Ruprecht-Karls-Universität Heidelberg
Fakultät für Verhaltens- und Empirische Kulturwissenschaften
Institut für Bildungswissenschaft
Sommersemester 2025

Einführung in die Schulpädagogik
Reflexionsaufgaben

Schmidt, Mike

4748498

Mike@schmidt-gp.de

Studiengang (BA-LA-Option, GymPO | 2009) / Mathematik, Informatik

Einführung in die Schulpädagogik

Benoteter Leistungsnachweis

Inhaltsverzeichnis

Vorwort **2**

Rahmenbedingungen **2**

Phase I: Instruktionsphase **3**

Aufbau einer Unterrichtsstunde in der Instruktionsphase *3*

1. Quiz 3
2. Live Coding 3
3. Übungsphase 4
4. Reflektion 5

Einführung *5*

Themen *7*

Termin 1 7
Termin 2 8
Termin 3 8
Termin 4: 9

Abschluss *10*

Phase II: Ko-Konstruktion **11**

Auswahl der Lernpfade *11*

Lernpfad 1: Objektorientiertes Programmieren 11
Lernpfad 2: Algorithmen 12
Lernpfad 3: Graphen 13

Einteilung der Gruppen *13*

Lernen mittels Lernpfad *14*

Übergang zur authentischen Leistung *16*

Phase III: Erstellung der Authentischen Leistung **17**

Aufgabenstellung *17*

Meilenstein *18*

Fertigstellung der Authentischen Leistung *19*

Benotung *20*

Reflektion der Einheit *21*

Bibliografie **22**

Vorwort

Informatik ist an den Gymnasien in Deutschland noch ein sehr dünn besetztes Fach. Auch der Bildungsplan weist einige Lücken auf, sodass die Grundlage von dieser Einheit nicht nur im Bildungsplan beruht. Stattdessen wurden auch eigenen Erfahrungen des Faches im Abitur und Gesprächen mit der entsprechenden Lehrkraft des Faches miteinbezogen.
Alle Materialien, die zusätzlich zu dieser Arbeit angefertigt wurden finden sie auch unter folgendem Link https://github.com/neogp05/Take-Home-Exam_SP25_Schmidt (ohne Download von Dateien einsehbar). Dort befinden sich auch zusätzliche Dateien wie Programme und Quizze, die man schlecht als physisches Exemplar an diese Arbeit anhängen kann, da diese dadurch die Interaktivität verlieren.
Sämtliche Arbeitsblätter, Quizze und anderen Materialien wurden extra händisch für diese Einheit entwickelt und stammen aus keinen anderen Quellen.

Rahmenbedingungen

Die Unterrichtseinheit ist für ein Allgemeinbildendes Gymnasium in Baden-Württemberg konzipiert. Diese Einheit im Rahmen des Wahlfaches Informatik in der Kursstufe statt. Dabei gehen wir von typischerweise 2 Schulstunden wöchentlich in Form einer Doppelstunde aus. Die Einheit ist für eine Länge von 12 Wochen mit anschließender Veranstaltung geplant. Als Überthema aus dem Bildungsplan gilt dabei das Thema „Implementation und Algorithmen“, wobei auch andere Themen einen Teil der Einheit ausmachen. Typische für ein solchen Kurs gehen wir mit einer Anzahl von Schüler:innen zwischen 12 und 16 aus.

Das Gymnasium sollte zu dem einen Computerraum mit mindestens so vielen Arbeitsplätzen mit Rechnern wie Teilnehmer an dem Kurs verfügen. Außerdem ist eine Beamer oder größerer Bildschirm für den Unterricht erforderlich. Diese Computer sollten zudem alle über das kostenlose Programm „Visual Studio Code“ verfügen. Des Weiteren muss ein git-Server für jede Gruppe, die sich in Phase II bilden wird, aufgesetzt werden, damit diese mittels dem „git“-plugin in VS-Code zusammen an einem Programm arbeiten können.

Die Schüler:innen benötigen für diese Deeper-Learning Einheit keine speziellen Vorkenntnisse. Das erlangte Wissen in Englisch und Mathematik bis zur Oberstufe in Kombination mit dem sicheren Umgang mit einem Desktop Computer reichen vollkommen aus. Dabei ist die Grundlage in der Mathematik für die Aspekte im Bereich der Logik und einfachen Berechnung und im Bereich Englisch das Verständnis teilweiser komplexerer englischer Sprachen in Erklärungen und Blog-Beiträgen entscheidend.

Phase I: Instruktionsphase

In der Instruktionsphase erarbeiten die Schüler:innen die Grundlagen von Python durch einen klar strukturierten Ablauf bestehend aus Quiz, interaktiver Einführung per Live Coding, eigenständiger Übungsarbeit und abschließender Reflexion. Diese Struktur wiederholt sich über die ersten vier Doppelstunden und schafft so einen kontinuierlichen Lernrhythmus, welcher sich aber auch direkt an den Lehrfortschritt der Schüler anpasst.

Aufbau einer Unterrichtsstunde in der Instruktionsphase

1. Quiz

Die Unterrichtsstunde beginnt mit einem kurzen interaktiven Quiz, welcher über Plattformen wie Particify durchgeführt wird. Die Fragen beziehen sich auf Inhalte der vorherigen Einheiten und sollen sowohl den aktuellen Lernstand der Schüler:innen sichtbar machen als auch mögliche Defizite aufzeigen. In der ersten Stunde kann das Quiz zudem genutzt werden, um das Vorwissen der Schüler:innen gezielt zu überprüfen.

Die Schüler:innen beantworten die Fragen eigenständig an ihren PCs. Anschließend geht die Lehrkraft die Fragen Schritt für Schritt durch, erklärt die richtigen Antworten und erläutert die Hintergründe der falschen Antworten. Durch das direkte Feedback kann die Lehrkraft den Umfang der Erklärungen flexibel an den Wissensstand der Klasse anpassen: Haben die meisten Schüler:innen die Frage richtig beantwortet, reicht eine kurze Erklärung aus, zeigen sich jedoch größere Defizite, wird die Wiederholung vertieft.

Das Quiz umfasst sowohl kurze Verständnisfragen als auch kleine Code- und Algorithmus Beispiele. Auf diese Weise wird sichergestellt, dass keine Schüler:innen mit Wissenslücken aus vorherigen Einheiten in die neue Einheit starten und es wird eine solide Grundlage für die folgenden Deeper-Learning-Phasen geschaffen

2. Live Coding

Nach dem Quiz folgt die Einführung in die neuen Inhalte durch Live Coding, bei dem die Lehrkraft in Echtzeit programmiert und den Code Schritt für Schritt erklärt. Dabei werden grundlegende Elemente der Python-Syntax vermittelt, wie Variablen, Datentypen, Schleifen, Bedingungen und Funktionen.

Die Schüler:innen können direkt eigene Ideen ausprobieren oder kurze Vorschläge einbringen, die sofort im Code getestet werden. So sehen sie unmittelbar, wie sich Änderungen auswirken, profitieren bereits in dieser Phase von vielfältigen Ideen und lernen typische Fehler zu erkennen.

Dieses Vorgehen legt den Grundstein für die Python-Syntax: Konzepte werden anschaulich erklärt, typische Fehler werden sichtbar und die Schüler:innen entwickeln ein praktisches Verständnis der Basisfunktionen, das sie für anschließende eigene Programmieraufgaben nutzen können. Dabei wird der Bogen von Theorie und Praxis direkt gespannt, sodass das Lernen nicht nur theoretisch, sondern auch unmittelbar anwendungsorientiert erfolgt.

Der Fokus liegt auf der allgemeinen Syntax von Python, die die Basis für die weiteren Phasen bildet. Gleichzeitig wird das logische und algorithmischen Denken gefördert, das in der Informatik eine zentrale Rolle spielt.

Bei der Live-Coding-Einheit hält sich die Lehrkraft nicht an ein genau festgelegtes Programm, sondern nutzt die Funktionen der Einheit um einen flexiblen, an die Ideen der Schüler:innen angepassten Code zu erzeugen, dies motiviert zusätzlich auch die Schüler:innen später selbst aktiv zu experimentieren und kreative Lösungswege zu entwickeln.

Um die Inhalte der Einheit nachhaltig zu verankern, erstellen die Schüler:innen parallel zum Live-Coding ein eigenes Glossar zu den wichtigsten Python-Funktionen. Darin halten sie zu jeder vorgestellten Funktion den Namen, eine kurze Erklärung (in Stichpunkten oder einem Satz) sowie ein kleines Codebeispiel fest. Zusätzlich markieren sie in diesem Beispiel mit Pfeilen die Arbeitsweise der Funktion und die Bedeutung der einzelnen Parameter.

Damit das Glossar sorgfältig geführt werden kann, lässt die Lehrkraft beim Live-Coding regelmäßig kurze Pausen, sodass alle Schüler:innen ihre Dokumentation vervollständigen können. Auf diese Weise wird sichergestellt, dass die Lernenden aktiv zuhören und den Stoff nicht nur passiv aufnehmen.

Das Glossar dient nicht nur als begleitende Unterstützung während der Einheit, sondern auch als Nachschlagewerk für zukünftige Programmieraufgaben, auf welche die Schüler:innen bei Unsicherheiten jederzeit zurückgreifen können, ohne sich durch lange und verwirrende Dokumentationen online durchzuklicken.

3. Übungsphase

In der Übungsphase bearbeiten die Schüler:innen Übungsaufgaben eigenständig, wobei dabei auch der Austausch mit dem Nachbarn möglich ist, um Ideen oder Lösungsansätze zu

diskutieren. Die Aufgaben sind dabei so strukturiert, dass zunächst klar definierte und eher striktere Aufgabenstellungen bearbeitet werden, um die grundlegenden Konzepte zu festigen und die Lernenden auf den „richtigen Weg“ zu bringen. Diese dienen hauptsächlich dem Ziel die korrekte Syntax beizubringen, also die Programmiersprache.

Im Anschluss folgen offenere Aufgaben, welche bewusst Freiräume für Experimente lassen. Die Schüler:innen können eigene Ideen ausprobieren, kreative Lösungen entwickeln und unterschiedliche Ansätze testen. Dies fördert aktiv das logische beziehungsweise algorithmisches Denken. Durch diese Abfolge - erst Fundament, dann Experiment - wird sichergestellt, dass die Basis sicher gelegt wird, bevor die Lernenden ihre Kreativität einsetzen.

Parallel stehen Erklärvideos und ergänzende Materialien zur Verfügung, die detaillierte Erklärungen bieten und es den Schüler:innen ermöglichen, Unklarheiten selbstständig zu beseitigen. So wird die Eigenständigkeit gefördert, während die Lehrkraft gezielt dort zusätzliche Unterstützung geben kann, wo sie benötig ist. Durch diese Kombination aus klaren Leitplanken, experimentellen Freiräumen und kollaborativem Austausch entwickeln die Schüler:innen ein Verständnis der Programmiergrundlagen und lernen, Probleme selbstständig und kreativ zu lösen.

4. Reflektion

Am Ende der Übungsphase folgt ein kurzes Quiz als Reflexion. Die Schüler:innen beantworten dabei gezielt Fragen zu den bearbeiteten Aufgaben, um ihren Lernfortschritt zu überprüfen, sowie allgemeine Fragen zur Python-Syntax des aktuellen Themas. So können offene Verständnislücken sofort erkannt und von der Lehrkraft gezielt geklärt werden. Gleichzeitig erhalten die Schüler:innen die Möglichkeit, ihren eigenen Lernstand einzuschätzen und die Inhalte der Übungsphase zu festigen.

Einführung

In der ersten Stunde der instruktiven Phase dient das Quiz dazu, die Vorkenntnisse der Schüler:innen zu erfassen. Für diese Einheit sind zwar Programmiererfahrungen von Vorteil, aber keinesfalls notwendig. Erwartet wird lediglich, dass die Schüler:innen mit dem Umgang am Computer vertraut sind, was im Wahlfach Informatik in der Oberstufe in der Regel gegeben ist. Darüber hinaus sind wie bereits erwähnt keine speziellen Vorkenntnisse erforderlich.

Das Quiz hilft der Lehrkraft, die Gruppe besser einzuschätzen und herauszufinden, wie groß die Spannweite an programmiertechnischen Fähigkeiten zu Beginn ist.

Direkt im Anschluss erhalten die Schüler:innen einen kurzen Überblick über die Ziele der Einheit.

Ziele für den Individuellen Schüler:innen

- Ich verstehe, wie ein Programm in Python aufgebaut ist, und kann erklären, was im Code passiert.
- Ich kann selbst Python-Programme schreiben, die schon etwas schwierigere Probleme lösen.
- Ich kann meinen eigenen Code so erklären, dass auch andere ihn nachvollziehen können.

Ziele für das gesamte Team

- Wir überlegen gemeinsam, wie man ein Problem Schritt für Schritt logisch angehen kann.
- Wir üben, miteinander zu reden und Ideen auszutauschen, damit wir bessere Lösungen finden.
- Wir lernen, große Aufgaben aufzuteilen und unsere einzelnen Teile am Ende wieder zu einem Ganzen zusammenzufügen.

Dabei hat die Einheit zwei Primäre Ziele:

Zum einen wollen wir, dass jeder der Schüler:innen die Grundfunktionen von Python versteht. Da Python eine recht einfache Syntax besitzt, können hier die Schüler:innen vor allem den Aufbau und die Grundkonzepte dahinter kennenlernen, den sie später dann mit veränderter Syntax auf andere Sprachen übertragen können. Aber wir wollen uns nicht nur anschauen, wie man ein Programm plant und schreibt, sondern etwas viel Nachhaltigeres:

Nämlich zweitens, das Koordinierte Arbeiten im Team. Dieses stellt nicht nur in der Informatik einen sehr wichtigen Skill da, sondern in Zeiten der Globalisierung und des Internets ist die Teamfähigkeit und das koordinierte Arbeiten im Team in fast allen Bereich von elementarer Bedeutung. Hierbei simulieren wir durch die Phase III, mittels des Projektes, eine realistische Teamumgebung und setzen uns unbewusst gleichzeitig mit Themen wie Führungspositionen, Zeitmanagement, Aufgabenteilung und Koordination auseinander.

Themen

Die Themen, die wir dabei an den vier Terminen durchgehen sind folgende:

Termin 1

Themen: Ein- & Ausgabe, Variablen, Rechenoperationen, Kommentare.

In der ersten Stunde beschäftigen wir uns mit den Grundlagen der Ein- und Ausgabe in Python. Die Schüler:innen lernen zunächst die Befehle *print()* und *input()* kennen, mit denen ein Programm Informationen auf der Konsole ausgeben und gleichzeitig Eingaben von Nutzer:innen entgegennehmen kann.

Darauf aufbauend wird das Konzept der Variablen eingeführt. Werte können in einer Variablen gespeichert und anschließend weiterverarbeitet werden. Dabei wird gezeigt, dass Python mit unterschiedlichen Datentypen arbeiten kann, etwa mit Zahlen oder Texten (Strings). Besonders wichtig ist in diesem Zusammenhang die Erkenntnis, dass Eingaben über *input()* standardmäßig als Text gespeichert werden. Damit diese Eingaben für Berechnungen genutzt werden können, müssen sie in eine Zahl umgewandelt werden, beispielsweise mit *int()* für ganze Zahlen oder *float()* für Dezimalzahlen.

Im nächsten Schritt folgt die Einführung in die Grundrechenoperationen. Neben Addition, Subtraktion, Multiplikation und Division lernen die Schüler:innen auch erweiterte Operatoren, wie die Ganzzahldivision (//), den Restoperator (%) sowie die Potenzierung (**) kennen. Dadurch können sie bereits erste kleine Rechenprogramme entwickeln, welche auf Nutzereingaben basieren.

Zum Abschluss wird auf die Bedeutung von Kommentaren im Quellcode eingegangen. Mit dem Zeichen # können Notizen oder Erklärungen eingefügt werden, die den Code übersichtlicher machen und diesen sowohl für die Schüler:innen selbst, als auch für andere nachvollziehbarer gestalten.

Die Stunde ist bewusst kompakt gehalten, denn in der anschließenden Übungsphase sollen die Schüler:innen ihre ersten eigenen Programme schreiben. Dabei sammeln sie grundlegende Erfahrungen mit der Programmierumgebung, üben das Umwandeln von Eingaben in verschiedene Datentypen, führen Berechnungen durch und lernen den Nutzen einer klaren Strukturierung des Codes kennen.

Termin 2

Themen: Vergleichsoperatoren, logische Operatoren, Bedingungen, Schleifen.

Am zweiten Termin beschäftigen wir uns ausführlich mit dem Thema Bedingungen. Dabei lernen die Schüler:innen die Struktur der Befehle *if*, *elif* und *else* kennen und setzen diese mithilfe verschiedener Vergleichsoperatoren um, darunter ==, *!*=, <, >, <= und >=. Ergänzend behandeln wir die logischen Operatoren *and*, *or* und *not*, mit denen komplexere Bedingungen formuliert werden können.

Bedingungen sind ein zentrales Element der Programmierung, da sie Programmen die Fähigkeit verleihen, auf unterschiedliche Zustände zu reagieren und Entscheidungen zu treffen. Durch den Einsatz von Vergleichsoperatoren können Variablen direkt miteinander verglichen werden, beispielsweise ob zwei Werte gleich (==) sind oder ob ein Wert größer oder gleich (>=) als ein anderer Wert ist. In der Einheit wird besonderer Wert daraufgelegt, dass die Schüler:innen den Aufbau einer einfachen if-Bedingung, sowie die Erweiterung durch *elif* und *else* verstehen und anwenden können.

Darüber hinaus werden in diesem Termin auch die Schleifen thematisiert. Die Schüler:innen lernen die Anwendung von *while*-Schleifen und *for*-Schleifen, ergänzt durch die Funktionen *range()*, *break* und *continue*. Dabei wird gezeigt, wie sich Schleifen auf unterschiedliche Weise aufbauen lassen und welche Vorteile die jeweilige Variante in verschiedenen Situationen bietet. Schleifen stellen, neben den Bedingungen, einen weiteren Hauptbestandteil der Programmierung dar. Während Bedingungen dafür sorgen, dass Programme Entscheidungen treffen können, ermöglichen Schleifen das wiederholte Ausführen von Programmteilen. Gerade in Kombination entfalten beide Konzepte ihre volle Bedeutung, da sie die Grundlage für nahezu alle komplexeren Programmstrukturen bilden. Aus diesem Grund nimmt diese Einheit innerhalb der vier Termine eine besonders zentrale Rolle ein.

Termin 3

Themen: Datenverarbeitung mit: Liste, Dictionary

Am dritten Termin liegt der Schwerpunkt auf dem Umgang mit Datenstrukturen in Python. Die Schüler:innen machen sich zunächst den praktischen Einsatz von Listen vertraut. Dazu gehört, wie man eine Liste erstellt, wie man auf einzelne Elemente zugreift und wie man Werte dynamisch hinzufügen (*append()*) oder entfernen (*remove()*) kann. Auch die Bestimmung der Länge einer Liste mithilfe der Funktion *len*() wird thematisiert. Aufbauend darauf wird gezeigt,

wie man mit for-Schleifen über Listen iterieren kann, um die gespeicherten Elemente gezielt zu bearbeiten oder zu verändern.

Darüber hinaus werden die Dictionaries als weitere zentrale Datenstruktur eingeführt. Diese ermöglichen das Speichern von Informationen in Schlüssel-Wert-Paaren und sind damit besonders für strukturierte Daten geeignet. Die Schüler:innen üben, wie sie auf Werte zugreifen, neue Schlüssel-Wert-Paare hinzufügen und bestehende Einträge verändern können.

Ein wichtiger didaktischer Bezug liegt in der Verknüpfung mit den Schleifen aus dem vorherigen Termin. Während dort Schleifen zunächst eher abstrakt behandelt wurden, zeigt sich nun ihr praktischer Nutzen in Verbindung mit Datenstrukturen. Mit nur kleinen Anpassungen der Syntax können Schleifen verwendet werden, um nicht nur einzelne Daten zu bearbeiten, sondern auch alle Elemente einer Liste oder eines Dictionaries systematisch zu verarbeiten.

Die Auseinandersetzung mit Datenstrukturen bildet einen wesentlichen Bestandteil der Programmierung. Die Fähigkeit, Daten zu speichern, zu organisieren und flexibel zu verändern, ist die Grundlage für das Erstellen komplexerer Programme. Besonders für den späteren Lernpfad im Bereich Algorithmen ist dieses Wissen unverzichtbar, da dort häufig mit großen Datenmengen in unterschiedlichen Strukturen gearbeitet wird. Die frühe Einführung in Listen und Dictionaries legt somit das Fundament für ein tieferes Verständnis algorithmischer Konzepte.

Termin 4:

Themen: Funktionen, Parameter, globale/lokale Variable

Am vierten Termin beschäftigen wir uns intensiver mit dem Thema Funktionen in Python. Dabei lernen die Schüler:innen, wie eigene Funktionen mit dem Schlüsselwort *def* erstellt werden, wie Werte über Parameter übergeben werden und wie man Ergebnisse mithilfe von Rückgabewerten (*return*) aus Funktionen zurückerhält. Zusätzlich betrachten wir den Einsatz von Standardwerten für Parameter, sodass bestimmte Argumente beim Funktionsaufruf optional bleiben.

Funktionen stellen ein zentrales Konzept in der Programmierung dar. Sie ermöglichen es, Aufgaben in wiederverwendbare und klar strukturierte Einheiten zu zerlegen, welche mit unterschiedlichen Werten oder in verschiedenen Situationen eingesetzt werden können. Ein

wichtiger Aspekt ist daher, dass die Schüler:innen lernen, Funktionen zu analysieren und zu verstehen, wie sich unterschiedliche Parameter auf die Funktionsweise auswirken.

Ein weiterer Schwerpunkt liegt auf dem Umgang mit globalen und lokalen Variablen. Hierbei wird insbesondere darauf eingegangen, wie globale Variablen innerhalb von Funktionen genutzt oder verändert werden können, um Daten zwischen verschiedenen Programmteilen auszutauschen. Gleichzeitig wird die Abgrenzung zu lokalen Variablen thematisiert, um spätere Fehlerquellen und Missverständnisse im Programmaufbau zu vermeiden.

Abschluss

Die erste Phase endet mit einem kompakten Verständnistest.

Dieser ist ähnlich aufgebaut wie die bisherigen Übungsblätter und enthält verschiedene Aufgabentypen: Dazu gehören die Analyse von Programmausgaben, das Ergänzen von Lücken in Programmen sowie die Suche und Korrektur von Fehlern. Zum Abschluss gibt es außerdem eine Aufgabe, in der die Schüler:innen einen kleinen Programmblock selbst entwickeln.

Der Test ist deshalb besonders wichtig, weil er sicherstellt, dass die grundlegenden Operationen der Programmiersprache verstanden wurden. Diese lassen sich mit den Vokalen einer gesprochenen Sprache vergleichen: Sie bilden die Basis und sind Voraussetzung dafür, dass später komplexere Strukturen sinnvoll eingesetzt werden können. Wenn diese Grundlagen sicher beherrscht werden, wird das weitere Lernen deutlich leichter.

Zudem fließt der Test auch in die Benotung ein (vgl. Kapitel Benotung). Sein Aufbau soll dazu beitragen, dass die Schüler:innen ein besseres Verständnis für Programmcode entwickeln. Dabei lernen sie beispielsweise, schon vor der Ausführung vorherzusagen, welche Ausgabe der Code erzeugt, oder typische Fehler schneller zu erkennen. Diese Fähigkeiten sind für das spätere Arbeiten in der Programmierung zentral, da sie zu effizienteren und präziseren Lösungen führen.

Ein weiterer Vorteil liegt in der Abstufung der Aufgaben nach Schwierigkeit. So können unterschiedliche Leistungsniveaus berücksichtigt werden, was eine faire und aussagekräftige Bewertung ermöglicht.

Phase II: Ko-Konstruktion

Auswahl der Lernpfade

Zu Beginn der Phase II können sich die Schüler:innen ganz nach dem Voice-and-Choice Prinzip für einen der drei Lernpfade entscheiden. Dabei stehen folgende zur Auswahl:

Lernpfad 1: Objektorientiertes Programmieren

Der erste Lernpfad befasst sich mit der objektorientierten Programmierung (OOP). Dieser Ansatz unterscheidet sich vom prozeduralen Programmieren dadurch, dass nicht mehr in erster Linie einzelne Befehlsabfolgen im Vordergrund stehen, sondern die Arbeit mit Objekten, welche bestimmte Eigenschaften besitzen und über Methoden miteinander interagieren können.

Gerade für Anfänger:innen bietet die OOP einen anschaulichen und oft intuitiveren Zugang. Sie ermöglicht es, reale Strukturen und Abläufe auf Programme zu übertragen, indem Objekte als Modelle für Dinge oder Systeme genutzt werden. Dadurch können Schüler:innen Zusammenhänge leichter nachvollziehen und Programmabläufe besser strukturieren. Besonders für Lernende mit einem Interesse am Ingenieurwesen ist dieser Ansatz sinnvoll, da hier Ideen wie Verknüpfungen, Hierarchien und Wechselwirkungen zwischen Objekten im Vordergrund stehen, ähnlich wie bei technischen Systemen oder Maschinen, die miteinander verbunden sind.

Im Rahmen dieses Lernpfades lernen die Schüler:innen zunächst, was Objekte sind, wie sie aufgebaut sind und wie man sie erzeugt. Anschließend werden wichtige Prinzipien der OOP behandelt, darunter Konstruktoren, Vererbung und die Interaktion zwischen Objekten. Diese Inhalte vertiefen nicht nur das logische Denken, sondern bilden auch die Grundlage für den späteren Umgang mit größeren, komplexeren Programmen.

Zur praktischen Anwendung bieten sich verschiedene Projektideen an. So könnte beispielsweise ein Smartphone mit seinen Hardwarekomponenten (Display, Knopf, Sensor) als System aus Objekten modelliert werden, welche miteinander interagieren. Auch kleinere Anwendungen wie ein To-Do-Listen-Manager lassen sich umsetzen und zeigen, wie OOP zur Strukturierung und Organisation von Programmen eingesetzt werden kann.

Lernpfad 2: Algorithmen

Im Lernpfad Algorithmen beschäftigen sich die Schüler:innen hauptsächlich mit verschiedenen Sortieralgorithmen. Dieser Bereich richtet sich besonders an diejenigen, die über ein gutes logisches Denkvermögen verfügen und komplexe Abläufe nachvollziehen können, da Algorithmen teilweise anspruchsvoller zu verstehen sind.

Im Gegensatz zum objektorientierten Programmieren bewegen sich die Schüler:innen hier näher an praktischen und alltäglichen Anwendungen. Sie erkennen, dass selbst scheinbar einfache Operationen, wie das Sortieren von Daten nach Größe, mitunter komplexe Abläufe erfordern. Durch die Auseinandersetzung mit Algorithmen erhalten die Schüler:innen gleichzeitig einen Einblick in einen der Kernbereiche der Informatik: die Datenverarbeitung. Für Schüler:innen, die sich vorstellen können, später beruflich im Bereich Informatik tätig zu sein, bietet dieser Lernpfad eine erste Orientierung und die Möglichkeit, grundlegende Prinzipien der Datenorganisation und -verarbeitung kennenzulernen.

Der Lernpfad beginnt mit einfachen und intuitiven Sortieralgorithmen und steigert sich später zu effizienteren, aber komplexeren Verfahren, welche in modernen Anwendungen standardmäßig eingesetzt werden. Dies ermöglicht einen realitätsnahen Bezug: Die Schüler:innen verstehen, welche Abläufe im Hintergrund ablaufen, wenn Daten auf digitalen Geräten sortiert werden. Gleichzeitig vermittelt die Beschäftigung mit verschiedenen Algorithmen, dass ein Problem oft auf unterschiedliche Arten gelöst werden kann und wie diese unterschiedlichen Ansätze die Effizienz erheblich beeinflussen.

Die Themenmenge wurde bewusst überschaubar gehalten, da für das erste Arbeiten mit Algorithmen zunächst Grundprinzipien wie Iteration und Rekursion verstanden werden müssen. Dieses Verständnis fördert nicht nur die Fähigkeit, logische Abläufe nachzuvollziehen, sondern auch das systematische und strukturierte Vorgehen bei Problemlösungen, eine Kompetenz, die weit über die Informatik hinaus von Bedeutung ist.

Mögliche Projektideen in diesem Lernpfad sind beispielsweise die Analyse eines Textes, bei der die Häufigkeit bestimmter Wörter oder Phrasen ermittelt wird, oder ein Programm zur Organisation von Noten, bei dem eine Liste nach Namen oder nach Noten sortiert werden kann.

Lernpfad 3: Graphen

Der Lernpfad Graphen stellt den anspruchsvollsten der drei Lernpfade dar. Er richtet sich vor allem an Schüler:innen, welche bereits sicher mit dem Programmieren umgehen können und idealerweise erste Vorerfahrungen besitzen, da die Konzepte komplexer sind und ein hohes Maß an logischem Denken erfordern.

Ein großer Vorteil dieses Bereichs ist die starke Anbindung an reale Anwendungen sowie die vielfältigen Möglichkeiten für spätere Projekte. Graphen begegnen uns nicht nur in Navigationssystemen, sondern auch in der Künstlichen Intelligenz, Netzwerkanalysen oder bei der Modellierung komplexer Systeme. Obwohl das Thema anfänglich komplex ist, eröffnet es den Schüler:innen nach dem Verständnis der Grundlagen zahlreiche spannende Anwendungsmöglichkeiten. Insbesondere für diejenigen, die ein späteres Studium oder eine berufliche Tätigkeit in der Informatik anstreben, ist die Auseinandersetzung mit Graphen besonders relevant, da sie zunehmend in modernen Technologien, wie beispielsweise neuronalen Netzen, verwendet werden.

Inhaltlich beschäftigen sich die Schüler:innen zunächst mit den Bestandteilen und dem Aufbau von Graphen. Darauf aufbauend werden Methoden zum Suchen nach bestimmten Knoten behandelt. Schließlich lernen sie einen Algorithmus kennen, der den kürzesten Weg zwischen zwei Knoten berechnet – ein Kernproblem in der Graphentheorie mit direktem Praxisbezug.

Mögliche Projektideen in diesem Lernpfad sind unter anderem die Implementation eines Navigationsalgorithmus, der anhand einer Liste von Städten und deren Entfernungen die schnellste Verbindung zwischen zwei Punkten berechnet. Eine weitere interessante Anwendung wäre die Analyse eines Bildes von einem Mikroskop, bei der Gruppen von Zellen als Knoten interpretiert und miteinander in Beziehung gesetzt werden, um Strukturen oder Muster zu identifizieren.

Einteilung der Gruppen

Nachdem die Schüler:innen sich für einen der Vertiefungspfade entschieden haben, werden sie in Gruppen von jeweils drei Personen eingeteilt. Sollte die Anzahl der Interessierten für einen Lernpfad nicht genau aufgehen, sind auch Gruppen von zwei Personen möglich. Bei vier Interessierten wird beispielsweise auf zwei Zweiergruppen aufgeteilt. Die Zuteilung der Gruppen erfolgt durch die Lehrkraft.

Obwohl Schüler:innen möglicherweise lieber mit ihren Freund:innen zusammenarbeiten würden, erfolgt die Gruppenzuweisung bewusst durch die Lehrkraft. Zum einen soll dadurch

sichergestellt werden, dass die Wahl des Lernpfads auf persönlichem Interesse basiert und nicht primär auf der sozialen Bindung zu Freund:innen. Zum anderen ermöglicht diese Vorgehensweise, dass Schüler:innen mit ähnlichem Leistungsstand zusammenarbeiten. Leistungsstärkere Schüler:innen profitieren so von einem anspruchsvolleren Arbeitsumfeld, in welchem sie sich gegenseitig fördern können, ohne dass die Arbeit einseitig auf wenige Gruppenmitglieder verteilt wird. Schüler:innen mit weniger Erfahrung oder offenen Wissenslücken können in entsprechend angepassten Gruppen an leichteren Aufgaben arbeiten und gezielt ihre Schwächen abbauen, ohne überfordert oder unterfordert zu werden.

Ein zentraler Bestandteil der Gruppenarbeit ist die Ko-Konstruktion: Die Schüler:innen erarbeiten Inhalte nicht nur individuell, sondern konstruieren ihr Wissen aktiv gemeinsam, indem sie Ideen austauschen, Probleme diskutieren und Lösungswege gemeinsam entwickeln. Durch gemeinsame Reflexion und Diskussion werden Verständnislücken identifiziert und behoben, sodass alle Mitglieder einen ähnlichen Wissensstand erreichen.

Dieses Vorgehen fördert nicht nur die Teamfähigkeit und Kommunikationskompetenz, sondern bereitet die Schüler:innen auch auf reale Arbeitsumfelder vor, in denen kooperatives Arbeiten und der Austausch von Wissen zentrale Erfolgsfaktoren sind – wie etwa in Unternehmen oder Informatik-Communities.

Zusammenfassend dienen also die Gruppen in dieser Phase in erster Linie dem gegenseitigen Austausch und der Unterstützung. Während die Schüler:innen ihren gewählten Lernpfad durcharbeiten, sollen sie Ideen diskutieren, Konzepte erklären und Lösungen vergleichen. Dabei lernen sie, Wissen aktiv gemeinsam aufzubauen, wodurch sowohl die individuelle Kompetenz als auch das Verständnis komplexer Inhalte gestärkt wird. Die Schüler:innen finden eine Balance zwischen selbstständigem Arbeiten und kooperativem Lernen, wodurch am Ende alle Mitglieder der Gruppe einen vergleichbaren Wissensstand erreichen und gleichzeitig Teamfähigkeit, Kommunikationskompetenz und Problemlösungsstrategien geschult werden.

Lernen mittels Lernpfad

Das Erarbeiten der Inhalte des Lernpfads ist so gestaltet, dass den Schüler:innen zwei verschiedene Materialien zur Verfügung stehen.

Zum einen gibt es den Lernpfad selbst, der die wichtigsten Themen und Schwerpunkte vorgibt, die bearbeitet werden sollten, um die Grundlagen des jeweiligen Gebiets zu erlernen. Teilweise werden hier kleinere Hilfestellungen eingebaut, um den Einstieg zu erleichtern. Darüber hinaus

werden Quellen wie Tutorials, Erklärungen oder weiterführende Literatur vorgeschlagen, welche die Schüler:innen für das Lernen nutzen können. Es aber ist ausdrücklich erwünscht, dass die Lernenden selbstständig recherchieren, beispielsweise im Internet, in speziellen Foren oder in Bibliotheken. Dies orientiert sich an gängigen Praktiken in der Informatik, bei denen Wissen häufig durch Lesen, Ausprobieren und Reflektieren neuer Ansätze erworben wird.

Ein weiterer Vorteil dieses Vorgehens ist, dass die Schüler:innen in ihrem eigenen Tempo lernen können. Durch den ko-konstruktiven Ansatz der Einheit wird dies zusätzlich verstärkt. Dank der vorherigen Einteilung der Gruppen in ungefähr gleich starke Lernende werden große Unterschiede im Lerntempo innerhalb der Gruppen minimiert, wobei das Lerntempo durch die Zusammenarbeit im Team deutlich verstärkt wird. Die Schüler:innen werden aktiv dazu angeleitet, gemeinsam Themen zu bearbeiten, Gedanken auszutauschen und Erklärungen zu diskutieren. Auf diese Weise lernen sie, verschiedene Herangehensweisen zu vergleichen, voneinander zu profitieren und die Teamdynamik zu stärken. Gleichzeitig wird der Lernprozess durch die Ko-Konstruktion effektiver und spannender gestaltet.

Die grobe Vorgabe der Themen ist besonders wichtig, da es in der Programmierung viele unterschiedliche Ansätze gibt, um Probleme zu lösen. Für Anfänger:innen ist es oft schwierig zu entscheiden, welcher Ansatz am besten geeignet ist, ohne die Hintergründe der einzelnen Funktionen vollständig zu verstehen. Durch die Strukturierung des Lernpfads entsteht ein roter Faden, der Orientierung bietet, aber gleichzeitig genügend Freiraum lässt, um eigene Ansätze zu entwickeln und zu experimentieren.

Die Schüler:innen erhalten zusätzlich ein beispielhaftes Übungsblatt, das die Möglichkeit bietet, die gelernten Inhalte praktisch anzuwenden und zu verinnerlichen. Es dient als Vorschlag, gibt Inspiration für eigene kleine Aufgaben und kann erste Ideen für spätere Projekte liefern. Die Lernenden werden ausdrücklich dazu ermutigt, eigene Probleme zu entwickeln und im Team gemeinsam zu experimentieren. Die Übungsblätter sollen den Lernprozess unterstützen, ersetzen jedoch nicht das aktive Arbeiten an eigenes Problem oder das Rumprobieren mit Verschiedenen Ansätzen im Team.

Im Mittelpunkt steht dabei immer die Ko-Konstruktion: Die Schüler:innen erarbeiten Inhalte aktiv gemeinsam, tauschen Ideen aus, erklären sich gegenseitig Inhalte und vergleichen Lösungswege, sodass dabei jede:r aktiv zum Lernprozess beiträgt. Durch gemeinsame Reflexion und Diskussion werden Verständnislücken geschlossen, sodass alle Mitglieder einen vergleichbaren Wissensstand erreichen.

Dieses Vorgehen fördert nicht nur die Teamfähigkeit und Kommunikationskompetenz, sondern bereitet die Schüler:innen auch auf reale Arbeitsumfelder vor, in denen kooperatives Arbeiten

und der Austausch von Wissen entscheidend sind. Die Lernenden lernen, eine Balance zwischen selbstständigem Arbeiten und gemeinsamer Wissenskonstruktion zu finden, wodurch der Lernprozess effektiver, nachhaltiger und gleichzeitig spannender gestaltet wird.

Übergang zur authentischen Leistung

Bereits in Phase II ist es sinnvoll, die bevorstehende authentische Leistung im Blick zu haben. Ziel ist, dass die Schüler:innen nicht nur neues Wissen erwerben, sondern von Anfang an überlegen, wie sie dieses Wissen später in ihrem eigenen Projekt anwenden können. Auf diese Weise wird der Lernprozess zielgerichtet und praxisnah gestaltet, da die Lernenden von Beginn an Verbindungen zwischen Theorie und Anwendung herstellen.

Die Gruppen sollten sich frühzeitig darüber verständigen, welche Themen oder Teilbereiche für die einzelnen Mitglieder besonders interessant sind. Dabei geht es nicht nur darum, persönliche Interessen zu berücksichtigen, sondern auch Synergien innerhalb der Gruppe zu erkennen. Die Schüler:innen lernen, ihre individuellen Stärken und Vorlieben zu kombinieren, um ein gemeinsames, abgestimmtes Projekt zu entwickeln. Dies fördert die Ko-Konstruktion und stärkt die Fähigkeit, in der Gruppe gemeinsam Entscheidungen zu treffen.

Die bisherigen Übungsaufgaben dienen in diesem Zusammenhang lediglich als Orientierungshilfe oder Beispiele. Sie sollen die Schüler:innen inspirieren und den Einstieg erleichtern, stehen aber nicht im Vordergrund der Projektarbeit. Stattdessen liegt der Fokus darauf, dass die Lernenden eigenständig Ideen entwickeln, Probleme analysieren und Lösungswege planen, die später in der authentischen Leistung umgesetzt werden.

Durch diese Herangehensweise wird in Phase II der Fokus nicht nur auf das Wiederholen beziehungsweise Üben von Themen, sondern auch auf das eigenständiges, praxisnahes Arbeiten gelegt. Schüler:innen lernen, ihr Wissen bewusst zu nutzen, Entscheidungen zu treffen und ihre individuellen Fähigkeiten sinnvoll in die Gruppenarbeit einzubringen. Gleichzeitig werden bereits in dieser Phase Projektmanagement-Fähigkeiten wie Planung, Aufgabenverteilung und Abstimmung innerhalb der Gruppe geübt. Diese Fähigkeiten sind vor allem später in der authentischen Leistung von zentraler Bedeutung.

Phase III: Erstellung der Authentischen Leistung

Nachdem alle Mitglieder einer Gruppe den Lernpfad gemeinsam durchgearbeitet haben, geht es mit der Authentischen Leistung weiter. Die Gruppen sollten damit spätestens bis zum 8. Termin beginnen da sonst die Zeit zu knapp wird. Sollte ein Team früher fertig sein können diese dann direkt den Übergang in die Phase III durchführen.

Aufgabenstellung

Für die Leistung sollen die Schüler:innen ein Programm entwickeln, was ein von ihnen definierte Aufgabe erfüllen soll und in einer Messe vorstellen. Dabei soll die Gruppe klar die Aufgabe, Funktionsweise, Einsatzgebiet und eine konkrete Anwendung für Ihr Projekt definieren, das ihr Programm nach dem Ablauf der Phase III abdecken soll. Dies dient dazu sich einen klaren Plan zu machen, wie ihr Programm nach Fertigstellung aussehen soll, ähnlich zu einem Auftrag das ein Unternehmen an einen Dienstleister stellt, um das Szenario wieder realitätsnah zu gestellten. Dabei soll die Gruppe auch ein Zwischenergebnis formulieren, welches aufzeigt, was ihr Programm nach einem zuvor von der Lehrkraft definiert Zeitpunktes (vorzugsweise an dem 10. Termin), also einem Art Meilenstein, können muss. Dadurch entsteht automatisch ein kleiner Zeitplan, der die Gruppe dabei unterstützen soll, einerseits ihre Zeit passend einzuteilen und überprüfen soll, ob ihrer Idee nicht zu ambitioniert und in der Zeit nicht umsetzbar ist. Diese Ziele und Anforderungen werden dann auch schriftlich auf einem Arbeitsblatt festgehalten.

Dabei muss die Gruppe ihre Idee auch der Lehrkraft pitchen. Dadurch kann die Lehrkraft, bei Ideen, die komplizierte Funktionen wie Bild oder Textverarbeitung erfordert mit kleinen Hilfsprogrammen, welche sie der Gruppe zu Verfügung stellt, ihnen zu helfen. Hierbei muss sich die Gruppe nicht mit Funktionen befassen, die zu kompliziert für den aktuellen Wissensstands wären, trotzdem kann sie ihr Projekt umsetzten und das mittels des Hilfsprogramms in einem spannenderen komplexeren Rahmen, ohne zu extremer Schwierigkeit.

Bei zu hohen Ambitionen des Teams kann die Lehrkraft auch Vorschläge machen, wie man die Aufgabe abändern könnte, damit das Projekt machbarer wird. Dies nimmt zwar ein wenig die Selbstbestimmung sorgt aber dafür das den Schüler:innen sehr viel Frustration erspart bleibt. Dies ist vor allem deshalb wichtig, weil die meisten der Schüler:innen wahrscheinlich ihr

erstes großen Programm schreiben und die Komplexität eines solchen noch nicht richtig einschätzen können.

Technischer Hintergrund

Damit das Arbeiten in der Gruppe reibungslos funktioniert, musst die Lehrkraft den Gruppe eine Oberfläche bieten, sodass die Mitglieder einer Gruppe lückenlos zusammen an einem Programm arbeiten können. Dies lösen wir mittels eines „git-Repositories“. Dadurch erreichen wir das die Mitglieder ihren Teil in den Code einfügen können und durch regelmäßiges Synchronisieren wird dabei immer der Teil jedes Mitgliedes in den gesamten Code hinzugefügt. Dabei können die Gruppen nicht nur nahtlos arbeiten, sondern könnten nach Einrichtung auch von zuhause oder neben dem Unterricht problemlos an ihrem Projekt werkeln. Ein weiterer Vorteil für die Lehrkraft ist das durch git immer die einzelnen Versionen gespeichert werden. Damit kann die Lehrkraft während und nach der Einheit genau nachvollziehen, wie die Schüler:innen an ihre Projekte herangegangen sind. Außerdem sieht sie auch genau, wer was beigetragen hat und ob zum Beispiel das Team ausgeglichen oder eher unausgeglichen ist. Dies ermöglicht der Lehrkraft für die aktuelle oder auch für zukünftige Einheiten kleine Änderungen vorzunehmen um somit den Lern Effekt der Schüler:innen zu verstärken. Jenes ist auch für die Schüler:innen ein Blick in die Zukunft, da git ein sehr häufiges Tool in der Informatik ist, welches Softwareentwickler täglich benutzen. Hier kommt auch wieder das Programm VS-Code ins Spiel, welches mittels eines Plugins das Bearbeiten dieses Repositories ganz einfach per Knopfdruck ermöglicht, ohne dabei Kenntnisse in der Komandozeile zu haben.

Meilenstein

Am festgelegten Meilensteintermin (10. Termin) präsentieren die Gruppen ihren bisherigen Fortschritt. Zunächst stellt die Gruppenleitung das Projekt in seiner aktuellen Form vor, indem sie die Aufgabenstellung, die grundlegende Funktion sowie die bisherigen Ideen und Ziele erläutert. Anschließend präsentieren die einzelnen Mitglieder über ihre Arbeitsschritte. Dabei gehen sie kurz darauf ein, welche Funktionen sie bereits umgesetzt haben, auf welche Probleme sie gestoßen sind und welche Lösungsstrategien sie angewandt haben. Außerdem reflektieren sie ihre bisherigen Gedankengänge und zeigen auf, wie diese den Projektverlauf beeinflusst haben.

Im Anschluss erklären sie, welche Arbeitsschritte noch geplant sind, wie sie die weitere Umsetzung strukturieren möchten und welche Schwierigkeiten sie dabei erwarten. Zur Unterstützung können sie entweder direkt mit ihrem Code arbeiten oder eine kleine Präsentation vorbereiten, um ihre Ergebnisse anschaulich darzustellen.

Dieser Meilensteinvortrag orientiert sich bewusst an realen Arbeitsabläufen in Unternehmen, in denen Teams regelmäßig ihre Fortschritte vorstellen und Feedback einholen. Die Schüler:innen lernen dadurch nicht nur, ihre Arbeit strukturiert aufzubereiten, sondern auch ihre Ergebnisse vor anderen verständlich und nachvollziehbar zu kommunizieren.

Darüber hinaus profitieren die Lernenden von der gegenseitigen Präsentation: Sie erhalten Einblicke in alternative Lösungsansätze anderer Gruppen, können voneinander lernen und Anregungen für ihre eigenen Projekte mitnehmen. Besonders wertvoll ist auch die Möglichkeit, bei ähnlichen Herausforderungen ins Gespräch zu kommen und sich gegenseitig zu unterstützen. Auf diese Weise wird der Meilenstein zu einem wichtigen Lernmoment, der sowohl die fachlichen Kompetenzen als auch die kooperative Zusammenarbeit stärkt.

Fertigstellung der Authentischen Leistung

Nach der 12-wöchigen Unterrichtseinheit findet eine Abschlussveranstaltung in Form einer kleinen Messe statt. Jedes Team erhält dabei einen eigenen Stand, an dem es sein Projekt präsentiert. Zur Verfügung steht den Gruppen mindestens ein Bildschirm, auf dem sie ihr Programm vorführen können. Ergänzend gestalten die Schüler:innen weitere Materialien wie Plakate, Infotafeln oder Handouts, die ihr Projekt und die zentralen Ideen anschaulich erläutern.

Die Veranstaltung verfolgt mehrere Ziele: Einerseits können die Gäste, wie Mitschüler:innen, Lehrkräfte, Eltern oder externe Besucher:innen, die entwickelten Programme ausprobieren und sich die Funktionsweise erklären lassen. Andererseits lernen die Schüler:innen, ihre Ergebnisse so aufzubereiten, dass sie auch für Außenstehende verständlich und nachvollziehbar sind. Besonders das abstrakte und zugleich präzise Erklären von Code und Gedankengängen ist eine wichtige Kompetenz, die im Unterricht nur schwer, aber in solchen Praxissituationen sehr authentisch geübt werden kann.

Darüber hinaus kann die Messe, je nach regionalem Umfeld, auch durch die Teilnahme von Vertreter:innen aus Unternehmen oder Hochschulen bereichert werden. Für die Schüler:innen ergibt sich dadurch die Gelegenheit, direkt mit Fachleuten ins Gespräch zu kommen, Einblicke in die Berufspraxis zu gewinnen und Anregungen für ihre eigene Zukunft zu sammeln.

Umgekehrt erhalten die Unternehmen die Chance, potenzielle Nachwuchskräfte kennenzulernen, Kontakte zu knüpfen und für beispielsweise Ausbildungsplätze oder duale Studienangebote zu werben.

Nicht zuletzt trägt eine solche Veranstaltung auch dazu bei, das Fach Informatik sichtbarer zu machen. Sie zeigt, wie Schule praxisnah arbeiten kann, wie junge Menschen eigene Ideen umsetzen und welche Bedeutung digitale Kompetenzen für die Zukunft haben.

Benotung

Bei der Benotung ist es wichtig das wir nicht den gesamten Fokus auf das Programm legen, welches als Ergebnis am Ende rauskommt. In dieser Einheit ist das selbstbestimmte Lernen der Programmiersprache und auch die Kooperation im Team viel wichtiger. Dadurch entsteht eine Benotung wie folgt:

20% Die Note im Abschlussquiz in Phase I

20% Projekt mit Präsentation

20% Eindruck und Mitarbeit in Phase I und II

40% Arbeiten im Team in Phase II und III

Auf den ersten Blick sieht diese Benotung recht komplex für die kürze der Einheit aus, kommt aber mit vielen Vorteilen. Zum einen bewerten wir die Ergebnisse des Quizzes in Phase I und das Projekt in Phase III, dies ist der Nähe zur Realität geschuldet, welche wir während der Einheit bewahren wollen und in einem Team oder Unternehmen später ist das Ergebnis elementar.

Darüber hinaus legen wir ein stärkeres Gewicht auf das konzentrierte Mitarbeiten während der Einheit sowie auf die Teamarbeit, die den zentralen Bestandteil dieser Lerneinheit bildet. Dabei ist jedoch zu berücksichtigen, dass auch der zeitliche Rahmen eine Rolle spielt. Da die Teamarbeit weniger als ein Drittel der Gesamtdauer ausmacht, wäre es nicht fair, ihr den überwiegenden Teil der Bewertung zuzuschreiben. Aus diesem Grund liegt ihr Anteil bei 40 % der Gesamtnote

Reflektion der Einheit

Im Anschluss an die Einheit wir es eine Stunde geben in der wir die Einheit reflektieren. Dabei werden die Schüler:innen gebeten sich im ersten Schritt selbst zu reflektieren und auf Feedback über die Einheit zu geben. Im Anschluss sollen sich die Teams in Selbstreflexion üben. Dabei liegt ein großer Fokus auf der Kommunikation und Teilung von Aufgaben. Dies ist einer der wichtigsten Fähigkeiten in der gesamten Informatik reibungslos als Team arbeiten zu können und Aufgaben in kleine Teile aufzuteilen, von verschieden Teams bearbeiten zu lassen und wieder zu einer gesamten Einheit lückenlos zusammenzuführen. Dies ist auch einer Hauptthemen dieser Einheit neben dem rein fachlichen Bereich.

Mit dieser Reflektion schließen wir die Unterrichtseinheit ab.

Bibliografie

b001. (2024, 7. Juni) *Shortest Path Algorithms Explained (Dijkstra's & Bellman-Ford).*🔗

datacamp.com. (2025. 16. Januar). *Objektorientiertes Programmieren in Python (OOP): Tutorial.* 🔗

education-wiki.com. (o.J.). *Sortieralgorithmen in Python - Top 6 Sortieralgorithmen in Python.* 🔗

geeksforgeeks.org. (2025, 23. Juli) *Depth First Search or DFS on Directed Graph.*🔗

Programmieren lernen. (2022. 16. Februar). *Objektorientierung in 10 Minuten.* 🔗

National University of Singapure, visualgo.net. (o. J.). *sorting.* 🔗

The Morpheus Tutorials. (2020, 5. April*) Algorithmen und Datenstrukturen – leicht erklärt* 🔗

Universität Freiburg, Institut für Informatik. (2012). *Graphen – Einführung.* 🔗

w3schools.com. (o.J.). *Python OOP.* 🔗