Semester-Programm “Einführung Programmieren”

Überblick der Themen

1. **Allgemeine Motivation und Einführung ins Programmieren**
   - 1-2 Beispiele mit Bezug zur Lebenswelt und Vorwissen von SuS, einige Algorithmen und Beispiele zu logischem Denken
   - Vordefinierte Grafik-Befehle mit Turtle (nur ein Parameter): `forward()`, `right()`
   - Einfache Schleifen mit `for i in range(...)`
2. **Datentypen und Syntax**
   - Variablen und Zuweisung vs. Mathematik
   - Arbeiten mit einfachen mathematischen Operatoren
   - Arbeiten mit Text
   - Debugging-Übungen (syntax vs. runtime vs. semantic errors), typische Fehler
3. **Verzweigungen** (`if...elif...else`) & `while`, elementare logische Operatoren (or, and, xor) und Flussdiagramme
4. **Befehle / Funktionen** (`def`)
   - Ohne Parameter, ohne `return`
   - Mit Parameter, ohne `return`
   - Mit Parameter, mit `return`
5. **Listen**
   - Koordinatengrafik, Zufallszahlen, `random`-Library (→ Turtle),
   - Schleifen / Traversieren (`for item in mylist`)

Inhaltsverzeichnis

1 DL 1: Allgemeine Motivation und Einstieg ins Programmieren

DL 1

Thema

Die Lektion soll den Einstieg in die, für die meisten SuS noch neue Welt des Programmierens machen. Dabei werden zuerst das Thema "Programmieren" im Allgemeinen und danach Python als spezifische Programmiersprache für diesen Semester-Kurs vorgestellt. Abschliessend führen einige einfache grafische Beispiele mit der `turtle`-Library die SuS an die Programmierung mit Python heran.

Operationalisierte Lernziele

- ☐ Die SuS verstehen, dass viele alltägliche Abläufe durch Algorithmen, d.h. automatisierte und logische Abläufe, modelliert und beeinflusst werden.
- ☐ Die SuS können einige Beispiele aus dem Alltag nennen, wo Algorithmen eine Rolle spielen.
- ☐ Die SuS verstehen grundsätzliche Elemente von Flussdiagrammen und können logische Prozesse aus der ihnen bekannten Alltagswelt damit modellieren.
- ☐ Die SuS haben Python sowie eine IDE installiert, mit welcher sie Python-Programme abspeichern und ausführen können.
- ☐ Die SuS können elementare Befehle mit der `turtle`-Library ausführen, um damit einfache Zeichnungen zu erstellen
- ☐ Die SuS können einfache Formen mit einer `for i in range(start, end, step)`-Schleife zeichnen, beispielsweise ein Quadrat oder ein Sechs-Eck.

Grober Lektionsablauf

Lektion 1/2: Programmieren & Algorithmen

Erste Hälfte der Lektion: Die SuS werden an das Thema Programmieren herangeführt und lernen dessen Relevanz anhand einiger alltagsnaher Beispiele kennen: Messgeräte in Spitäler (unterliegende Idee: `if...elif...else`), Sensoren in Autos (z.B. `while`), etc. Dabei lernen die SuS, die unterliegenden Algorithmen als Flussdiagramme dazustellen. Zweite Hälfte: Die SuS lösen eigenhändig Übungen und wenden ihr neu gewonnenes Wissen zu Flussdiagrammen auf die Übungen an (Abgabe auf Moodle).

Lektion 2/2: Programmieren mit Python und `turtle`

Die SuS lernen Python kennen und installieren ein lokales IDE. Danach führen sie, geleitet durch die LP, einige Übungen durch, um einfache Formen mit der Schildkröte zu zeichnen. Dabei lernen Sie die erste einfache Schleife (`for i in range(start, end, step)`) kennen.

Mögliche Schwierigkeiten & geeignete Massnahmen

1. Abstraktes Denken zu Flussdiagrammen ist komplex und herausfordernd. Der wäre es hilfreich, bereits zu Beginn des Unterrichts ein Nachschlagewerk zur Verfügung zu stellen, welches die SuS dabei unterstützt, die Übungen zu lösen, indem sie bei Bedarf nachlesen können.
2. BYOD als Herausforderung: Möglicherweise gestaltet sich der Installationsprozess komplexer und langwieriger als gedacht. Daher sollte die Installation des IDE nach Möglichkeit bereits im Voraus als Hausaufgabe (mittels Schritt-für-Schritt-Dokumentation) ausgeführt worden sein.
3. Technische Herausforderungen beim Ausführen von Code: Gerade in der ersten Lektion brauchen einige SuS noch Unterstützung beim Ausführen und Debuggen von Code ("wo ist der Run-Knopf?"). Mögliche Lösungsansätze könnten sein, genügend Zeit für die Übungen einzuplanen (s. Punkt 2), bzw. den Einstieg nach Möglichkeit in Halbklassen zu machen.

2 DL 2: Datentypen und Syntax (Variablen & Zahlen)

DL 2

Thema

In dieser Doppellektion geht es darum, die SuS an die Grundlagen von Datentypen sowie einige grundlegende syntaktische Elemente in Python heranzuführen. Dabei soll, in der ersten der beiden dafür vorgesehenen Doppellektionen, der Fokus auf mathematische Operationen gelegt werden, wodurch die SuS hoffentlich auch einen Bezug zu ihrem Vorwissen herstellen können.

Operationalisierte Lernziele

- ☐ Die SuS können das Gleichzeichen verwenden, um Variablen zu erstellen. Sie verstehen zudem den Unterschied zwischen dem Gleichzeichen in Python und dem Gleichzeichen in der Mathematik.
- ☐ Die SuS können einfache Rechnungen machen, indem Sie Variablen erstellen und einfache mathematische Operatoren verwenden.
- ☐ Die SuS können den Inhalt einer Variable `x` mit `print(x)` ausgeben.
- ☐ Die SuS können den Modulo-Operator (`%`) in sinnvoller Weise verwenden, beispielsweise um alle geraden Zahlen in einer Schleife auszugeben (s. DL 1 für einfache Schleifen).
- ☐ Die SuS können Speicherinhalte verändern, indem Sie *compound operators* in einer einfachen Schleife verwenden, also beispielsweise `+=`, `-=`, `*=` oder `/=`.
- ☐ Die SuS können Funktionen, um die Wurzel einer Zahl (z.B. `sqrt(x)`) oder deren Quadrat (z.B. `x**2`) sinnvoll verwenden und das Resultat in einer Variable speichern sowie ausgeben.
- ☐ Die SuS können einfache Formen mit Pythagoras zeichnen (beispielsweise ein schräges Dach in einem Haus)
- ☐ Die SuS können Probleme aus der Mathematik wie etwa polynomiale Gleichungen in einen Python-Code transformieren und diese damit lösen.

Grober Lektionsablauf

Lektion 1/2: Einführung und Übungen

Einfache Beispiele werden aufgezeigt, um das Interesser für mathematische Operatoren mit einigen alltagsnahen Beispielen einzuführen. Beispielsweise könnte aufgezeigt werden, wie aus einer Liste von Grössen diejenigen ausgegeben werden können, deren Körpergrösse gerade ist. Da die Doppellektion eher voll beladen ist, wird diese Sequenz jedoch relativ kurz gehalten, um genug Zeit für die Übungen zu lassen. Die Übungen sind, wie immer, im Skript, und die Erklärungen zu den neuen Konzepten, welche mit dem einführenden Beispiel vermittelt wurden, können dort bei Bedarf nochmals nachgelesen werden.

Lektion 2/2: Übungen

Lektion 2 widmet sich hautpsächlich den Übungen. Am Schluss wird, je nach dem welche Schwierigkeiten bei den SuS während den Übungen beobachtet wurde, nochmals auf einige herausfordernde Aufgaben eingegangen, und danach wird eine Ergebnissicherung (neue Konzepte) durchgeführt.

Mögliche Schwierigkeiten & geeignete Massnahmen

1. Die SuS könnten verwirrt sein, weshalb mit dem Gleichzeichen nun etwas anderes gemeint ist als in der Mathematik (Zuweisung statt Gleichheit). Dies könnte durch klare grafische Darstellungen gelöst werden, etwa wie folgt: `a` = `10+3` Die Erklärung dazu könnte lauten: "Wir werten `10+3 = 13` aus und speichern den Wert `13` in der Variable `a`".
2. Die Lektion könnte zu kurz sein für alle Inhalte. Allenfalls müsste nach einer ersten Durchführung entschieden werden, ob Teile der Lektion in DL 3 durchgeführt werden, welche aktuell daher etwas weniger umfangreich geplant ist.

3 DL 3: Datentypen und Syntax (Text & Debugging)

DL 3

Thema

In dieser zweiten, den Datentypen und Grundlagen sowie Syntax gewidmeten Doppellektion geht es darum, elementare Text-Operationen kennezulernen und diese sinnvoll zu verwenden. Ebenfalls soll sichergestellt werden, dass die SuS geläufige Laufzeit-Probleme aufgrund von Datentypen selbstständig entdecken und beheben können.

Operationalisierte Lernziele

- ☐ Die SuS wissen, was eine Zeichenkette in Python ist und können eine solche selbstständig erstellen.
- ☐ Die SuS können Zeichenketten mit `"..." + "..."` verketten.
- ☐ Die SuS können Zeichenketten mit `"..." * [zahl]` beliebig viele Male wiederholen.
- ☐ Die SuS können mithilfe der genannten Befehle, Spezialzeichen (z.B. `\n`) sowie `print()` einfache ASCII-Zeichnungen machen, wie etwa einen Pfeil.
- ☐ Die SuS können Zeichenketten sowohl innerhalb einfacher wie doppelter Klammern (`'` oder `"`) schreiben und verstehen wozu dies dient (*escape character*, etwa in folgendem Befehl: `print('Meine Freunde nennen mich "Mr. X"')`.
- ☐ Die SuS können geläufige Laufzeit-Probleme wie etwa die nicht funktionalen Ausdrück `print("1+2")` oder `print(Hallo Welt)` selbstständig als solche erkennen und beheben.
- ☐ Die SuS könenn die Funktion `input("Fragetext")` verwenden, um einen Wert (Text oder Zahl) während der Laufzeit zu generieren. Diesen können die SuS in einer Variable abspeichern und weiterverwenden.

Grober Lektionsablauf

Ähnlich wie die vorherige Doppellektion beginnt auch diese Doppellektion mit einigen alltagsnahen Beispielen, die direkt im Code-Editor vorgezeigt und ausgeführt werden. Beispielsweise könnte mit `input` nach einem Namen gefragt werden, der dann in der Konsole 20 mal als ASCII-Zeichnung ausgegeben wird. Nach einer relativ kurzen Einführungssequenz wird der Rest der Doppellektion den Übungen sowie einer Ergebnissicherung gewidmet, wobei letztere insbesondere nochmals Debugging-relavante Aufgaben aufgreift (welche im Skript an letzter Stelle dieser Doppellektion vorkommen).

Mögliche Schwierigkeiten & geeignete Massnahmen

1. Die SuS könnten unter umständen verwirrt sein, weshalb `turtle`-Grafiken in einem separaten Fenster gezeichnet werden, währenddem der `print()`-Befehl in der Konsole ausgegeben wird. Darauf sollte in der Einführung eingegangen werden.
2. Die SuS könnten Mühe haben mit dem Abstraktionsgrad von Zeichenketten (Wiederholungen etc.) und durch die Syntax verwirrt sein (beispielsweise die Möglichkeit, Zeichenketten mit einem einfachen oder doppelten Anführungszeichen zu schreiben). Diesen Aspekten sollte daher sowohl während der Präsentation wie auch im Skript besondere Aufmerksamkeit gewidmet werden.

4 DL 4: Verzweigungen mit `if...elif...else`

DL 4

Thema

Die SuS lernen das Programmieren mit konditionaler Logik kennen, indem Sie mit den Befehlen `if`, `elif` und `else` vertrautgemacht werden. Dabei wird nochmals das in DL 1 angeeignete Wissen zu Flussdiagrammen aufgefrischt, um die neuen Konzepte grafisch (im Sinne der *dual-code*-Theorie) zu unterstützen und aufzuwerten.

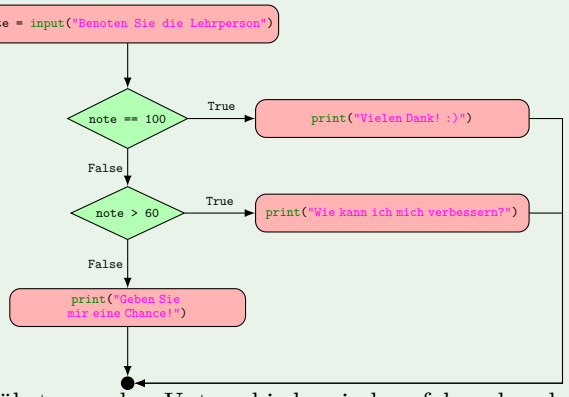
Operationalisierte Lernziele

- ☐ Die SuS können konditionale Logik mittels den Befehlen `if`, `elif` und `else` anwenden, um Befehle nur unter gewissen Bedingungen auszuführen
- ☐ Die SuS verstehen die Bedeutung der Einrückung in Python
- ☐ Die SuS können einen gegebenen Code als Flussdiagramm veranschaulichen
- ☐ Die SuS können erklären, weshalb ein bestimmter Code (nicht) ausgeführt wird.
- ☐ Die SuS können logische Tests formulieren, indem Sie die Operatoren `!=`, `<`, `<=`, `>=`, `>` und `==` korrekt auf Zahlen oder Text anwenden.
- ☐ Die SuS können logische Tests korrekt negieren, indem sie den Operatoren `not` verwenden.
- ☐ Die SuS können zwischen der Negation und dem Gegenteil einer (logischen) Aussage unterscheiden, indem sie eigene, alltagsnahe Beispiele erstellen.

Grober Lektionsablauf

Ähnlich wie die vorherigen Doppellektionen beginnt auch diese Doppellektion mit einigen alltagsnahen Beispielen, die direkt im Code-Editor vorgezeigt und ausgeführt werden. Dabei wird parallel zum Code ebenfalls eine Visualisierung des Codes als Flussdiagramm gezeigt, wie folgt:

```
1 note = input("Benoten Sie die Lehrperson
      (0-100).")
2 if note == 100:
3     print("Vielen Dank! :)")
4 elif note > 60:
5     print("Wie kann ich mich verbessern?")
6 else:
7     print("Geben Sie mir eine Chance!")
```



Das Beispiel würde aber Schritt für Schritt eingeführt, um den Unterschied zwischen folgenden drei Dingen klar aufzuzeigen: 1. `if` *ohne* `elif` und *ohne* `else` 2. `if` *ohne* `elif` aber *mit* `else` 3. `if` *mit* `elif` und *mit* `else`. Ebenfalls können weitere alltagsnahe Beispiele zumindest konzeptuell aufgezeigt werden, um aufzuzeigen, wozu die konditionale Logik dienen kann (medizinische Bereiche, selbstfahrende Autos etc.). Danach üben die SuS das neu gewonnene Wissen, indem Sie es auf Zeichenketten, Texte und mit `turtle` anwenden. Am Schluss findet wie immer eine Ergebnissicherung statt.

Mögliche Schwierigkeiten & geeignete Massnahmen

1. Die SuS finden es schwierig zu erkennen, ob Code nach einem `elif` oder `else` noch ausgeführt wird, da sie nicht verstehen, dass diese Blöcke nur dann ausgeführt werden, wenn vorher das `if False` war. Im Unterricht sollten daher einige Beispiele aufgezeigt werden, die dies veranschaulichen
2. Die Negation einer Aussage ist nicht dasselbe wie deren Gegenteil. Dies sollte anhand von alltagsnahen Beisielen (z.B. der Aussage “der Schrank ist voll”) aufgezeigt werden.

5 DL 5: Verzweigungen mit `while`

DL 5

Thema

In dieser Doppellektion soll die `while`-Schleife eingeführt werden, die gewissermassen die bisher gewonnenen Kenntnisse zu Schleifen sowie konditionaler Logik kombiniert. Dabei wird das Augenmerk wieder vermehrt auf "unterhaltsame", grafische Übungen gelegt, wie etwa dem Zeichnen von Spiralen mit `turtle` und mit dem `while`-Befehl. Als gegenüber zur "kopfgesteuerten" `while`-Schleife wird der Befehl `break` eingeführt, sozusagen als "fussgesteuerte" Schleife. Die Visualisierung von `while`-Schleifen als Flussdiagramm hilft, den Code besser zu verstehen.

Operationalisierte Lernziele

- ☐ Die SuS können `while`-Schleifen, bzw. `repeat`-Befehle verwenden, um Schleifen zu schreiben, die unter bestimmten Konditionen enden
- ☐ Die SuS können erklären, weshalb ein Code der nach einem ausgeführten `repeat`-Ausdruck steht, nicht mehr ausgeführt wird
- ☐ Die SuS können für ein einfaches, gegebenes Beispiel bestimmen, wie häufig eine bestimmte `while`-Schleife ausgeführt wird.
- ☐ Die SuS können die geeignetste Schleifenart auswählen, um ein bestimmtes Problem zu lösen (kopf- oder fussgesteuerter Abbruch, bzw. `for`-Schleife mit vorgegebener Anzahl Wiederholungen)
- ☐ Die SuS können Flussdiagramme um die neu erlernten Schleifentypen erweitern.

Grober Lektionsablauf

Als Einführung in die Lektion werden einige Beispiele ausgewählt, die die Schwierigkeiten, bzw. Komplexitäten gewisser Aufgaben *ohne* `while`-Schleife aufzeigen. Beispielsweise könnte dies das Zeichnen einer Schleife sein, *solange* die Seitenlänge unter, bzw. über einem gewissen Wert liegt. Das gegebene Problem soll einem als `for`-Schleife ohne Abbruch (mit berechneter Anzahl Wiederholungen) gelöst werden, einmal mit `while` und einmal mit `repeat`. Jede Lösung wird als Flussdiagramm aufgezeichnet. Die SuS sollen sich überlegen, welche der Lösungen am geeignetsten ist für ihr Problen. Wie in den Vorgängerlektinonen wird danach der Grossteil der DL den Übungen gewidmet.

Mögliche Schwierigkeiten & geeignete Massnahmen

1. Die SuS finden es schwierig, zwischen einem `break`- und einem `while`-Konstrukt zu unterscheiden, bzw. die richtige Lösung für die richtige Situation auszuwählen. Zur Linderung dieses Problems könnte beitragen, dass die SuS zuerst eine Reihe von einfachen Beispielen von der einen in die andere Variante umgeschrieben werden müssen. Dabei sollen sie sich stets überlegen, was sie bevorzugen. Ihre Antworten fliessen in die Diskussion vor der Ergebnissicherung ein.
2. Die SuS verstehen nicht, wie sich eine Variable innerhalb einer `while`-Schleife verändert. Dazu sollen sie spezielle Tabellen anfertigen, welche die Entwicklung der Variable in jedem Durchlauf der Schleife veranschaulichen.

6 DL 6: Befehle / Funktionen ohne Parameter, ohne `return`

DL 6

Thema

Diese Doppellektion soll als Einführung in die modulare Programmierung dienen. Dabei soll klargestellt werden, dass wir für das schreiben eines komplexen Programms zunächst kleinere "Bausteine"benötigen, die zum Lösen von Teilproblemen nötig sind. Die verschiedenen Programme, die sie bisher geschrieben haben, sollen somit für den zukünftigen Gebrauch als Befehle festgehalten werden. Für die Einführung verwenden wir weder Parameter noch das return statement.

Operationalisierte Lernziele

- ☐ Die SuS können mithilfe der Turtle-Library einen eigenen Befehl ohne Parameter schreiben und aufrufen, der eine einfache geometrische Form (bspw. Dreieck, Quadrat usw.) erzeugt.
- ☐ Die SuS können anhand eines vorgegeben Befehls herleiten, was nachdem aufrufen des Befehls auf dem Bildschirm zu sehen ist. Der vorgegebene Befehl kann folgenden Aufbau haben: Sequentielle Folge von einfachen Turtle-Befehlen, wie bspw. `forward()`, `left()`, `right()`. While-Schleifen oder Verzweigungen, die einfache Turtel-Befehle beinhalten.

Grober Lektionsablauf

Lektion 1/2: `def` Statement

Zu Beginn das Konzept der Schleifen in Erinnerung rufen (als Automatisierung eines sich wiederholenden Vorgangs) Übertragung dieses Konzeptes auf den Aufbau des Codes. ("Bündelung von vorprogrammierten Befehlen durch die Nutzung des `def` Statements, um Wiederholungen zu vermeiden.) Anschliessend sollen die SuS bereits verwendete Befehlsfolgen von Turtle-Befehlen aus den vorherigen Lektionen in einen neuen, eigenen Befehl zusammenfassen und aufrufen.

Lektion 2/2: Verdichtung

Die SuS sollen zunächst neue Befehle schreiben, die sich von den bisher im Unterricht behandelten Programmen unterscheiden. Prinzipiell sollen sie sich zunächst mit einfach geometrischen Formen auseinandersetzen. Im Anschluss an diese Übung erhalten die SuS verscheiden vorgegebene Befehle und sollen diese auf ihren Zweck untersuchen. Ihre Vermutungen können sie durch das ausführen des Befehls überprüfen.

Mögliche Schwierigkeiten & geeignete Massnahmen

1. Es könnte hier leicht zu Problemen mit der Verwendung von Parametern kommen. Durch den Einsatz von vorprogrammierten Turtle-Befehlen wissen sie bereits, dass man Parameter in die Klammern eines Befehls eingeben kann. Daher könnte es für sie naheliegend sein, dies auch beim aufrufen eines eigenen Befehls zu versuchen. Die LP könnte bereits andeuten, dass das verwenden von Parametern in unserem Fall noch nicht möglich ist/ klarstellen, dass bei den eigenen Befehlen die klammer leer sein muss.
2. Ein weitere Schwierigkeit wird die Syntax sein (fortlaufendes Problem bei der Einführung der Programmierung). In Python muss die Verschachtelung des Codes strikt eingehalten werden. D.h nach dem Doppelpunkt muss die nächste Zeile einen zusätzlichen Abstand enthalten. IDE's, wie Thonny, machen diesen Abstand in der regel automatisch.
3. Zudem könnte der Unterscheide zwischen der Definition und dem Aufrufen des Befehls ein Problem darstellen. Bspw könnte ein SuS versuchen, den Befehl aufzurufen, bevor er ihn definiert hat/ das Statement `def` zu nutzen, um den Befehl aufzurufen/ nach dem Aufrufen des Befehls einen Doppelpunkt zu platzieren.

7 DL 7: Befehle / Funktionen mit Parameter, ohne `return`

DL 7

Thema

Die bereits erworbenen Erkenntnisse zu Befehlen soll nun noch erweitert werden, indem man den nutzen von Parametern einführt. Zunächst sollen ein-Parameter Befehle und anschliessend mehr-Parameter Befehle behandelt werden. In dieser Doppellektion gehen wir noch nicht auf das `return` Statement ein.

Operationalisierte Lernziele

- ☐ Die SuS können in eigenen Worten beschreiben, was ein Parameter ist.
- ☐ Die SuS können einen eigenen Befehl mit einem Parameter schreiben und aufrufen, der eine einfache Tätigkeit bewältigt. Als einfache Tätigkeiten zählen: Das Zeichnen einer einfachen geometrischen Form, wie bespw. ein Dreieck oder Viereck, mithilfe von Turtle-Befehlen/ die Zuweisung eines Wertes zu einer Variable, wobei mindestens eine mathematische Berechnung vorkommt.
- ☐ ...

Grober Lektionsablauf

Lektion 1/2: ...

Einführung: Gewisse Programme unterscheiden sich nur in ein bis zwei kleinen Punkt voneinander. Bspw Das Zeichen eines regulären n-Ecks unterscheidet sich in der Ausführung nur vom verwendeten Winkel und der Anzahl Schleifenwiederholungen.

Lektion 2/2: ...

...

Mögliche Schwierigkeiten & geeignete Massnahmen

1. ...

8 DL 8: Befehle / Funktionen mit Parameter, mit `return`

DL 8

Thema

...

Operationalisierte Lernziele

- ☐ ...
- ☐ ...
- ☐ ...

Grober Lektionsablauf

Lektion 1/2: ...

...

Lektion 2/2: ...

...

Mögliche Schwierigkeiten & geeignete Massnahmen

1. ...

9 DL 9: Koordinaten & Zufallszahlen

DL 9

Thema

...

Operationalisierte Lernziele

- ☐ ...
- ☐ ...
- ☐ ...

Grober Lektionsablauf

Lektion 1/2: ...

...

Lektion 2/2: ...

...

Mögliche Schwierigkeiten & geeignete Massnahmen

1. ...

10 DL 10: Listen

DL 10

Thema

...

Operationalisierte Lernziele

- ☐ ...
- ☐ ...
- ☐ ...

Grober Lektionsablauf

Lektion 1/2: ...

...

Lektion 2/2: ...

...

Mögliche Schwierigkeiten & geeignete Massnahmen

1. ...

11 Begleitdokument für Lehrpersonen

- Einführung in den Semesterplan
- Kontext (Voraussetzungen, Stufe, Empfehlung zur Einbettung und zur Fortsetzung)
- Mögliche Schwierigkeiten der SuS und Lösungsvorschläge (für Gesamt-Programm)