# Jupyter Notebook – Einsatz als digitale Lehr-Lern-Umgebung für aufgabenbasiertes Lernen am Beispiel eines Programmierkurses

Robert Ringel[1]



DOI: 10.55310/jfhead.45

## Abstract

Dieser Beitrag stellt das Potential von Jupyter Notebook für die Methodik des aufgabenbasierten Lernens in der Informatik dar. Basierend auf dem Konzept von 4C/ID werden Arbeitsmaterialien für einen Programmierlehrgang gestaltet, bei dem Jupyter Notebook als interaktive digitale Lehr-Lern-Umgebung für Vorlesungen und praktische Übungen eingesetzt wird. Neben den methodischen Grundlagen und dem konzeptionellen Aufbau der Arbeitsmaterialien reflektiert der Beitrag die subjektive Sicht von Studierenden auf die Arbeit mit diesem Medium.

1 Robert Ringel
Wissenschaftlicher Mitarbeiter, HTW Dresden, Fakultät Informatik/Mathematik
E-Mail: robert.ringel@htw-dresden.de

## Keywords

Jupyter Notebook; Programmierenlernen; Lehr-Lern-Umgebung; 4C/ID; Informatikdidaktik

## 1 Was ist Jupyter Notebook?

Jupyter Notebook ist ein digitales, webbasiertes Medium, welches seinen Ursprung in den 2010er Jahren für wissenschaftliches Rechnen an Universitäten hat. Seine Weiterentwicklung erfolgt als freie Software unter dem Titel „Project Jupyter" in der Rechtsform einer Non-Profit-Organisation. Schwerpunkte der Nutzung liegen in den Bereichen Data Science, Informatik und Naturwissenschaften, wo es zunehmend auch als Lernumgebung Einsatz findet (Schiele 2023). Juypter Notebook ist kostenfrei für alle gängigen Computer und Betriebssysteme verfügbar. Dabei ist es möglich, persönliche Installationen auf dem eigenen Computer einzurichten oder über Webbrowser auf Serverinstallationen zuzugreifen.

Im Rahmen von Lehrveranstaltungen zum Programmierenlernen kann es als einheitliches, interaktives Medium, in dem das gesamte Lehr- und Arbeitsmaterial in digitaler Form vorliegt, genutzt werden. Das Medium ermöglicht es, Programmcodes zu schreiben und auszuführen. Auch das Schreiben und Ändern von Text ist damit möglich. Zudem können Grafiken und PDF-Dokumente integriert werden. Die Verlinkung von Webseiten und Videos wird ebenfalls unterstützt. Die Verteilung der Jupyter-Notebook-Dokumente erfolgt als Dateiordner über Bildungsportale wie zum Beispiel OPAL. Lernende können ihren aktuellen Arbeitsstand des Jupyter Notebooks lokal auf dem Computer oder Server speichern und die Arbeit daran jederzeit wieder aufnehmen.

**Abbildung 1** zeigt den Ausschnitt eines Jupyter-Notebook-Dokuments mit der Programmieraufgabe aus einem Python-Kurs. Darin sind die Aufgabenformulierungen mit der Angabe des Arbeitsauftrags sowie einem Bereich zum Formulieren der Antwort erkennbar. Unter dem Textbereich steht der Python-Quelltext, in den gemäß Aufgabenstellung Codekommentare einzufügen sind. Durch Anklicken des Run-Buttons im Menü kann das Python-Programm gestartet werden. Das Ergebnis der Programmausführung bewirkt die Ausgabe der Textzeile unterhalb des Programmquelltexts. Durch die Bearbeitung der Lernaufgaben in einem eigenen Jupyter-Notebook-Dokument haben die Lernenden ihr eigenes digitales Medium zum Lösen der Aufgaben, in dem auch textuelle Informationen ergänzt werden können und in dem echtes Programmieren möglich ist. Anhand dieser Möglichkeiten von Jupyter Notebook lässt sich sein Potential für das aufgabenbasierte Lernen im Bereich der Informatikausbildung erkennen.

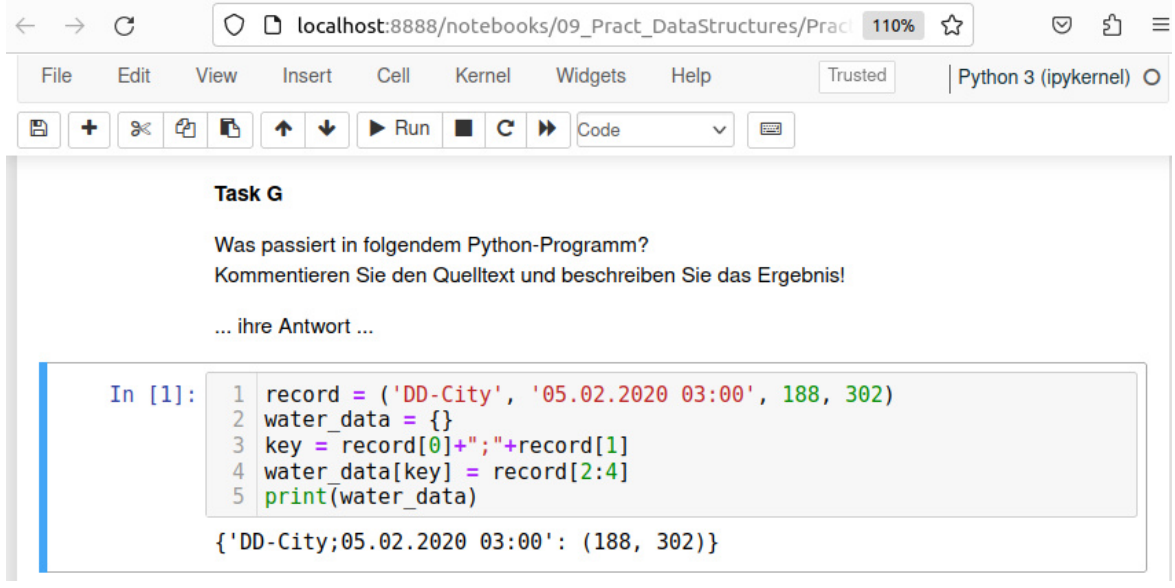


Abb. 1: Ausschnitt eines Jupyter-Notebook-Dokuments mit Darstellung einer Programmieraufgabe

## 2 Nutzung als Lehr-Lern-Umgebung im Grundlagenkurs zur Python-Programmierung

An der HTW Dresden wird seit mehr als drei Jahren ein Kurs zu den Grundlagen der Python-Programmierung angeboten. Die Methodik der Lehrveranstaltung basiert auf dem 4-Component Instructional Design (4C/ID) nach van Merriënboer und Kirschner (2018). Das Kernelement bilden fachpraktische Aufgabenfolgen mit ansteigender Schwierigkeit bei gleichzeitiger systematischer Variation der Aufgabenschwerpunkte. Mit diesem methodischen Ansatz wird der ganzheitliche Erwerb komplexer Fähigkeiten und Zusammenhänge in einem induktiven Lernprozess angestrebt. Die Aufgabenfolgen sind dabei in unterstützende Informationen und Handlungsanleitungen eingebettet. Van Merriënboer gibt grundlegende Aufgabentypen an, die sich in ihrem Grad aufgabeninhärenter Unterstützung unterscheiden (van Merriënboer & Kirschner 2018, 72). Dabei empfehlen die Autoren, am Beginn der Aufgabenfolge Lernaufgaben mit einem hohen Unterstützungsgrad zu bearbeiten, um dann systematisch zu Aufgabentypen mit einem reduzierten bis geringen Unterstützungsgrad überzugehen. Erst am Ende der Aufgabenfolge sollen klassische Gegeben-Gesucht-Problemlöseaufgaben ohne aufgabeninhärente Unterstützung bearbeitet werden. **Abbildung 2** zeigt Beispiele dieser Aufgabentypen nach 4C/ID anhand konkreter Programmieraufgaben. Dabei handelt es sich jeweils um einzelne Beispielaufgaben und nicht um eine Aufgabenfolge. Die wesentlichen Eigenschaften dieser Aufgabentypen werden im Folgenden kurz dargestellt.

Aufgaben mit hohem Unterstützungsgrad sind das Worked-out-Example und die Reverse Task. Die Lernhandlungen bei der Bearbeitung dieser Aufgaben bestehen primär im Lesen und Verstehen eines vollständigen und fehlerfreien Programmquelltextes. Zu diesem Quelltext kann beim Worked-out-Example ein ergänzender Arbeitsauftrag formuliert werden, der dazu dient, das erlangte Verständnis darzustellen. Bei der Reverse Task besteht der Arbeitsauftrag stets darin, die Aufgabenstellung zu formulieren, die zum dargestellten Ergebnis führt. Aufgaben mit einem mittleren Anforderungsgrad sind die Imitation Task und die Non-Specific-Goal Task. Bei der Imitation Task wird ein bereits bekannter Lösungsansatz direkt nachgeahmt, um damit eine ganz ähnliche Aufgabe zu lösen. Die Non-Specific-Goal Task gibt den Lernenden die Möglichkeit, die in der Aufgabe geforderten Kenntnisse und Fähigkeiten für ein selbst zu wählendes Anwendungsbeispiel einzusetzen. Beide Aufgabentypen erfordern Lernhandlungen, die Elemente vorgegebenen fachlichen Wissens und die dazugehörigen Fertigkeiten zur Anwendung bringen. Bei der Imitation Task besteht der Lösungsprozess im nahen Transfer unter Nutzung von Analogieschlüssen für einen bekannten Lösungsansatz. Die Non-Specific-Goal Task erlaubt es den Lernenden, durch die freie Wahl des Anwendungsbeispiels den Anforderungsgrad zur Aufgabenlösung auf ein individuell beherrschbares Niveau zu setzen. Die Completion Task und die Conventional Task sind die Aufgabentypen mit den höchsten Anforderungen. Bei der Completion Task ist der vorliegende Lösungsansatz zunächst zu analysieren und zu verstehen, um anschließend für diesen Ansatz die geforderte Vervollständigung zu gestalten. Bei der Conventional Task ist ausgehend vom Aufgabeninhalt eine Problemlösungsstrategie zu entwerfen und fachlich korrekt zu implementieren.

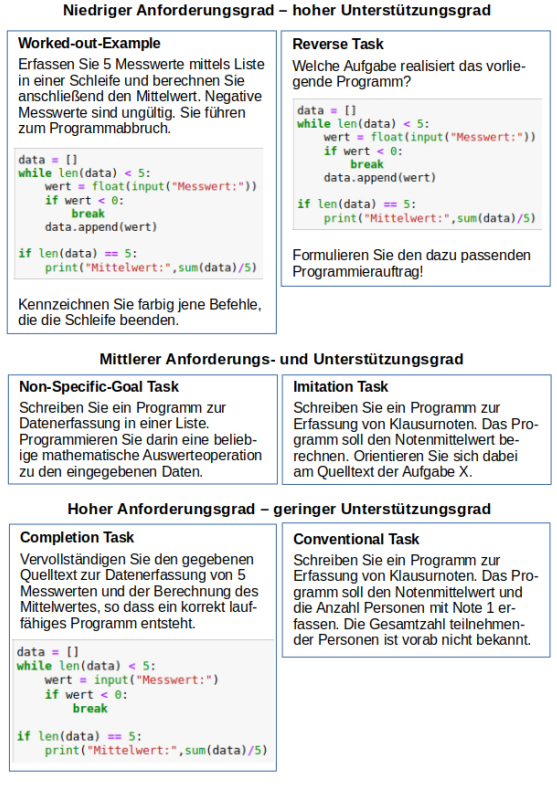


Abb. 2: Aufgabentypen nach 4C/ID am Beispiel von Programmieraufgaben

Die hier beschriebenen Aufgabentypen und die zugehörigen unterstützenden Informationen lassen sich in Jupyter Notebook unmittelbar umsetzen.

Eine der wesentlichen Aussagen in der Analyse von Robins (2019) zu den Herausforderungen des Programmierenlernens ist, dass dazu praktisches Handeln in erheblichem Umfang nötig ist. Aus diesem Grund wurde der Vorlesungsteil des Python-Kurses in Lehrvideos übertragen, die die Lernenden vor den jeweiligen Kursstunden daheim anschauen müssen. Der Inhalt dieser Lehrvideos befasst sich mit den fachlichen Grundlagen der Programmiersprache Python. Auch der Vorlesungsinhalt wird mit Hilfe von Jupyter Notebook gestaltet. Die entsprechenden Notebooks stehen den Lernenden zur Verfügung, so dass sie den Inhalt der Vorlesungsvideos aktiv selbst nachvollziehen oder mitprogrammieren können. Durch das Vorlesungsvideo und die beiden wöchentlichen Lehrveranstaltungszeiten an der Hochschule haben die Lernenden drei Mal pro Woche die Möglichkeit, mit Jupyter Notebook aktiv selbst an Programmieraufgaben zu arbeiten. Mit diesem Setting besteht für die Studierenden ein wesentlich größerer Anteil eigener praktischer Programmierarbeit als im klassischen Ablauf von Vorlesung und Übung.

## 3 Gestaltung von Lehr-Lernmaterial mit Jupyter Notebook

Grundlage der Gestaltung des Lehr-Lernmaterials für eine Lehrveranstaltung ist die Definition der Lernziele. Daraus leitet sich die inhaltliche Struktur der Themen ab, wobei jedem Thema die spezifischen Lernziele zuzuordnen sind. Das kritische Element zum Erreichen der Lernziele besteht darin, Lernhandlungen zu identifizieren, die im Ergebnis des Lernprozesses von den Lernenden beherrscht werden sollen. Diese Lernhandlungen gilt es im Prozess der Bearbeitung der Lernaufgaben zu entwickeln, zu trainieren und zu reflektieren. Mit dem Wissen um die notwendigen Lernhandlungen ist es möglich, geeignete Aufgabenstellungen zur Gestaltung des Lernprozesses, zum Beispiel in der methodischen Konzeption nach 4C/ID, zu entwerfen.

Für den Grundlagenkurs zur Python-Programmierung wurde die Methodik von 4C/ID in das Task Based Learning nach Willis (2004) eingebettet, das mit den Elementen PRE-TASK, TASK und POST-TASK eine methodische Gesamtstruktur liefert. Damit entsteht das methodische Rahmenmodell für den Kurs, wie es in **Abbildung 3** dargestellt ist

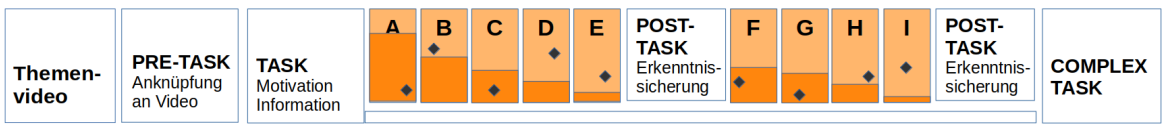


Abb. 3: Kombinierte Methodik zum aufgabenbasierten Lernen unter Nutzung der Modelle von 4C/ID und Task Based Learning

Das Themenvideo enthält im Stil einer Vorlesung die fachlich-theoretischen Grundlagen eines bestimmten Themenbereichs. Es ist vor Beginn der Arbeit an der Aufgabenfolge anzuschauen. Die Aufgabenfolge selbst beginnt mit dem Abschnitt PRE-TASK, der mit Hilfe eines kurzen Live-Codings die kognitive Aktivierung der Inhalte des Videos bewirkt. Der TASK-Baustein dient der Einführung in die eigentliche Aufgabenfolge. Er baut eine fachliche Motivation für die nachfolgenden Lernaufgaben auf und stellt die unterstützende Information dafür bereit. Anschließend erfolgt die Bearbeitung der Aufgabenfolge. Übersteigt die Aufgabenfolge den zeitlichen Rahmen einer Unterrichtsstunde, so ist am Ende der Stunde ein POST-TASK-Baustein zur Erkenntnissicherung empfehlenswert. An die dort fixierten Erkenntnisse wird bei der Fortsetzung der Aufgabenbearbeitung angeknüpft. Am Ende der Aufgabenfolge steht nach entsprechender Erkenntnissicherung eine komplexe Anwendungsaufgabe (COMPLEX TASK) als fachpraktisches Programmierproblem. Diese methodische Grundstruktur lässt sich unmittelbar als Gliederung für die Jupyter Notebook-Dokumente übernehmen.

Im Folgenden wird anhand von Beispielen aufgezeigt, wie Aufgabenfolgen zum Programmierenlernen mit Jupyter Notebook gestaltet werden können. Dazu zeigen die **Abbildungen 4, 5 und 6** Ausschnitte aus dem Jupyter Notebook des Themenbereiches „Eingabe-Berechnung-Ausgabe“ des Python-Kurses.


Abb. 4: Jupyter Notebook zur Einleitung in die Aufgabenfolge „Eingabe-Berechnung-Ausgabe“

**Abbildung 4** zeigt den einführenden Teil in die Aufgabenfolge des Themenbereiches. Direkt unter der Überschrift folgt eine kurze Textbeschreibung der Lernziele. Im Abschnitt PRE-TASK wird ein konkretes fachliches Problem vorgestellt, um anhand der Relevanz des Problems die Motivation für das Thema aufzubauen. Die Lösung des Problems erfolgt durch ein Live-Coding der Lehrkraft in der darunter stehenden Code-Zelle. Dieses Live-Coding führen die Lernenden zeitgleich mit der Lehrkraft aus. Es endet mit einem fehlerfreien, ausführbaren Python-Programm. Das Programm deckt die Kerninhalte des zugehörigen Lehrvideos ab, welches durch die Studierenden vor der Arbeit an der Aufgabenfolge angeschaut wurde. Mit diesem Vorgehen werden die Kerninhalte des Videos im Gedächtnis der Lernenden aktiviert.

Der TASK-Abschnitt beinhaltet die Folge der Lernaufgaben. Vor der ersten Aufgabenstellung wird durch die Bearbeitung einer übergeordneten verbalen Fragestellung der Problembereich für die folgenden Programmieraufgaben definiert. Hierzu findet ein kurzes Zweiergespräch mit einem/r anderen Studierenden statt, dessen Ergebnis als Text im Jupyter Notebook niederzuschreiben ist. Danach folgt eine Auflistung von Quellen mit unterstützenden Informationen. Hierzu gehören die Verlinkung des Lehrvideos, Links auf Webseiten zu diesem Themenbereich und eventuell Hinweise auf ausgewählte Lehrbuchseiten.

**Abbildung 5** zeigt die ersten drei Aufgaben der Aufgabenfolge Eingabe-Berechnung-Ausgabe. Task A ist ein Worked-Out-Example. Die Lernenden können den gegebenen Quelltext im Jupyter-Notebook ausführen und seine Funktionsweise nachvollziehen. Dabei sollen sie erkennen, was das Programm leistet. Die entsprechende Antwort ist als Text im Jupyter Notebook unter der Aufgabenformulierung von Task A einzutragen.

Task B ist eine Imitation-Task mit Angabe von unterstützenden Lernhandlungen beim Lösen der Aufgabe. Die Programmierung erfolgt direkt im Feld „...your code...", dessen Größe sich dynamisch an den Umfang des Programmquelltexts anpasst.

Task C ist eine Completion Task, bei der in einem gegebenen Quelltext Lücken mit korrekten Python-Anweisungen zu füllen sind. Die Lücken sind zusätzlich als Kommentar markiert. In der hier gezeigten Abbildung sind die Lücken bereits mit entsprechenden Python-Anweisungen gefüllt. Die Ausführung dieses Quelltexts ist im unteren Bereich der **Abbildung 5** zu sehen.

Der Abschluss einer Aufgabenfolge gemäß dem methodischen Rahmenmodell ist in **Abbildung 6** dargestellt. Hier ist im Abschnitt POST-TASK ein kurzes Python-Programm vorgegeben. Für dieses Programm sind die Teilaufgaben 1 und 2 zu bearbeiten. Die Ergebnisse der Bearbeitung werden durch die Lernenden im Jupyter Notebook aufgeschrieben. Bei Aufgabe 1 erfolgt dies als Kommentar im Programmcode und bei Aufgabe 2 als Tabelle. Damit sind wichtige Lernergebnisse dieser Aufgabenfolge dokumentiert.

**TASK A**

Was leistet das folgende Programm? Was leistet die type-Funktion?

... Ihre Antwort ...

In [ ]:
```
print("Berechnung ..... ")
d = 5
pi = 3.14
u = pi * d
print("Durchmesser d:", d)
print("Umfang:", u)
print("----- Datentypen -----")
print("d: ", d, "Datentyp:", type(d))
print("pi:", pi, "Datentyp:", type(pi))
print("U :", u, "Datentyp:", type(u))
```

**TASK B**

Programmieren Sie eines Ihrer selbstgewählten Beispiele! Lassen Sie sich mit der type-Funktion die Datentypen der Variablen anzeigen!
Vergleichen Sie Ihr Programm mit dem eines anderen Studenten - erklären Sie sich Ihre Programme gegenseitig!

In [ ]:
```
... your code ...
```

**TASK C**

Füllen Sie die Lücken in der folgenden Zusammenstellung!

In [1]:
```
print("***** Grundlegende Datentypen in der Übersicht *****")

count = 120          # Ganzzahl
print("count:", count, " - der Datentyp ist", type(count))

factor = 20.5        # <<<< Gleitkomma-Wert einsetzen
print("factor:", factor, " - der Datentyp ist", type(factor))

logic = True       # <<<<<< Wahrheitswert einsetzen
print("logic:", logic, " - der Datentyp ist", type(logic))

name = "Hallo"       # <<<<< den eigenen Namen einsetzen - String, Zeichenkette
print("name:", name, " - der Datentype ist", type(name))
```

```
***** Grundlegende Datentypen in der Übersicht *****
count: 120  - der Datentyp ist <class 'int'>
factor: 20.5  - der Datentyp ist <class 'float'>
logic: True  - der Datentyp ist <class 'bool'>
name: Hallo  - der Datentype ist <class 'str'>
```

**Abb. 5:** Jupyter Notebook mit drei Lernaufgaben der Aufgabenfolge Eingabe-Berechnung-Ausgabe

**POST-TASK - Zusammenfassung**

In [ ]:
```
name = input("Name")
height = input("Height [cm]:")
mass = input("Mass [kg]:")
h = int(height)
m = int(mass)
bmi = m / ((0.01*h)**2)
print(name, "your body-mass-index is:", bmi)
```

**Aufgabe 1**

Markieren Sie die Typumwandlungsfunktionen als Kommentar im Programm-Code.

**Aufgabe 2**

Füllen Sie die Tabelle aus.

| Variable | Wert bei Programmende | Datentyp | Zweck des Datentyps |
|---|---|---|---|
| name | Paula | string | Text speichern |
| mass | 76 | int | Store the weight |

## Komplexe Übungsaufgabe

Schreiben Sie ein Programm, welches die Entfernung zwischen zwei GPS-Koordinatenpunkten berechnet. Dabei wird angenommen, dass die beiden Punkte relative nahe beieinander liegen.

*Hinweis:* Die Eingabe der Koordinatenwerte soll mittels input erfolgen. Grundlage der Berechnung ist der Satz des Pythagoras. Anstelle der Wurzel-Funktion soll entsprechend potenziert werden.

Die folgende Webseite beschreibt das konzeptionelle Vorgehen (Abschn. "Einfachste Entfernungsberechnung"): https://www.kompf.de/gps/distcalc.html

In [ ]:
```
# This is a simple python demo to calculate the GPS distance

... write your code here ...
```


Abb. 6: Jupyter Notebook mit dem Abschlussteil der Aufgabenfolge Eingabe-Berechnung-Ausgabe

Durch eine Besprechung der korrekten Lösung dieser beiden Aufgaben wird sichergestellt, dass bei den Lernenden keine fehlerhaften Ergebnisse festgehalten werden.

Jede Aufgabenfolge eines Themenbereichs endet gemäß dem methodischen Rahmenmodell mit einer komplexen Problemlöseaufgabe. Diese besteht aus dem Text der Aufgabenstellung und ggf. ergänzenden Informationen oder Hinweisen. Dazu kommt der Bereich für die Eingabe des Programmcodes. Symbole, die Verlinkungen mit abgestuften Lösungshilfen bereitstellen, können hier ebenfalls angeordnet werden. Nach der Aufgabenlösung besteht im Abschnitt REFLECT die Möglichkeit, Erkenntnisse oder Hinweise aus der abschließenden Lösungsbesprechung einzutragen. Hier können die Lernenden individuelle Notizen vornehmen.

Die obigen Abbildungen zeigen, wie Jupyter Notebook die Implementierung von Aufgabenfolgen zum Programmierenlernen gemäß dem methodischen Rahmenmodell (siehe Abbildung 3) unterstützt. Damit bietet sich die Möglichkeit, alle Handlungen für das Erlernen des Programmierens in unterschiedlichen Aufgabentypen in einem Medium zu realisieren. Die ergänzende Einbettung von Grafiken, PDF-Dokumenten, Tabellen und Videolinks erlaubt die Gestaltung zweckentsprechender Arbeitsmaterialien. Der große Vorteil von Jupyter Notebook liegt in der Möglichkeit zur interaktiven Ausführung des geschriebenen Codes und der Niederschrift gewonnener Erkenntnisse durch die Lernenden selbst.

## 4 Qualitative Aussagen von Studierenden zur Arbeit mit Jupyter Notebook

Die Lehrveranstaltung zum Erlernen der Python-Programmierung wurde u.a. im Wintersemester 2021/22 und 2022/23 im ersten Semester des Studiengangs Wirtschaftsingenieurwesen als Pflichtlehrveranstaltung durchgeführt. Jeweils nach Abschluss des Kurses wurden zehn Studierende zufällig ausgewählt, von denen einmal neun und einmal acht Personen an Leitfadengesprächen (Döring, Bortz & Pöschl-Günther 2016, 372-373) zu wesentlichen methodischen Elementen der Lehrveranstaltung teilnahmen. Dabei wurden folgende Fragen mit Bezug zu den Lehrmaterialen unter Nutzung von Jupyter Notebook als digitale Lehr-Lern-Umgebung gestellt:

- Wie beurteilen Sie die Lehrvideos anstelle einer Vorlesung?
- Welche Aussagen treffen Sie zur Gestaltung der Aufgabenblätter mit Jupyter Notebook?
- Wie schätzen Sie Jupyter Notebook als Werkzeug zum Programmierenlernen ein? Wie haben Sie es verwendet?

Die Gesprächsinhalte wurden aufgezeichnet und anschließend schriftlich mit Bezug zum verwendeten Gesprächsleitfaden dokumentiert. Danach erfolgte eine tabellarische Zusammenstellung der Aussagen der einzelnen befragten Personen. Die dabei entstandenen Tabellen sind hier dargestellt.

**Tabelle 1** zeigt die Zusammenstellung der am häufigsten geäußerten Antwortinhalte zu Frage 1 (Beurteilung der Lehrvideos anstelle einer Vorlesung). In allen nachfolgenden Tabellen besagt die Häufigkeit der Nennungen, wie oft die rechtsstehende Aussage von interviewten Personen geäußert wurde.

Die mit Abstand häufigsten Nennungen begrüßen das Konzept eines Vorlesungsvideos anstatt einer Vorlesung und bewerten den Informationsgehalt sowie die Gestaltung der Videos als sehr gut. Zudem führen mehrere Personen an, dass sie dank der Videos im eigenen Tempo arbeiten können und dass damit auch ein wiederholtes Anschauen möglich ist. Neben dieser grundlegenden Zustimmung zum Konzept und zu seiner Realisierung mit Jupyter Notebook treffen die befragten Personen noch eine Vielzahl individueller Aussagen, die ebenfalls überwiegend positiv zu bewerten sind, hier jedoch aus Platzgründen nicht aufgeführt werden.

| HÄUFIGKEIT DER NENNUNG | AUSSAGE |
|---|---|
| 5x | das Konzept Video statt Vorlesung ist sehr gut |
| 4x | guter Informationsgehalt der Videos; gute Übersicht; gute Gestaltung |
| 3x | Lehrvideos sind sehr hilfreich, ermöglichen Arbeit im eigenen Tempo |
| 3x | Video statt Vorlesung schafft mehr Zeit für aktives Programmieren |
| 3x | wiederholtes Anschauen sowie Vorwärts-/Rückwärtsspulen im Video möglich |
| 2x | Inhalt ist sehr gut dargestellt, sehr verständlich |

**Tab. 1:** Aussagen zur Beurteilung der Lehrvideos anstelle einer Vorlesung

**Tabelle 2** enthält die Aussagen zur Gestaltung der Aufgabenblätter mit den Lernaufgaben. Diese Aussagen betreffen die gestalterische Implementierung der Lernaufgaben mit den technischen Möglichkeiten von Jupyter Notebook. Die ersten drei Nennungen zusammengenommen zeigen, dass die methodische und gestalterische Realisierung der Aufgabenblätter von den befragten Personen als sehr gelungen und lernwirksam wahrgenommen wird. Die weiteren Aussagen betreffen die Mischung von Inhalten, methodischen und gestalterischen Elementen sowie die Anpassung an die Möglichkeiten von Jupyter Notebook. Auch hier wird die positive Wahrnehmung der Studierenden sichtbar. Die Aufzählung in Tabelle 2 ist keine inhaltliche Auswahl, sie umfasst alle Aussagen der befragten Personen. Dies gilt auch für alle folgenden Tabellen.

| HÄUFIGKEIT DER NENNUNG | AUSSAGE |
|---|---|
| 5x | sehr gute Aufgabengestaltung |
| 5x | Aufbau vom einfachen zum schwierigen bringt Lernerfolge |
| 4x | Hilfestufen zur Aufgabenlösung sind nützlich |
| 2x | auch Beachtung der Theorie ist einbezogen |
| | guter Einstieg mit Pre-Task und nachfolgend schrittweise Vertiefung |
| | Praxisbezug und Live-Coding zum Einstieg sind sehr gut |
| | gut angepasst an Jupyter Notebook |
| | gute Mischung aus Text und Aufgaben |
| | man muss nicht so viel Text selber schreiben |
| | in Aufgabenblättern Vorwärts- und Rückwärtsblättern gut möglich |
| | viele offene Aufgaben zum Finden eigener Lösungen |

Tab. 2: Aussagen zur Gestaltung der Aufgabenblätter

**Tabelle 3** zeigt die Interviewaussagen zur Verwendung von Jupyter Notebook für das Programmierenlernen. Dabei machen die Studierenden Aussagen, welche Lernhandlungen sie in dieser digitalen Umgebung ausführen. Die Aussagen zeigen deutlich, dass die meisten Personen Jupyter Notebook zum Schreiben, Testen, Bearbeiten von Code und Kommentieren von Quelltext verwenden. Dies sind Handlungen, die unmittelbar mit dem Prozess des Programmierens verbunden sind. Aus den Aussagen geht auch hervor, dass den Studierenden die Möglichkeit zum Einfügen eigener Textanmerkungen oder auch zu Inhalten der Aufgabenreflexion bewusst ist, dass sie diese jedoch nicht nutzen. Gründe dafür werden in den Aussagen angedeutet. Sie lassen darauf schließen, dass Selbstreflexion zu Lernergebnissen und zu Ergebnissen der Aufgabenbearbeitung neu und ungewohnt ist und dass möglicherweise der Wert dieser methodischen Elemente für den Lernprozess nicht erkannt wird. Die befragten Studierenden des ersten Semesters sehen kein Potential für den Einsatz von Jupyter Notebook in anderen Lehrveranstaltungen.

| HÄUFIGKEIT DER NENNUNG | AUSSAGE |
|---|---|
| 9x | vorrangige Nutzung: programmieren, testen, Aufgaben bearbeiten |
| 6x | meistens zum Programmieren und Testen verwendet |
| 4x | Selbstreflexion wird nur von wenigen genutzt, z.T. ist unklar, was, in der Kursstunde gelernt wurde |
| 3x | Selbstreflexion ist neu, ungewohnt |
| 3x | einzelne Studierende schreiben Erkenntnisse ins Jupyter Notebook |
| 3x | in anderen Lehrveranstaltungen eher nicht nutzbar |
| | auch für Kommentieren von Quelltext genutzt |
| | Jupyter Notebook leider nicht in Prüfung nutzbar |
| | habe selbst Aufgaben in die Jupyter Notebooks hineinprogrammiert und entwickelt |
| | Studierende nutzen es eher selten zum Einfügen eigener Texte |

Tab. 3: Aussagen zur Verwendung von Jupyter Notebook zum Programmierenlernen

**Tabelle 4** enthält die Einschätzung von Jupyter Notebook insgesamt. Die Aussagen zeigen, dass dieses Medium für die Studierenden neu war und dass es anfangs demzufolge ungewöhnlich und schwierig zu verstehen war (eine Nennung). Die Mehrzahl der Angaben deutet auf eine sehr gute Nutzbarkeit für das Programmieren hin. Zudem wird nochmals die inhaltliche Gestaltung der Arbeitsmaterialien mit Jupyter Notebook positiv erwähnt. Die Interviewaussagen der Studierenden zu Jupyter Notebook belegen eindrucksvoll, dass es damit möglich ist, lernwirksame Arbeitsmaterialien für eine interaktive Lehr-Lern-Umgebung zum Programmierenlernen zu gestalten. Diese Aussage bezieht sich sowohl auf die Gestaltung von Lernvideos als auch auf die Gestaltung aufgabenbasierter Arbeitsmaterialien.

Obwohl die Beachtung der psychologischen Grundbedürfnisse des Menschen im Sinne der Selbstbestimmungstheorie von Ryan und Deci (2017) kein Entwurfsziel des Kurses war, so ermöglichen die Arbeitsmaterialien und ihre methodische Konzeption die Erfüllung der Bedürfnisse nach Kompetenzerleben, sozialer Eingebundenheit und Autonomie. Das Erreichen unmittelbarer Programmiererfolge durch den Einstieg mit einfachen Aufgaben und die direkte Unterstützung der Programmausführung fördern das Kompetenzerleben. Eine Vielzahl von Partneraufgaben bis hin zum gemeinsamen Pair-Programming sind bewusste Gestaltungsmittel der Aufgabenfolgen zur Förderung sozialer Eingebundenheit. Autonomie wird durch das Anregen eigener Programmierideen sowie Wahlmöglichkeiten bzgl. eigener Anwendungsbereiche für Inhalte von Programmieraufgaben und bzgl. alternativer Lösungsansätze unterstützt. Damit sind diese Elemente der Förderung von Kompetenzerleben, sozialer Eingebundenheit und Autonomie wichtige Bausteine für das individuelle Wohlbefinden und die Motivation der Lernenden.

| HÄUFIGKEIT DER NENNUNG | AUSSAGE |
|---|---|
| 2x | Jupyter Notebook ist ein komplett neues Medium |
| 2x | gute Möglichkeit, in den Aufgaben vorwärts oder rückwärts zu blättern |
| | Jupyter Notbook ist viel besser und übersichtlicher zum Lernen als Visual Studio |
| | optimales Arbeitsmittel für das Programmieren und die Selbstreflexion |
| | gute Materialzusammenstellung inklusive Bilder |
| | inhaltlich gut geordnet |
| | sehr gute Lösungshilfen zu den Aufgaben |
| | Jupyter Notbooks sind anfangs schon ungewöhnlich und schwierig zu verstehen |

**Tab. 4:** Aussagen zur Einschätzung von Jupyter Notebook

## 5 Zusammenfassung

Dieser Beitrag berichtet von den Erfahrungen bei der Verwendung von Jupyter Notebook als digitale Lehr-Lern-Umgebung für das aufgabenbasierte Lernen am Beispiel eines Programmierlehrgangs. Dabei zeigt sich, dass es unter Verwendung eines geeigneten methodischen Rahmenkonzeptes mit Jupyter Notebook möglich ist, hochwertige Arbeitsmaterialien für Vorlesungen und praktische Übungen zu gestalten. Die Interviewaussagen von Teilnehmerinnen und Teilnehmern dieses Kurses belegen dies eindrucksvoll.

Die Anwendung von Jupyter Notebook unter Nutzung der Methodik des kombinierten Rahmenmodells zum aufgabenbasierten Lernen (siehe Abbildung 3) ist in vielen anderen Informatik-Lehrveranstaltungen denkbar. Voraussetzung für aktives Programmieren ist das Vorhandensein eines entsprechenden Jupyter-Kernels. Gegenwärtig sind viele der gängigen Programmiersprachen bis hin zu Datenbanken verfügbar. Eine aktuelle Liste kann in Jupyter (2023) eingesehen werden.

Darüber hinaus erscheint es machbar, Jupyter Notebook und die hier vorgestellte Methodik zum aufgabenbasierten Lernen in Lehrveranstaltungen anderer Ingenieurdisziplinen einzusetzen. Dabei kommt es darauf an, möglichst auch in diesen Kursen eine direkte Bearbeitung der Lernaufgaben am besten mit einer Ergebnisdarstellung unmittelbar im Jupyter Notebook einzurichten.

Die aktuellen Jupyter-Notebook-Dokumente des hier beschriebenen Python-Lehrgangs sind als Open Educational Ressource unter entsprechender Lizenz im OPAL der HTW Dresden unter folgendem Titel verfügbar: Python programmieren lernen (OPAL 2023)

## 6 Literaturverzeichnis


Döring, N., Bortz, J. & Pöschl-Günther, S. (2016). Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften. 5. vollständig überarbeitete, aktualisierte und erweiterte Auflage. Berlin, Springer.-

Jupyter (2023). Jupyter kernels. GitHub. https://github.com/jupyter/jupyter/wiki/Jupyter-kernels (21.12.2023).

van Merriënboer, J. & Kirschner, P. (2018). Ten steps to complex learning. A systematic approach to four-component instructional design. Third edition. New York, Routledge, Taylor & Francis Group.

OPAL (2023). Online platform for academic learning and teaching. https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/2395822490l?3 (23.11.2023)

Robins, A. (2019). Novice Programmers and Introductory Programming. In: The Cambridge Handbook of Computing Education Research. S. A. Fincher & A. V. Robins (Eds.), Cambridge Handbooks in Psychology. Cambridge University Press

Ryan, R. & Deci, E. (2017). Self-determination theory basic psychological needs in motivation, development, and wellness. The Guilford Press.

Schiele, V. (2023). Jupyter Tutorial 1.1.1. https://jupyter-tutorial.readthedocs.io/de/latest/intro.html (11/21/2023).

Willis, J. (2004). A framework for task-based learning. Longman handbooks for language teachers. Harlow: Longman.



**Zitiervorschlag:**

Ringel, R. (2024). Jupyter Notebook. Einsatz als digitale Lehr-Lern-Umgebung für aufgabenbasiertes Lernen am Beispiel eines Programmierkurses. In: Perspektiven auf Lehre. Journal for Higher Education and Academic Development, 4(1), 1-12.

DOI: 10.55310/jfhead.45