierens bewusst. Dazu reflektieren sie ihren eigenen Erkenntnis- und Programmierprozess und Perspektiven auf die Rolle des Programmierens mithilfe des Arbeitsblatts 5 – Reflexion des Programmierprozesses. Dieses ist in zwei Phasen unterteilt. In der ersten Phase bearbeiten die Schüler\*innen individuell die Aufgaben 1-5 des Arbeitsblattes, die in zwei Teile (eigenes Projekt bzw. Erkenntnis- und Programmierprozess und Perspektiven auf die Rolle des Programmierens) unterteilt sind. In den ersten 3 Reflexionsaufgaben geht es zunächst darum, den Erkenntnis- und Programmierprozess der eigenen Gruppe noch einmal zu beleuchten und festzuhalten, wie dieser abgelaufen ist bzw. welche Rolle der Computer bzw. das Programmieren gespielt hat. In den daran anschließenden Aufgaben 4 und 5 nehmen die Schüler\*innen dann eine Meta-Perspektive ein und bewerten die Wahrnehmung des Programmierens als Mittel zum Erkenntnisgewinn auf persönlicher oder gesellschaftlicher Ebene und entwickeln von dort aus weitere Ideen für Programmierprojekte, in denen es ebenfalls um diesen Gewinn von Erkenntnissen über persönlich oder gesellschaftlich relevante Themen geht. Wichtig ist es, dass den Schüler\*innen verdeutlicht wird, dass es bei diesen Fragestellungen kein „richtig“ oder „falsch“ gibt, sondern dass es darum geht, dass die Schüler\*innen sich individuell ihre Wahrnehmung auf das Programmieren als Erkenntnismittel bzw. dessen Rolle in Erkenntnisprozessen bewusst machen, um es ggf. später als ein solches Werkzeug entsprechend dieser Wahrnehmung einsetzen zu können.  Nachdem die Schüler\*innen diese ersten 5 Aufgaben bearbeitet haben, gibt es eine finale Austausch- und Diskussionsphase, in der sich die Schüler\*innen hinsichtlich der Wahrnehmung des Programmierens austauschen können und ggf. weitere Perspektiven/Ideen aufnehmen und mitnehmen können. Dabei kann die Diskussionsphase wahlweise entlang der 5 Aufgaben erfolgen oder alternativ auch offener gestaltet sein. Als ein möglicher Einstiegspunkt könnte dabei die Aussage aus Aufgabe 4 dienen, welche die Schüler\*innen diskutieren können. Dabei könnten sie dann auch Erfahrungen aus ihren eigenen Erkenntnis- und Programmierprozessen einbringen, die sie in den Aufgaben zuvor festgehalten haben. Ebenfalls könnten sie danach auch weitere Ideen aus Aufgabe 5 austauschen.  Zur Sicherung bearbeiten die Schüler\*innen dann noch die Aufgabe 6 auf dem Arbeitsblatt, in der sie individuell relevante Charakteristika des diskutierten Programmierbegriffes für sich festhalten, sodass diese ggf. in späteren (eigenen) erkenntniszentrierten Programmierprojekten als gedanklicher Anker dienen können, sodass die Schüler\*innen erneut in einen ähnlichen Prozess eintauchen können.  Didaktischer Kommentar:  In dieser Phase sollen sich die Schüler\*innen ihre Wahrnehmung über die in diesem Unterrichtsmodul vermittelte erkenntniszentrierte Perspektive auf das Programmieren bewusst machen. Dies hat im wesentlichen 2 Ziele: Zum einen soll den Schüler\*innen dadurch bewusst werden, dass Programmieren nicht „nur“ für die Entwicklung professioneller Software geeignet ist, sondern ein hilfreiches Werkzeug sein kann, um persönlich oder gesellschaftlich relevante Fragestellungen oder Sachverhalte zu erkunden. Zum anderen sollen die Schüler\*innen für sich selbst überlegen, wo sie diese Art des Programmierens – sei es im Alltag oder in anderen Kontexten – nutzen können. Hier sollte ihnen in der Diskussionsphase (ggf. durch Erläuterung der Lehrkraft) auch bewusst gemacht werden, dass es für eine Vielzahl an Projektideen ähnlich wie in dem hier behandelten Kontext von Umweltdatenauswertungen frei zugängliche Bibliotheken und Dokumentationen gibt sowie bereits fertige und kommentierte Teillösungen (ähnlich oder genau wie Worked Examples).  Die Schüler\*innen sollen so in dieser Phase ermuntert werden, Programmieren – bildlich gesprochen – als ein Werkzeug für ihren „Werkzeugkasten“ zu übernehmen, um eigenen Fragestellungen oder Interessen – ggf. auch datenbasiert nachzugehen. Im Rahmen dieses Projektes haben sie eine erste „Grundbefähigung“ dafür bekommen, dies basierend auf Worked Examples und mithilfe bereits fertiger Bibliotheken zu tun.  *Eine kleine Bitte:*  In unserer Forschung im Rahmen des ProDaBi-Teams beschäftigen wir uns unter anderem mit der Entwicklung des Epistemischen Programmierens als erkenntniszentriertem Programmieransatz. Ein wesentlicher Teil bezieht sich dabei darauf, die Perspektive der Schüler\*innen auf das Programmieren nach der Durchführung dieses Unterrichtsmoduls zu erheben und auszuwerten. An dieser Stelle gibt es daher die Möglichkeit, dass die Schüler\*innen Teil dieser Forschung werden, indem wir ihre Bearbeitungen des Arbeitsblattes 5 einsammeln und auswerten. Falls Sie und Ihre Schüler\*innen interessiert daran sind, uns in unserer Forschung zu unterstützen, würden wir uns sehr freuen, wenn Sie uns eine E-Mail an Sven Hüsing ([sven.huesing@upb.de](mailto:sven.huesing@upb.de)) senden, der verantwortlich für dieses Modul ist. Sie erhalten dann eine Einverständniserklärung, die die Schüler\*innen und ihre Erziehungsberechtigten ausfüllen müssen, um an der Studie im Rahmen der beschriebenen Forschung teilzunehmen. Wir wären Ihnen sehr dankbar, wenn Sie und Ihre Schüler\*innen uns in dieser Hinsicht unterstützen würden. | * Bewusstmachen der individuell wahrgenommenen Rolle des Programmierens * Reflexion der eigenen Programmierprozesse sowie der Rollen von Mensch und Computer darin * Herausarbeiten typischer Charakteristiken erkenntniszentrierter Programmierprozesse | AB5\_Reflexion |

# Beschreibungen der Materialien und weiterführende Hintergrundinformationen

In diesem Abschnitt sollen einzelne Materialien – und besonders die Jupyter Notebook-Umgebung, in der die Schüler\*innen während des Unterrichtsmoduls arbeiten, genauer erläutert werden.

## Arbeit mit Jupyter Notebooks und Jupyter Notebook Umgebung

In diesem Unterrichtsmodul werten die Schüler\*innen zuvor erhobene oder gesammelte Daten in Jupyter Notebooks aus. Die Jupyter Notebook Umgebung bezeichnet eine webbasierte, interaktive Plattform, auf der Nutzende programmierbasierte Narrative erstellen können, die Live-Code, erläuternden Text, interaktive Elemente und andere Medien beinhalten können (siehe Perez & Granger, 2015, S. 2). Die Dokumente, die Nutzende erstellen können, sogenannte Jupyter Notebooks, bestehen aus Code- und Markdown-Zellen, die untereinander angeordnet sind. Die Nutzenden können Code in die Code-Zellen schreiben, dessen Output beim Ausführen des Codes direkt unter der Codezelle angezeigt werden kann. Durch die Nutzung mehrerer Code-Zellen kann man so den gesamten Programmcode aufteilen und sich Zwischenergebnisse anzeigen lassen. Zusätzlich kann man in den Markdown-Zellen Kommentare hinzufügen, beispielsweise um den Programmcode zu dokumentieren.

Ein Beispiel für ein Jupyter Notebook findet man hier: <https://jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb> (Aufruf am 20.11.2024; Hinweis: Nach Aufrufen des Links muss man zunächst oben in die leere Zelle auf der Seite klicken, um das Notebook zu aktivieren.)

Ein Vorteil, den die Verwendung von Jupyter Notebooks für die Arbeit in der Schule bietet, ist, dass Lehrkräfte bereits vorgefertigte Jupyter Notebooks zur Verfügung stellen können, in denen beispielsweise schon Programmcode (aus verschiedenen Bibliotheken) vorhanden ist. Die Schüler\*innen können dann auf diesem Programmcode aufbauen und ihn hinsichtlich eigener Interessen erweitern. Auch kann ein Jupyter Notebook wie ein digitales Arbeitsblatt genutzt werden, indem Lehrkräfte Aufgabenstellungen in das Jupyter Notebook (als Markdown-Zellen) schreiben. Die Schüler\*innen können die Aufgabenstellungen dann in (ggf. vorbereiteten oder schon teilweise ausgefüllten) Codezellen bearbeiten.

Weitere Informationen über Jupyter Notebooks findet man in der Dokumentation unter <https://docs.jupyter.org/en/latest/> (Aufruf am 20.11.2024)

### Computational Essays als Programmierprodukte

Ein Computational Essay bezeichnet ein interaktives Dokument, welches erläuternden Text, kleine Programme oder Programmschnipsel und Outputs beinhaltet, um Ergebnisse einer Untersuchung zu präsentieren oder eigene Ideen auszudrücken (siehe Atkinson et al., 2000; Odden & Malthe-Sørenssen, 2021; Wolfram, 2017). Ein Computational Essay kann folglich dazu genutzt werden, den Programmierprozess verständlich darzustellen, sodass Lesende nachvollziehen können, wie man zu den Ergebnissen und Schlussfolgerungen gelangt ist (siehe auch Hüsing & Podworny, 2022). Gleichzeitig können Leser\*innen eines Computational Essays mit den Inhalten oder dem Code interagieren und auf diese Weise den Programmierprozess eigenständig weiter/tiefer explorieren.

Im Kontext der Datenexploration stellen Computational Essays ein geeignetes Mittel dar, um den verwobenen Datenexplorations- und Programmierprozess festzuhalten und zu dokumentieren. In einem Computational Essay kann die Auswertung Schritt für Schritt erklärt werden und gleichzeitig können die Programmierergebnisse wie Visualisierungen oder die berechneten Kennwerte beschrieben und interpretiert werden. Die Verwendung von Jupyter Notebooks und die Unterteilung in viele Programmier- und Markdownzellen ist hierzu sehr gut geeignet.

Nachfolgend sieht man einen Ausschnitt eines beispielhaften Computational Essays, in dem ein Vergleich der Temperatur in London und Paderborn gezeigt wird. In den grünen Zellen befindet sich eine Erklärung zu dem genutzten Programmcode sowie eine (kurze) Interpretation der beiden dargestellten Graphiken.

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

Automatisch generierte Beschreibung

### Worked Examples als Unterstützung beim Programmieren nutzen

Ein Worked Example bezeichnet ein vollständig ausgearbeitetes und dokumentiertes Beispiel zu einem ähnlichen Projekt, wie dem, was gerade betrachtet und angegangen wird (siehe Atkinson et al., 2000; Muldner et al., 2023). Es kann als Quelle für Inspiration und als Orientierung genutzt werden – insbesondere, wenn man noch wenig Erfahrung im Kontext des Projektes hat. Im Kontext des Programmierens können Worked Examples überdies als „Steinbruch“ für Programmcode dienen, indem Lernende den Programmcode aus einem Worked Example zunächst kopieren, dann an die eigenen Bedürfnisse bzw. an das eigene Projekt anpassen und anschließend erweitern (siehe auch „Use-Modify-Make“ in Lee et al., 2011 sowie Hüsing et al., 2024). Hier eignen sich vor allem Computational Essays als Format für ein Worked Example, um den Programmierprozess für Lernende transparent zu machen und aufzuzeigen, wie die Programmierschnipsel genutzt werden können.

Im Rahmen dieses Unterrichtsmoduls gibt es zwei Worked Examples, welche die Schüler\*innen dabei unterstützen können, eigene Explorationen von Umweltdaten durchzuführen. Im Worked Example zur CO2-Analyse wird exemplarisch gezeigt, wie sich CO2-Daten im Zusammenhang mit anderen Größen analysieren lassen. Darüber hinaus wird im Worked Example zur Wetteranalyse gezeigt, wie sich Wetterdaten verschiedener Orte explorieren lassen. Lernende können diese Worked Examples als Orientierung oder „Code-Steinbruch“ für ihre eigenen Umweltdatenprojekte nutzen.

### Zugriff auf die Jupyter Notebook-Umgebung für diese Unterrichtsreihe (Erstellen eines Accounts und Login)

Für die Phase der Datenanalyse steht für dieses Unterrichtsmodul eine vorbereitete Jupyter Notebook Umgebung zur Verfügung. Diese kann über den folgenden Link aufgerufen werden: <https://go.upb.de/ep-umweltexploration>

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

Automatisch generierte BeschreibungBeim ersten Öffnen dieses Links müssen sich die Schüler\*innen einen eigenen Account erstellen. Dazu geben sie in die Anmeldemaske einen Benutzernamen und ein Passwort ein. Beides können sie sich frei überlegen. Wichtig ist nur, dass sie sich beides merken, um den Arbeitsfortschritt im eigenen Account immer wieder abrufen zu können. Dazu können sie sich später mit ihren bei der ersten Anmeldung vergebenen Daten wieder unter dem gleichen Link anmelden.

### Aufbau und Inhalte der Jupyter Notebook Umgebung

In der Jupyter Notebook Umgebung finden sich verschiedene Jupyter Notebooks, die von den Schüler\*innen im Unterrichtsmodul verwendet werden können. Nach dem Einloggen in die Umgebung gelangt man zu einer Ordner-Übersicht. Jeder Ordner (bis auf den res-Ordner) stellt dabei eine separate Umgebung dar, auf die im Folgenden kurz eingegangen wird.

* Cheatsheets
  + Cheatsheet Datenexploration in Jupyter Notebooks mit Python.ipynb
    - *Cheatsheet, welches wesentliche Python-Befehle und Codeschnipsel zur Datenexploration in Jupyter Notebooks zeigt und kurz erläutert; Schüler\*innen können dieses Cheatsheet als Hilfestellung während ihrer eigenen Datenexplorationen nutzen*
* Tutorial\_JupyterNotebooks\_Python

*Dieser Ordner enthält mehrere Tutorial-Jupyter-Notebooks, die in die Verwendung von Jupyter Notebooks mit der Programmiersprache Python einführen. Die Jupyter Notebooks können dabei individuell durchlaufen und bearbeitet werden.*

* + 0\_Start.ipynb
  + 1\_Erste\_Schritte.ipynb
  + 2\_Python\_Basics. ipynb
  + 3\_Fallunterscheidungen.ipynb
  + 4\_Wiederholungen.ipynb
  + 5\_Funktionen.ipynb
  + 6\_DataFrames.ipynb
* Umgebung eigene Daten

*Dieser Ordner enthält die Umgebung für die Exploration eigener Daten, die entweder selbst erhoben (siehe Phase 2a der Unterrichtsreihe) oder aus verschiedenen Quellen gesammelt und zusammengetragen wurden (siehe Phase 2b der Unterrichtsreihe).*

* + Computational\_Notebook.ipynb
    - *Jupyter Notebook, in dem die Schüler\*innen ihre eigene Datenanalyse durchführen können; enthält bereits den Import der wesentlichen Bibliotheken sowie die Aufgabenstellung/Einführung*
  + Eigene\_Basisinformationen.ipynb
    - *Jupyter Notebook, in dem Basisinformationen über die Daten (u.A. Messmethoden/Herkunft etc.) festgehalten werden können.*
  + Messprotokoll.ipynb
    - *Jupyter Notebook, in dem Informationen über die Messung und das Messprotokoll festgehalten werden kann, falls die Schüler\*innen eigene Daten erhoben haben*
  + Tutorial\_Daten\_einlesen.ipynb
    - *Kurzes Tutorial, in dem erklärt wird, wie eigene Daten in die Jupyter Notebook Umgebung geladen werden können*
* Umgebung NASA-Daten

*Dieser Ordner enthält die Umgebung für die Exploration der Daten aus dem POWER-Projekt des NASA Langley Research Center (LaRC), das durch das NASA Earth Science/Applied Science Program finanziert wird.*

* + Computational\_Notebook\_NASA-Data.ipynb
    - *Notebook, in dem Schüler\*innen verschiedene Wetterdaten zu beliebigen Orten und für beliebige Zeiträume direkt in das Jupyter Notebook laden und dort analysieren können; Folgende Wetterdaten können in das Jupyter Notebook geladen werden:* 
      * *Luftfeuchtigkeit in %*
      * *Niederschlag in mm/Stunde*
      * *Temperatur in °C*
      * *Windgeschwindigkeit in m/s*
  + Computational\_Notebook\_NASA-Data\_Beispielvisualisierungen.ipynb
    - *Jupyter Notebook, in dem anhand von Beispielen gezeigt wird, wie bestimmte Visualisierungen mit den „NASA-Daten“ erstellt werden können*
* Worked Examples

*Dieser Ordner enthält 2 verschiedene ausgearbeitete Beispielprojekte zur Exploration von Umweltdaten. Das Projekt wird dabei jeweils als ein Worked Example dargestellt, welches die Lernenden nutzen können, um sich daran zu orientieren oder auch, um einzelne Codeschnipsel zu nutzen/adaptieren/erweitern.*

* + Worked\_Example\_CO2-Messstation
    - Basisinformationen.ipynb
      * *Jupyter Notebook, in dem wesentliche Informationen zur Datenmessung festgehalten sind*
    - WorkedExample\_CO2.ipynb
      * *Jupyter Notebook, in dem exemplarisch eine Exploration von CO2-Daten im Klassenraum gezeigt wird*
  + Worked\_Example\_Wetterdaten
    - WorkedExample\_Wetterdaten.ipynb
      * *Jupyter Notebook, in dem exemplarisch eine Exploration von Wetterdaten (aus London und Paderborn) gezeigt wird, wobei eigene Daten als csv-Datei eingelesen und aufbereitet werden*
    - WorkedExample\_Wetterdaten\_NASA.ipynb
      * *Ähnliches Jupyter Notebook zur Exploration von Wetterdaten aus London und Paderborn, mit dem Unterschied, dass hier Daten aus dem POWER-Projekt des NASA Langley Research Center (LaRC) (s.o.) genutzt werden. Hier wird auch gezeigt, wie die Daten des POWER-Projekts heruntergeladen und eingelesen werden können.*

*Hinweis: Die Ordner mit den Namen „res“ oder „Daten“ enthalten Dateien (RESsourcen) wie Bilder und Daten, die in den Jupyter Notebooks verwendet werden, auf die aber nicht direkt über die Ordnerstruktur zugegriffen werden muss.*

### Herunterladen der eigenen Jupyter Notebooks und Hochladen von fremden Jupyter Notebooks (beispielsweise von anderen Gruppen)

Hier im Unterrichtsmodul sollen die Gruppen in Phase 4 ihre erstellten Jupyter Notebooks austauschen und sich gegenseitig Feedback dazu geben. Auch in anderen Situationen kann es sinnvoll sein, die Jupyter Notebooks untereinander auszutauschen, beispielsweise, wenn die Jupyter Notebooks präsentiert werden sollen und sich interessierte die erstellten Jupyter Notebooks angucken. In diesen Fällen müssen die Jupyter Notebooks aus der Jupyter Notebook Umgebung heruntergeladen werden. Später können sie dann im Account einer anderen Person wieder hochgeladen werden.

Herunterladen eines Jupyter Notebooks

Um ein Jupyter Notebook aus der Jupyter Notebook Umgebung herunterzuladen, gibt es mehrere Möglichkeiten. Entweder man lädt das Jupyter Notebook herunter, wenn es geöffnet ist oder man lädt es aus der Ordneransicht herunter.

Wenn man das aktuell geöffnete Jupyter Notebook herunterladen möchte, klickt man oben in der Liste auf „File“ und dann auf „Download“. Das Jupyter Notebook wird dann als ipynb-Datei heruntergeladen.

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

Automatisch generierte BeschreibungAlternativ kann man das Jupyter Notebook auch aus der Ordneransicht herunterladen. Dazu klickt man den Haken neben dem herunterzuladenen Jupyter Notebook an und klickt dann oben in der Leiste auf „Download“. Ebenfalls kann man hier auch mehrere Dateien auf einmal herunterladen, indem man den Haken neben mehreren Dateien aktiviert und anschließend auf „Download“ klickt.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

***Wichtiger Hinweis:***

*Damit das Jupyter Notebook auch in einem anderen Account geöffnet werden kann, ist es wichtig, alle Daten und Ressourcen, wie Bilder, herunterzuladen und später wieder in derselben Ordnerstruktur hochzuladen. Liegt also beispielsweise eine Datei „data.csv“ in einem Unterordner „Daten“, so muss in dem neuen Account ebenfalls ein solcher Unterordner erstellt werden, in den dann die Datei „data.csv“ hochgeladen wird.*

Hochladen eines Jupyter Notebooks

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

Automatisch generierte Beschreibung*Um ein heruntergeladenes Jupyter Notebook in einem anderen Account wieder hochzuladen, klickt man in der Ordneransicht auf den Button „Upload“. Anschließend wählt man alle Dateien aus, die man hochladen möchte. Die Dateien sollten dann direkt im Ordner erscheinen. Per Drag&Drop oder durch einen Rechtsklick auf einzelne oder mehrere ausgewählte Dateien kann man sie auch in andere Ordner kopieren bzw. verschieben.

***Wichtiger Hinweis:***

*Damit das hochgeladene Jupyter Notebook korrekt funktioniert, müssen auch alle Ressourcen, auf die das Jupyter Notebook Zugriff haben muss (wie z.B. Daten oder Bilder) in derselben Ordnerstruktur wie zuvor hochgeladen werden. Liegt also beispielsweise eine Datei „data.csv“ in einem Unterordner „Daten“, so muss in dem neuen Account ebenfalls ein solcher Unterordner erstellt werden, in den dann die Datei „data.csv“ hochgeladen wird.*

### Export eines Jupyter Notebooks als HTML-Datei

Um ein Jupyter Notebook für eine breitere Öffentlichkeit zugänglich zu machen, kann man es als HTML-Datei exportieren. Dies bietet den Vorteil, dass der Programmcode zwar nicht mehr verändert werden kann, die Programmierergebnisse aber noch immer interaktiv sind. Insbesondere kann man beispielsweise in entwickelte Visualisierungen herein und herauszoomen. Die HTML-Datei kann man beispielsweise nutzen, um das Computational Essay der Lernenden auf der eigenen Homepage zu veröffentlichen und so über die ggf. auch lokalgesellschaftlich relevanten Erkenntnisse der Lernenden nachvollziehbar und transparent zu informieren.

Um ein Jupyter Notebook als HTML-Datei zu exportieren, öffnet man das Jupyter Notebook und klickt dann auf „File => Save and Export Notebook As => HTML“. Die dann heruntergeladene Datei kann man in beliebigen Browsern öffnen. Dort wird dann das Jupyter Notebook entsprechend angezeigt und man kann mit den Programmierergebnissen interagieren. Zugleich kann man den Code aus der HTML-Datei nutzen, um das erstellte Computational Essay beispielsweise auf der Schulhomepage anzuzeigen.

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

KI-generierte Inhalte können fehlerhaft sein.

## Arbeit mit der Sensebox und der OpenSenseMap

### Die Sensebox

Die Sensebox ist ein programmierbares Messinstrument, mit dem eigene Daten gesammelt und gespeichert werden können. Sie besteht aus einem Mikrocomputer, an den verschiedene Messsensoren angeschlossen werden können, um Daten zu erhoben. Die Daten können dabei über eine SD-Karte oder über die OpenSenseMap (mithilfe eines WiFi- oder LoRaWan-Moduls) übertragen werden.

Die Programmierung der Sensebox kann wahlweise mithilfe der blockbasierten Sprache Blockly   
oder mit der Arduino IDE (textbasierte Programmierung) erfolgen.

* Informationen zu Blockly: <https://docs.sensebox.de/docs/boards/mcu/mcu-erster-sketch>
* Informationen zur Arduino IDE: <https://docs.sensebox.de/docs/category/arduino>

Hier gibt es weitere Informationen zur Sensebox: <https://docs.sensebox.de/>

### Die OpenSenseMap

Die OpenSenseMap ist eine Plattform, auf der man die eigene Sensebox registrieren kann, um Daten an die Plattform zu schicken (via WiFi- oder LoRaWan-Modul auf der Sensbox), zu sammeln und zu veröffentlichen. Gleichzeitig hat man über die Plattform Zugriff auf die Daten aller weltweit registrierten Senseboxen. Auf diese Weise lassen sich Datensätze als csv-Dateien von verschiedenen Standorten auf der Welt herunterladen und für die eigene Auswertung nutzen.

Weitere Informationen zur OpenSenseMap: <https://sensebox.de/de/opensensemap>

## Arbeit mit den Daten aus dem POWER-Projekt des NASA Langley Research Center (LaRC)

In der „Umgebung NASA-Daten“ können Schüler\*innen Wetterdaten von überall auf der Welt direkt in ihre Jupyter Notebooks laden. Die Daten stammen aus dem POWER-Projekt des NASA Langley Research Center (LaRC), das durch das NASA Earth Science/Applied Science Program finanziert wird (<https://power.larc.nasa.gov/data-access-viewer/>). Auf der Website lassen sich neben den Daten über Temperatur, Luftfeuchtigkeit, Niederschlag und Windgeschwindigkeit noch viele weitere Daten in verschiedenen Dateiformaten herunterladen. Diese können beispielsweise dann auch in der „Umgebung eigene Daten“ hochgeladen werden.

Hier findet man weitere Informationen über das POWER-Projekt: <https://power.larc.nasa.gov/docs/>

Hinweis: Das POWER-Projekt bittet freundlich um einen Verweis, einen Weblink und/oder einen Abdruck von veröffentlichten Papieren oder Berichten oder eine kurze Beschreibung anderer Verwendungszwecke (z. B. Poster, mündliche Präsentationen usw.) von Datenprodukten, die sie bereitgestellt/ermöglicht haben. Dies hilft ihnen, die Verwendung der Daten zu bestimmen, was für die Optimierung der Produktentwicklung hilfreich ist. Es hilft ihnen auch, den Wert für die Gemeinschaft zu beurteilen. Für weitere Informationen zur Einsendung von Referenzmaterial wenden Sie sich gerne an das POWER Project Team: [larc-power-project@mail.nasa.gov](mailto:larc-power-project@mail.nasa.gov)

## Glossar

### Epistemisches Programmieren

Epistemisches Programmieren (Hüsing et al., 2023) bezeichnet einen erkenntniszentrierte Programmierpraktik, bei der Programmieren als Mittel zum Erkenntnisgewinn hinsichtlich persönlich oder gesellschaftlich relevanter Phänomene oder Fragestellungen genutzt wird. Ein typisches Beispiel für ein epistemisches Programmierprojekt ist die Durchführung einer Datenexploration zur Untersuchung eines Kontextes, der durch Daten abbildbar ist. Um epistemisches Programmieren als Praktik zu erlernen, bietet sich die Durchführung projektbasierten Unterrichts an, in dem Lernende in einer vorbereiteten Umgebung Programmieren nutzen, um sich in einem tüftelartigen Prozess einer eigenen Fragestellung zu nähern. Computational Essays (siehe auch Abschnitte 2.3, 9.1.1 und 9.4.4) bieten sich hier sowohl als angestrebte Produkte als auch als sogenannte Worked Examples (siehe auch Abschnitte 2.4, 9.1.2 und 9.4.5) an, um den Erkenntnis- und Programmierprozess besonders für Programmiernoviz\*innen zu unterstützen.

### PPDAC-Zyklus

Der PPDAC-Zyklus (Wild & Pfannkuch, 1999)ist ein Modell zur Durchführung von Datenprojekten und ist in die Phasen „Problem“, „Plan“, „Data“, „Analysis“ und „Conclusions“ unterteilt. Diese Phasen können dabei zyklenartig nacheinander durchlaufen und wiederholt werden.

* In der *Problem-Phase* geht es darum, das Problem genauer einzugrenzen und zu definieren.
* In der *Plan-Phase* wird die Datenerhebung und -auswertung geplant.
* In der *Data-Phase* erfolgt die Datensammlung, das Datenmanagement und die Datenbereinigung.
* In der *Analysis-Phase* werden die Daten exploriert und darauf aufbauend Ergebnisse und Hypothesen generiert.
* In der *Conclusions-Phase* wird eine Interpretation der Ergebnisse vorgenommen und es werden Schlussfolgerungen und neue Ideen generiert.

### Epistemic Agency

Das Konstrukt Epistemic Agency bezieht sich auf die Befähigung, Wissen und Wissenspraktiken einer Community mit zu bestimmen (Miller et al., 2018; Stroupe, 2014) bzw. kognitive Kontrolle und Verantwortung über das eigene Lernen zu übernehmen (Odden et al., 2023). Lernende können folglich Epistemic Agency erlangen, indem sie aktiv in Wissens- und Erkenntnisprozesse eingebunden werden und indem ihnen Freiheiten zum eigenständigen Erkunden persönlich oder gesellschaftlich relevanter Interessen gegeben wird. Miller et al. (2018) geben hierzu vier Möglichkeiten an, um die Epistemic Agency in Lernenden zu fördern:

* Das Wissen der Schüler\*innen als Ressource für das Lernen nutzen und darauf aufbauen
* Lernende Wissen aktiv entwickeln lassen
* Wissen aufbauen, das für die Schüler\*innen nützlich ist (z.B. durch authentische Fragestellungen)
* Lernen und Wissen nutzen um Veränderungen von Strukturen oder Handlungen initiieren

### Computational Essay

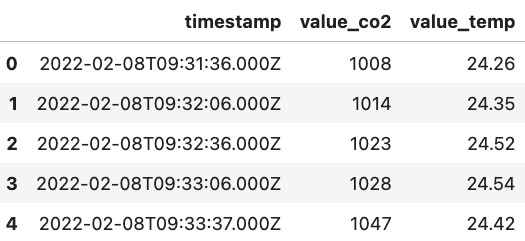
Ein Computational Essay lässt sich als ein ausformuliertes Essay beschreiben, welches neben textuellen Beschreibungen, Interpretationen und Erläuterungen insbesondere auch manipulierbaren Programmcode enthält, der direkt im Medium ausgeführt werden kann (DiSessa, 2000; Odden et al., 2023; Wolfram, 2017).

### Worked Examples

Im Kontext des Epistemischen Programmierens bezeichnet ein Worked Example ein ausgearbeitetes und dokumentiertes Programmier-Beispiel, in welchem Prozesse und Methoden angewandt werden, die für das Programmierprojektes der Schüler\*innen nützlich sein könnten (Atkinson et al., 2000; Hüsing et al., 2024b; Muldner et al., 2023). Lernende können ein Worked Example einerseits als Orientierung für den eigenen Programmierprozess verwenden als auch als „Steinbruch“, aus dem sie Code verwenden, adaptieren und erweitern können (siehe auch Lee et al. (2011) und Sentance et al. (2019)).

### Zeitreihendaten

Zeitreihendaten bezeichnen Daten, die wiederholt über einen bestimmten Zeitraum erhoben wurden. Für jeden erhobenen Wert wird auch der Zeitpunkt der Erhebung festgehalten. Zeitreihendaten ermöglichen es so, einen zeitlichen Verlauf der erhobenen Werte analysieren zu können. Beispielsweise können die Daten in einem Scatterplot dargestellt werden:



# Literatur

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. Zeitreihendaten = Daten, die in einem bestimmten Messintervall wiederholt erhoben werden und mit einem Zeitpunkt versehen sind, siehe auch Abschnitt 9.4.6 [↑](#footnote-ref-2)
2. Epistemic Agency zu entwickeln bedeutet, Wissen und Wissensprozesse eigenständig zu beeinflussen und kognitive Kontrolle und Verantwortung für das eigene Lernen zu übernehmen (Miller et al., 2018; Odden et al., 2023); siehe auch den Glossar im Abschnitt 9.4.3 [↑](#footnote-ref-3)
3. Zu dieser Debatte gibt es einen interessanten Artikel von Rule et al. (2018), der sich mit unterschiedlichen Perspektiven auf des Erstellens von Jupyter Notebooks hinsichtlich der grundlegenden Ideen „Explorieren“ und „Erklären“ in Jupyter Notebooks auseinandersetzt. [↑](#footnote-ref-4)