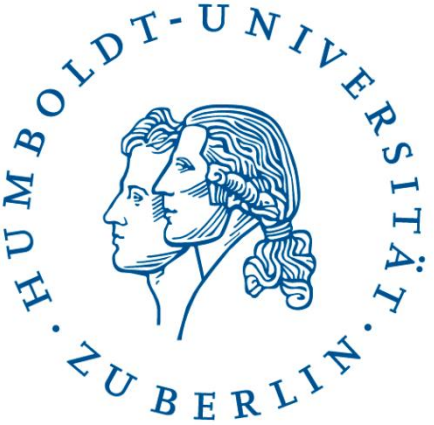
Bericht zum Unterrichtspraktikum

im Fach Informatik



HUMBOLDT-UNIVERSITÄT ZU BERLIN

MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT

INSTITUT FÜR INFORMATIK

Wintersemester 2014/2015

Dozent: Dr. Nguyen-Thinh Le

Verfasserin: Mina Ghomi

Inhaltsverzeichnis

1. Einleitung

Im Rahmen der Veranstaltung „Schulpraktische Studien“ in meinem Zweitfach Informatik absolvierte ich ein Unterrichtspraktikum an der A.-F.-Schule (Abk. AFS). Das Unterrichtspraktikum begann mit 30 Hospitationsstunden semesterbegleitend seit dem 28.11.2014. Der Zeitraum für die Unterrichtsstunden mit eigener Tätigkeit erstreckte sich vom 23.02.2015 bis zum 20.03.2015. Davon waren drei 90 minütige Unterrichtsstunden vollständig von mir und drei gemeinsam mit der Lehrkraft durchgeführt. Meine Mentorin ist verantwortlich für den Fachbereich Informatik.

1.1 Profil der Schule

Die AFS ist ein Oberstufenzentrum für Sozialwesen in Berlin. Die Schule hat ca. 1300 Schüler[1] und 100 Lehrer. Die AFS bietet ihrer Schülerschaft die besonderen Fächer Psychologie, Pädagogik, Philosophie und Gesellschaftswissenschaften an, wobei letzteres Sozialwissenschaften, Politische Wissenschaften, Soziologie, Geschichte und Recht beinhaltet. Das OSZ ist eine allgemein- und berufsbildende Schule mit folgenden Bildungsgängen und Abschlüssen:

Bildungsgänge	**Abschlüsse (Ausbildungsdauer)**
Berufliches Gymnasium	Allgemeine Hochschulreife (3 Jahre)
Doppelqualifizierender Bildungsgang	Allgemeine Hochschulreife und staatlich geprüfte Erzieher/in (4 Jahre)
Fachoberschule für Sozialwesen (Abk. FOS)	Allgemeine Fachhochschulreife (2 Jahre) Fachgebundene oder Allgemeine Hochschulreife (3 Jahre)
Berufsoberschule (Abk. BOS)	Allgemeine Fachhochschulreife (1 Jahr),

[1] Geschlechtsbezogene Begriffe werden hier in ihrer männlichen Form verwendet; dies impliziert selbstverständlich auch die weibliche Form. Aus Gründen der Orthographie wird auf die Verwendung des Großbuchstabens „I“ im Wortinneren verzichtet („LeserInnen“); aus Gründen der besseren Lesbarkeit auch auf die permanente Verwendung der kumulativen Form („Leser und Leserinnen“).

	Fachgebundene oder Allgemeine Hochschulreife (2 Jahre)
Berufsfachschule	Staatlich geprüfte/r Sozialassistent/in, Mittlerer Schulabschluss bei entsprechenden Noten (2 Jahre)
Fachschule für Sozialpädagogik	Staatlich geprüfte Erzieher/in (3 Jahre)

Tabelle 1: Ausbildungsgänge an der AFS[2]

Als ehemalige Schülerin und nun Praktikantin der AFS wirken die Lehrer und das Konzept der Schule sehr sozial, hilfsbereit und aufgeschlossen, was möglicherweise mit dem Schwerpunkt der Schule zusammenhängt.

Das Fach Informatik wird an der AFS ausschließlich für die Schüler der Klassenstufen 12 und 13 der gymnasialen Oberstufe als Grundkurs mit drei Wochenstunden angeboten. Da es keinen Informatik-Kurs in der 11. Klasse gibt, kann dieses Fach an der AFS nicht als 3. bzw. 4. Prüfungsfach im Abitur gewählt werden. „Allerdings bietet sich auf Grund der großen Themenbreite und vielfältiger Beziehungen zu anderen Unterrichtsfächern Informatik als Referenzfach für die 5. Prüfungskomponente (Präsentationsprüfung) an.“[3]

Auf der Internetseite des Fachbereichs Informatik der AFS werden, angelehnt an den Berliner Rahmenlehrplan, für die gymnasiale Oberstufe (Abk. RLP der Sek. II) einige Themenschwerpunkte der vier Kurshalbjahre genannt:

- Rechner und Rechnernetze (einschließlich Internet)
- Datenbanken
- Datenschutz
- Kryptologie
- Grundlagen des Programmierens
- Grafikprogramme. [4]

[2] http://www.anna-freud-osz.de/Ausbildungsgaenge.176.0.html [Zugriff: 10.04.2015]

[3] http://www.anna-freud-osz.de/Informatik.91.0.html [Zugriff: 10.04.2015]

[4] http://www.anna-freud-osz.de/Informatik.91.0.html [Zugriff: 10.04.2015]

Dabei wird nur freie und kostenlose Software im Unterricht eingesetzt, sodass Schüler auch daheim mit den Programmen weiterarbeiten können. Dies ist auch einer der Gründe, warum die Programmiersprache Python gelehrt wird. Hinzu kommt, dass Python für alle Plattformen (Windows/Linux/Mac OS) verfügbar ist. Es ist eine interpretative Programmiersprache mit dem Vorteil, „dass ein und dasselbe Programm auf allen Rechnerplattformen läuft.“[5] Für Nutzer sind viele Hilfen, sowie Tutorials und weiterführende Literaturen vorhanden. Für die Zusatzkomponenten wie z.B. dem Turtle-Modul gelten ebenfalls diese Vorzüge. Die Programmiersprache wird in einer internationalen Community weiterhin gepflegt und weiterentwickelt. „Python besitzt einen interaktiven Modus. Sie können einzelne Befehle direkt eingeben und ihre Wirkung beobachten. Python unterstützt das Experimentieren und Ausprobieren. Das erleichtert das Erlernen neuer Programmierkonzepte und hilft vor allem Anfängern.“[6]

[5] WEIGEND: *Python 3: Lernen und professionell anwenden,* Seite 29.

[6] WEIGEND: *Python 3: Lernen und professionell anwenden,* Seite 21.

2. Bedingungsanalyse

Die Bedingungsanalyse dient, laut Hilbert Meyer, der Erfassung und der didaktischen Bewertung der für eine Unterrichtseinheit wesentlichen Einflussfaktoren, die sich unterstützend oder hemmend auf den Unterrichtsprozess und die Unterrichtsergebnisse auswirken können.[7] Bei der Analyse wird, auf Empfehlung von H. Meyer, eine Unterscheidung in vier Bedingungsfeldern vorgenommen. In Kapitel 2.1 werden die Lernvoraussetzungen der Schüler sowie die Klassensituation erläutert. Kapitel 2.2 befasst sich mit der Einordnung der Stunden in die Unterrichtseinheit und fasst die Richtlinien- und Fachvorgaben zusammen, bevor dann im dritten Feld die Lehrvoraussetzungen der Lehrperson thematisiert werden. Die institutionellen Rahmenbedingungen, die von der Schule selbst auf den Unterricht einwirken, werden als viertes Feld im Laufe der Beleuchtung der drei anderen Felder ebenfalls dabei dargestellt.

2.1 Klassensituation & Lernvoraussetzungen

Der Informatik Grundkurs findet derzeit jahrgangsübergreifend statt, da eine zu geringe Anzahl an Schülern diesen Kurs gewählt haben. Seit Beginn des Schuljahres haben vier Schüler den Kurs, z.B. aufgrund von Überschneidungen im Semesterplan, verlassen, sodass zu Beginn meiner Hospitation (Ende November) nur noch vier Schüler am Kurs teilnahmen.

Da der Leistungsstand der vier Schüler sehr heterogen ist, habe ich, angelehnt an Hilbert Meyers „pragmatischen Kompetenzstufenmodells“[8] und den Konstruktionsregeln, ein Kompetenzstufenmodell entwickelt, nach welchem man die Leistungen der einzelnen Schüler einordnen kann, wobei alle Schüler sich mindestens auf Stufe 1 befinden.

---

[7] MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 130.

[8] MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 157.

Stufe	Kompetenzdimensionen
1	Die Schüler können Gegenstände naiv sortieren bzw. suchen.
2	Die Schüler können Sortier- bzw. Suchalgorithmen nachvollziehen und nach Vorgabe an Gegenständen oder an Zahlen (Belegungstabellen) ausführen.
3	Die Schüler können die Vorgehensweise von Sortier- bzw. Suchverfahren erläutern, überflüssige Arbeitsschritte identifizieren und die Vor- und Nachteile der Verfahren angeben.
4	Die Schüler können problemorientiert Sortier- bzw. Suchverfahren auswählen und anwenden und Algorithmen in Form von Pseudo-Codes und Struktogrammen darstellen.
5	Die Schüler können Algorithmen implementieren.

Tabelle 2: Kompetenzstufenmodell

Aufgrund der sehr geringen Anzahl an Schülern werden in der folgenden Tabelle die Voraussetzungen, Interessen und Vorkenntnisse der Schüler stichpunktartig einzeln dargestellt, um so auch die extreme Heterogenität zu veranschaulichen.

	W.	**A.**	**B.**	**C.**
Jahrgang	13. Klasse, kurz vor den Abiturprüfungen.	12. Klasse.	Wiederholt zur Zeit die 12. Klasse.	12. Klasse, möchte diese im nächsten Jahr wiederholen.
Sozio-kulturelle Voraus-setzungen	In Berlin geboren und beherrscht die deutsche Sprache.	In Berlin geboren und beherrscht die deutsche Sprache.	In Berlin geboren und beherrscht die deutsche Sprache.	In Berlin geboren und beherrscht die deutsche Sprache. Hat türkische Wurzeln.
Arbeits- und Sozial-verhalten	Stets anwesend, höflich, tolerant und beteiligt sich am Unterricht. Sie hat zu den drei Mitschülern keinen außerschulischen Kontakt.	Stets anwesend, höflich, tolerant und beteiligt sich am Unterricht. C. und A. sind gute Freunde.	Von 01. - 02.2015 nur zweimal anwesend, spricht häufig unerlaubt, sorgt für Unruhe oder spielt mit seinem Smartphone.	Von 01. - 02.2015 nur einmal anwesend, lässt sich schnell ablenken, ist eher still und höflich, beteiligt sich wenig am Unterrichts-geschehen.
Interessen	App-Programmierung	App-Programmierung, selbstfahrende Autos, Roboter	App-Programmierung, Hacken, Roboter	App-Programmierung

Inhaltliche Vorkennt-nisse	Breite und korrekte Vorkenntnisse und kann selbstständig damit weiter-arbeiten. <u>Inhalte:</u> • Python Programmie-rung: Datentypen (ganze Zahlen, Gleitkomma-zahlen, Strings, Boolean), Ein- und Ausgabe, bedingte Anweisungen, while- und for-Schleifen, Funktionen, Parameter-übergabe • Algorithmus-begriff • Sortieralgo-rithmen Selectionsort & Quicksort • Strukto-gramme • Belegungs-tabellen	Basale Vorkenntnisse, aber braucht Auffrischungen und Hilfestellungen. <u>Inhalte:</u> • Python Programmie-rung: Datentypen (ganze Zahlen, Gleitkomma-zahlen, Strings, Boolean), Ein- und Ausgabe, bedingte Anweisungen, while- und for-Schleifen, Funktionen, Parameter-übergabe • Algorithmus-begriff • Sortieralgo-rithmen Selectionsort & Quicksort • Strukto-gramme • Belegungs-tabellen	Sehr große Kenntnisdefizite, benötigt viel Hilfe und Unterstützung. <u>Inhalte:</u> • Algorithmus-begriff • Sortieralgo-rithmen Selectionsort & Quicksort • Strukto-gramme • Belegungs-tabellen War bei der Unterrichtsreihe von Frau L. zur Python Programmierung nur einmal anwesend.	Sehr große Kenntnisdefizite, benötigt viel Hilfe und Unterstützung. <u>Inhalte:</u> • Algorithmus-begriff • Sortieralgo-rithmen Selectionsort & Quicksort • Strukto-gramme • Belegungs-tabellen War bei der Unterrichtsreihe von Frau L. zur Python Programmierung gar nicht anwesend.

Tabelle 3: Voraussetzungen, Interessen und Vorkenntnisse der Schüler

Das Klassenklima wirkt sehr freundlich und aufgeschlossen. Es gibt keine Verfeindungen unter den vier Schülern. Leider sind B. und C. seit dem neuen Semesterbeginn im Januar nur noch sehr selten anwesend. Frau L.s Bemühungen, per E-Mail und durch Durchsagen im Sekretariat, die beiden Schüler umzustimmen, waren vergebens. A. berichtete, dass B. plane im Sommer 2015 eine Ausbildung anzufangen. C. soll unglücklich mit seiner Leistungskurswahl sein und daher die Wiederholung der 12. Klasse planen.

Im Informatikunterricht werden keine Lehrbücher verwendet. Die Schule hat auch keine Lehrbücher zur Verfügung. Frau L. teilt meist selbsterstellte Arbeitsblätter aus. So zum

Beispiel auch Skripte zur Python Programmierung, welche sie aus mehreren Literaturquellen zusammenstellt.

Die AFS verfügt über zwei Computer-Räume mit je 17 funktionsfähigen PCs. Auf allen Rechnern ist das Betriebssystem Windows. Im Raum, in dem Informatik unterrichtet wird, sind die Tische U-förmig direkt entlang der Wände angeordnet. Auf ihnen sind je zwei mit Monitoren etc. ausgestattete Computer positioniert sind. Die Blickrichtung der Schüler ist beim Arbeiten am PC daher gen Wand. In der Mitte des Raumes sind weitere Tische (ohne Computer) I-förmig angeordnet, so dass die Schüler einander gegenübersitzen. Der Raum ist mit einem mobilen, zweitürigen, insgesamt 6 m² großem Whiteboard und einem Beamer, welcher mit dem Lehrer-PC verbunden ist, ausgestattet.

2.2 Einordnung der Stunden in die Unterrichtseinheit

„Das Thema der Stunde benennt konkret, was der Unterrichtsgegenstand sein soll.“[9]

Für einen besseren Überblick werden im Folgenden die Themen der Unterrichtsstunden tabellarisch dargestellt. Die in blau gefärbten Unterrichtsblöcke wurden von mir eigenständig unterrichtet, die in grün hinterlegten Blöcke wurden teilweise von mir und teilweise von Frau L. unterrichtet. Bei den anderen nicht farbig hinterlegten Stunden habe ich häufig den Schülern Hilfestellungen beim Programmieren oder auch Feedbacks zu ihren Vorträgen gegeben.

Tag	**Thema**
Fr. 28.11.2014	Algorithmusbegriff, Einführung in Sortierverfahren
Mi. 03.12.2014	Übung zu Selectionsort, Struktogramm zu Selectionsort
Fr. 05.12.2014	Quicksort, Demoprogramme
Fr 12.12.2014	Schülervortrag von W. zu „Computerschädlingen“ Schülervortrag von C. zum „Bundesdatenschutzgesetz“

[9] MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 196.

Mi. 17.12.2014	Schülervortrag von A. zu „Soziale Netzwerke“
Fr. 19.12.2014	Schülervortrag von B. zu „Gefahren der Vernetzung“
Fr. 09.01.2015	Anmeldung und Tutorials auf touchdevelop.com
Mi. 14.01.2015	Tutorials auf touchdevelop.com
Fr. 16.01.2015	Vorstellen der HA zu Problemen/Grenzen/Schwierigkeiten der App-Entwicklung auf touchdevelop.com
Fr. 23.01.2015	Einführung in Python (Taschenrechner-Modus, Ein- & Ausgabe, einfache Datentypen)
Mi. 28.01.15	Python: Entscheidungen
Fr. 30.01.2015	Python: Wiederholungen
Fr. 13.02.2015	Python: Funktionen
Mi. 18.02.2015	Python: Funktionen (Parameterübergabe)
Fr. 20.02.2015	1. Test, Besprechung und Python: Übungsaufgaben
Fr. 27.02.2015	Klausur
Mi. 04.03.2015	Klausurbesprechung, Übungsaufgaben und Animationen von Selectionsort und Quicksort beschreiben
Fr. 06.03.2015	Sortieralgorithmus Bubblesort mit Struktogramm & Belegungstabelle
Fr. 13.03.2015	Sortieralgorithmus Bubblesort programmieren und vorstellen, Einführung in Suchalgorithmen (lineare und binäre Suche)
Mi. 18.03.2015	Suchalgorithmen (lineare und binäre Suche)
Fr. 20.03.2015	Test und Besprechung; Verabschiedung mit Kaffee und Kuchen

Tabelle 4: Themen der Unterrichtsstunden

Frau L. und ich haben gemeinsam geplant, was und wie ab dem Januar unterrichtet werden soll. Da die Schüler sich einheitlich für die App-Programmierung interessierten, haben wir im Januar mit den Tutorials auf der Internetseite www.touchdevelop.com von Microsoft Research begonnen, um die Schülermotivation zu steigern. Denn wie auch Hilbert Meyer

bereits vermutet: „Wenn Schüler Interesse an den gestellten Aufgaben haben, lernen sie auch besser und mehr.“[10]

Ähnlich wie bei Scratch[11] kann man bei TouchDevelop per Drag & Drop fertige Programm-Elemente zusammenfügen, um so einem Anfänger beispielsweise den Körper einer Schleife visuell zu verdeutlichen. Auf der Internetseite wird der Vorteil hervorgehoben: „TouchDevelop runs in Internet Explorer, Chrome, Firefox and Safari on Windows, Windows Phone, Mac OS, Linux, iPad, iPhone, Chromebook or Android. No installation required, it just works.“[12] Man hat auf der Internetseite ebenfalls die Möglichkeit fertige Programme von anderen Usern zu verwenden bzw. nachzuvollziehen. Die Schüler hatten daher die Hausaufgabe bis zum 16.01.15 sich ein bereits programmiertes Spiel auszusuchen, es nachzuvollziehen und im Plenum zu präsentieren und mögliche Probleme, Grenzen sowie Schwierigkeiten der App-Entwicklung zu diskutieren.

Im Anschluss wurde die Programmiersprache Python durch die Eingabe von leichten Anweisungen in der Python Shell eingeführt. Dafür wurde das erste sowie zweite Arbeitsblatt von Frau L., welche im Anhang unter 11.1.1 und 11.1.2 aufgeführt sind, zusammengestellt und verwendet. Abwechselnd wurde durch einen Lehrervortrag Wissen vermittelt und durch Einzelarbeit der Schüler am PC das Neuerlernte geübt und gefestigt. Dabei wurde das Wissen zu IF-Anweisungen, WHILE-Schleifen und FOR-Schleifen vor allem durch die zahlreichen Aufgaben auf dem sechsten Arbeitsblatt (siehe 11.1.6) vertieft, indem die Schüler für die einzelnen Probleme Lösungen mit Hilfe eines Struktogramms finden und den Algorithmus in einem Python-Programm umsetzen sollten, wobei dies den Schülern nur mit Hilfe und Unterstützung von Frau L. gelang. Darauf aufbauend wurden Funktionen in Python eingeführt, indem auch hier Beispielaufgaben zum Aufbau, zu Eingabeparametern und Rückgabewerten gemeinsam bearbeitet wurden.

---

[10] MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 145.

[11] siehe hierfür auch https://scratch.mit.edu/

[12] https://www.touchdevelop.com/ [Zugriffsdatum 25.04.15]

Meine darauffolgenden Unterrichtsthemen „Sortieralgorithmus Bubblesort“ sowie „lineare und binäre Suche“ bauen auf diesem Wissen auf. Die Schüler haben bereits eine Vorstellung vom Algorithmus-Begriff und erste Erfahrungen im Programmieren gesammelt. Nun gilt es diese Kenntnisse und Fertigkeiten zu vertiefen und vor allem selbstständig anzuwenden. Die Schüler sollen in meinem Unterricht nun eigenständig einen Algorithmus beschreiben und anwenden, einen Pseudo-Code, ein Struktogramm und eine Belegungstabellen dazu erstellen und letztlich diesen Algorithmus (Bubblesort) programmieren.

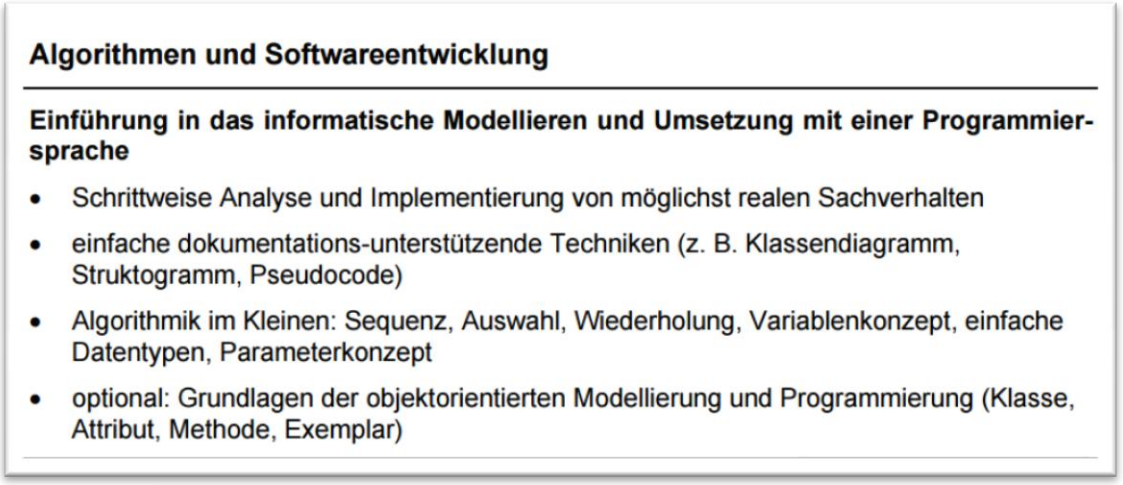
Da die Schüler der AFS in der Einführungsphase (11. Klasse) keinen Informatikunterricht haben und der Grundkurs keine 3. oder 4. Prüfungskomponente im Abitur darstellen kann, treffen die im RLP der Sek. II genannten Zielsetzungen, Kompetenzen und Inhalte der Einführungsphase auch auf den Informatikkurs der 12. Klasse der AFS zu, denn wie „in der Einführungsphase kommen Schülerinnen und Schüler mit unterschiedlichen Kenntnissen und Fähigkeiten zusammen. [...] Die vier Themenbereiche für die Einführungsphase sind identisch mit den Empfehlungen für die Gestaltung des Wahlpflichtunterrichts in den Jahrgangsstufen 9 und 10 im verkürzten gymnasialen Bildungsgang.“[13]

Eines der vier Themenbereiche ist „Algorithmen und Softwareentwicklung“. Nachfolgend ist diesbezüglich ein Auszug aus dem Rahmenlehrplan gegeben, wobei „im Vordergrund Programmstrukturen (Algorithmik im Kleinen), das Variablenkonzept und elementare Datentypen stehen.“[14]

---

[13] RLP der Sek. II, Seite 3.

[14] RLP der Sek. II, Seite 3.



Algorithmen und Softwareentwicklung

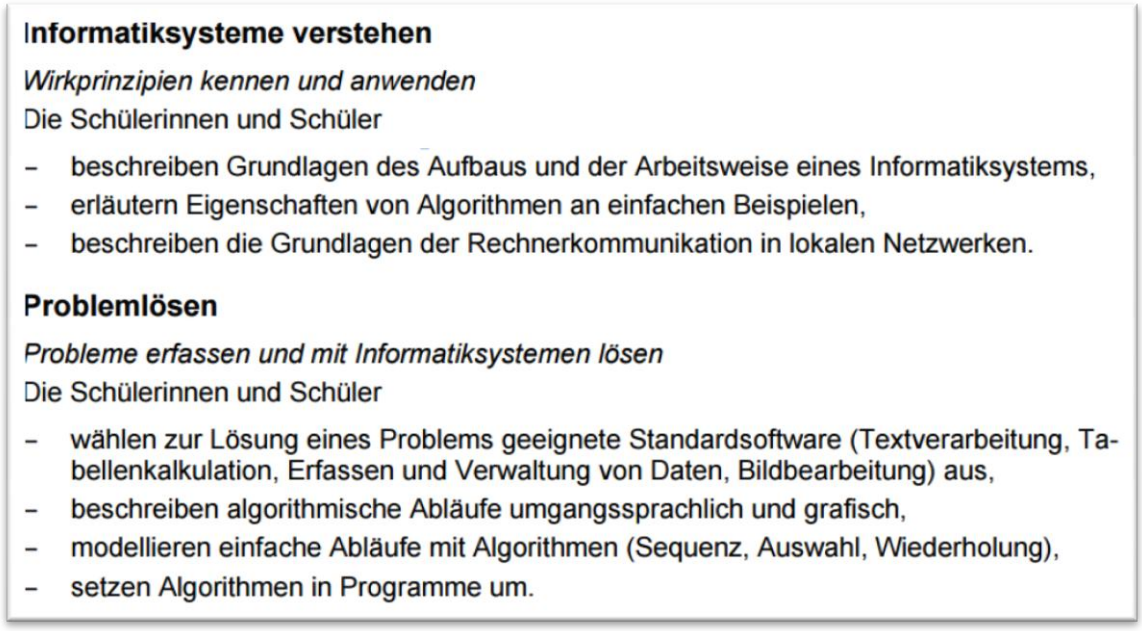
Einführung in das informatische Modellieren und Umsetzung mit einer Programmiersprache

- Schrittweise Analyse und Implementierung von möglichst realen Sachverhalten
- einfache dokumentations-unterstützende Techniken (z. B. Klassendiagramm, Struktogramm, Pseudocode)
- Algorithmik im Kleinen: Sequenz, Auswahl, Wiederholung, Variablenkonzept, einfache Datentypen, Parameterkonzept
- optional: Grundlagen der objektorientierten Modellierung und Programmierung (Klasse, Attribut, Methode, Exemplar)

Abbildung 1: Zweiter Themenbereich der Einführungsphase aus dem RLP der Sek. II, Seite 4.

In Bezug auf das Thema meiner Unterrichtsreihe werden neben inhaltsbezogenen Kompetenzen wie „Informatiksysteme verstehen" auch prozessorientierte Kompetenzen wie das „Problemlösen" gefördert, welche inhaltlich den fachlichen Kompetenzen der EPA Informatik entsprechen.

Da alle vier Schüler des Grundkurses keinen Informatikunterricht in der Sekundarstufe I hatten, gelten die im RLP der Sek. II genannten Eingangsvoraussetzungen für einen erfolgreichen Kompetenzerwerb als noch zu erreichendes Ziel dieses Kurses. Der folgende Auszug skizziert die zu erreichenden fachlichen Anforderungen.



Informatiksysteme verstehen

Wirkprinzipien kennen und anwenden

Die Schülerinnen und Schüler

- beschreiben Grundlagen des Aufbaus und der Arbeitsweise eines Informatiksystems,
- erläutern Eigenschaften von Algorithmen an einfachen Beispielen,
- beschreiben die Grundlagen der Rechnerkommunikation in lokalen Netzwerken.

Problemlösen

Probleme erfassen und mit Informatiksystemen lösen

Die Schülerinnen und Schüler

- wählen zur Lösung eines Problems geeignete Standardsoftware (Textverarbeitung, Tabellenkalkulation, Erfassen und Verwaltung von Daten, Bildbearbeitung) aus,
- beschreiben algorithmische Abläufe umgangssprachlich und grafisch,
- modellieren einfache Abläufe mit Algorithmen (Sequenz, Auswahl, Wiederholung),
- setzen Algorithmen in Programme um.

Abbildung 2: Ausschnitt der Eingangsvoraussetzungen im RLP der Sek II, Seite 12.

Ein schulinternes Curriculum ist für das Fach Informatik an der AFS nicht vorhanden. Frau L. findet, dass Sortier- und Suchalgorithmen relevant für den Lehrplan sind, da diese Algorithmen anschaulich gut zu gestalten und wie sie sagt „zum Anfassen" sind. Vor allem ist das Thema für ihre Schüler geeignet, da es mit Einstiegskenntnissen bewältigt werden kann und sich gut für die Vertiefung der Python Programmierkenntnisse eignet.

2.3 Lehrvoraussetzungen der Lehrperson

Laut Hilbert Meyer sind Lehrvoraussetzungen die geistigen, körperlichen und materiellen Grundlagen des zielorientierten, effektiven und ethisch akzeptablen Lehrerhandelns im Unterricht. Die Medienausstattung eines Klassenraumes nennt er als ein Beispiel für die materiellen Grundlagen, die personenbezogenen Anteile hingegen definiert H. Meyer als Lehrkompetenz: „Lehrkompetenz bezeichnet die durch Erfahrung und Lernen erworbene Fähigkeit eines Lehrers, in immer wieder neuen, nicht genau planbaren Unterrichtssituationen kreativ, situationsangemessen und auf der Grundlage eines mit den Kollegen geteilten Berufsethos zu arbeiten."[15]

Bezogen auf meine Unterrichtsreihe habe ich mein Fachwissen zum Thema in zahlreichen Fachbüchern auffrischen können, da Such- und Sortieralgorithmen ein klassischer Bestandteil von Literatur zu Datenstrukturen und Algorithmen ist. Das Erlernen der Programmiersprache Python stellte aus den in Kapitel 1.1 bereits erläuterten Gründen keine große Hürde für mich dar. Zahlreiche Hilfestellungen, Tutorials und Skripte zur Python Programmierung im Internet erleichtern einem das Selbststudium ungemein. Mein Methodenrepertoire für meine Unterrichtsreihe muss sowohl die Fragen-Taxonomien des fragend-entwickelnden Unterrichts, sowie das sinnhafte Stellen von Lernaufgaben beinhalten. Ich bin gespannt auf die Umsetzung der erlernten Theorie in die Praxis. Auch auf den praktischen Umgang mit nicht planbaren Situationen und mit heterogenen Leistungsständen der Schüler eines Kurses bin ich neugierig. Zu meiner persönlichen Entwicklungsaufgabe zählt die Steigerung der Schülermotivation am Fach Informatik.

---

[15] MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 162.

3. Lernziele

Ein Lernziel ist nach der Definition von Hilbert Meyer „die sprachlich artikulierte Vorstellung über ein gewünschtes Lernergebnis.“[16] „Das, was die Schüler am Stundenschluss neu beherrschen, ist das Stunden- oder Lernergebnis“[17], wobei H. Meyer zwischen geplanten und ungeplanten, sowie zwischen direkt beobachtbaren und nicht beobachtbaren Lernergebnissen unterscheidet.

Hilbert Meyer empfiehlt bei der Formulierung von Lernzielen zwei Abstraktionsstufen, („Hauptziel“ und „Teilziele“) vorzusehen. „Das Hauptziel sollte dann allerdings so breit formuliert sein, dass die kognitiv-fachlichen, die emotionalen und die sozialen Anteile zumindest angedeutet werden.“[18] Formuliert man die Lernziele ganz konkret aus, „sodass die Zielerreichung präzise kontrolliert werden kann, so spricht man auch von der Lernzieloperationalisierung.“[19]

Benjamin Bloom und seine Mitarbeiter ordnen Lernziele bereits in den 50er und 60er Jahren in den USA in die drei Dimensionen „kognitive, affektive und psychomotorische Lernziele“ ein. Mit Hilfe von den Lernzieltaxonomien von Bloom kann für jede der drei Dimensionen eine Hierarchisierung der Lernziele vorgenommen werden, wobei hier für eine weitere Ausdifferenzierung der Stufen auf weiterführende Literatur[20] verwiesen wird:

Kognitive Lernziele „beziehen sich auf das Denken, Wissen, Problemlösen, auf Kenntnisse und intellektuelle Fähigkeiten.“	1. Kenntnisse (Knowledge)
	2. Verständnis (Comprehension)
	3. Anwendung (Application)
	4. Analyse (Analysis)
	5. Synthese (Synthesis)
	6. Beurteilung (Evaluation)

16 MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 193.

17 MEYER: *Trainingsbogen zur Lernzielanalyse*, Seite 2.

18 MEYER: *Fehler-Vermeidung beim Lernzielformulieren,* Seite 2.

19 MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 194.

20 BLOOM, Benjamin: *Taxonomie von Lernzielen im kognitiven Bereich,* Weinheim 1972.

Affektive Lernziele „beziehen sich auf die Veränderung von Interessenlagen, auf die Bereitschaft, etwas zu tun oder zu denken und auf die Entwicklung dauerhafter Werthaltung.“	1. Beachten (Attending)
	2. Reagieren (Responding)
	3. Wertung (Valuing)
	4. Wertordnung (Organization)
	5. Bestimmtsein durch Werte (Characterization by a Value or a Value Complex)
Psychomotorische Lernziele „beziehen sich auf die manipulativen und motorischen Fähigkeiten eines Schülers.“[21]	1. Imitation
	2. Manipulation
	3. Präzisierung
	4. Handlungsgliederung
	5. Naturalisierung

Tabelle 5: Dimensionen der Lernziele nach Bloom

Der Grad der Komplexität wächst von Stufe 1 aufwärts stetig an, wobei (wie beim Kompetenzstufenmodell) die Stufen aufeinander aufbauen. Die Lernzielhierarchisierung kann laut H. Meyer bei der Analyse von Unterricht und von Unterrichtsschwierigkeiten helfen.

3.1 Hauptziel der Unterrichtseinheit

Die Schüler sollen Algorithmen handelnd (enaktiv) ausführen können. Sie sollen durch die graphische Darstellung von der Anwendung eines Algorithmus mindestens die Vorgehensweise erläutern können und unnötige Schritte sollen identifiziert werden. Die Schüler sollen lernen Algorithmen mit der Programmiersprache Python zu programmieren, wobei sie dafür erlernte dokumentations-unterstützende Techniken nutzen sollen.

Die Schüler sollen ein Bewusstsein für angemessene und respektvolle (sowohl positive als auch negative) Kritikäußerungen an den Ideen und Lösungen ihrer Mitschüler entwickeln, Kritik auch selbst annehmen und somit partnerschaftlich handeln.

[21] MEYER: *Leitfaden zur Unterrichtsvorbereitung,* 12. Auflage, Seite 143.

3.2 Hauptziele der einzelnen Unterrichtsstunden

In der folgenden Tabelle wird das Hauptziel für jeden von mir unterrichteten Block angegeben.

Fr. 06.03.2015	Die Schüler sollen unterstützende Techniken für die Implementierung des Bubblesort-Algorithmus verwenden können.
Fr. 13.03.2015	Die Schüler sollen den Bubblesort-Algorithmus in der Programmiersprache Python implementieren und Suchalgorithmen nennen sowie ausführen können.
Mi. 18.03.2015	Die Schüler sollen Suchalgorithmen nennen, erläutern sowie ausführen und miteinander vergleichen können.

Tabelle 6: Hauptziele der Unterrichtsstunden

3.3 Teilziele der Unterrichtsstunden

Auch hier eignet sich eine Tabelle für eine übersichtliche Darstellung der Teilziele einzelner unterrichteter Blöcke.

Fr. 06.03.2015	Die Schüler sollen ... • den Sortieralgorithmus Bubblesort anhand einer Animation beschreiben können. • unnötige Arbeitsschritte des Algorithmus identifizieren können. • den Sortieralgorithmus handelnd auf Spielkarten anwenden können. • für den Algorithmus einen Pseudo-Code entwerfen können. • für den Algorithmus ein Struktogramm erstellen können. • anhand eines Struktogramms eine Belegungstabelle erstellen können. • Struktogramme ihres Mitschülers beurteilen, kritisieren und verbessern können. • Kritik zu ihrem Struktogramm annehmen und umsetzen können.

	• den Bubblesort Algorithmus in der Programmiersprache Python implementieren können (nur unter günstigen Umständen evtl. teilweise zu erreichen).
Fr. 13.03.2015	Die Schüler sollen ... • Listenoperationen in Python anwenden und deren Funktion erklären können. • den Bubblesort Algorithmus in der Programmiersprache Python implementieren können. • ihren Programmcode präsentieren und erläutern können. • ihre eigene Vorgehensweise bei der Suche nach einem DVD-Titel bzw. bei der Suche nach einer gedachten Zahl ihres Partners beschreiben können. • die allgemeine Vorgehensweise von der linearen und binären Suche beschreiben und miteinander vergleichen können.
Mi. 18.03.2015	Die Schüler sollen ... • den Bubblesort Algorithmus in der Programmiersprache Python mit zusätzlichen Ausgaben der einzelnen Vergleichsschritte und der Anzahl der Vertauschungen, Vergleiche und Durchläufe implementieren können. • ihren Programmcode präsentieren und erläutern können. • ihre eigene Vorgehensweise bei der konkreten Suche nach einem DVD-Titel bzw. bei der Suche nach einer gedachten Zahl ihres Partners beschreiben können. • die allgemeine Vorgehensweise von der linearen und binären Suche beschreiben und miteinander vergleichen können, um jeweils die Vor- und Nachteile angeben zu können.

Tabelle 7: Teileziele der Unterrichtstunden

4. Didaktische Strukturierung

„Die Sachanalyse zielt auf die Klärung der wichtigsten mit dem Thema verknüpften Sachfragen. Sie hilft, den Unterrichtsinhalt zu strukturieren und zu portionieren.“[22] „Durch die Didaktische Analyse wird geklärt, ob die Auswahl der Unterrichtsthemen auf der Grundlage eines modernen Bildungsbegriffs gerechtfertigt werden kann.“[23] Nach Hilbert Meyers Empfehlung werden dabei die fünf Grundfragen[24] von Klafki als Strukturierungshilfe verwendet. In der Methodischen Analyse wird die „Zugänglichkeit des vorgesehenen Themas und die daraus abgeleiteten Methoden für die Aufgabenlösung“[25] geklärt. Anstelle des Dreischrittes von Sachanalyse, Didaktischer Analyse und Methodischer Analyse, werden nun die „drei Analysen in einem integrierten Argumentationszusammenhang“[26] dargestellt.

Frau L. führte im Unterricht vom 28.11.2014 den Algorithmusbegriff ein und definierte einen Algorithmus als „die eindeutige Beschreibung eines Verfahrens zur Lösung von gleichartigen Problemen. Er gibt an, wie Eingabegrößen schrittweise in Ausgabegrößen umgewandelt werden.“ Desweiteren führte sie in diesem Zusammenhang auch die Eigenschaften (Allgemeingültigkeit, Ausführbarkeit, Endlichkeit, Eindeutigkeit und Terminiertheit) eines Algorithmus ein. Man kann einen Algorithmus beschreiben als eine „Anleitung zur Lösung einer Aufgabe. Er besteht aus einer Folge von Anweisungen, die so präzise formuliert sind, dass sie auch von einem völlig Unkundigen rein mechanisch ausgeführt werden können.“[27] Beispiele für Alltags-Algorithmen im Leben der Schüler können Backrezepte für Kuchen oder Weihnachtsplätzchen sein, aber auch die Gebrauchsanweisung für das Färben der Haare oder eben auch unbewusst durchgeführte Algorithmen, wie dem Schuhe zuschnüren, dem Zähneputzen oder dem Suchen bzw. dem

---

[22] MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 198-199.

[23] MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 200.

[24] MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 200.

[25] MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 199.

[26] MEYER: *Leitfaden Unterrichtsvorbereitung*, Seite 197.

[27] WEIGEND: *Python 3: Lernen und professionell anwenden,* Seite 27.

Sortieren von Gegenständen. Denn auch Sortier- und Suchalgorithmen sind im Alltag allgegenwärtig. Beim Kartenspielen sortiert man seine Karten auf der Hand, im Telefonbuch oder auf Klassenlisten sind die Telefonnummern von Personen lexigraphisch nach Namen sortiert u.v.m.. Der Hauptgrund für eine Sortierung ist hierbei meist, das schnelle Suchen zu ermöglichen, denn wie ein bekanntes Sprichwort lautet: „Zeit ist Geld“.[28]

Informatiksysteme sind in dem heutigen Lebensumfeld der Schüler der Sekundarstufe II überall und immer gegenwärtig. Zunehmend wird es auch im Alltag notwendig sein, Bindeglieder zwischen Anwendungen zu konfigurieren, um sie dem eigenen Bedarf anzupassen.[29] Somit ist das Erfassen von Problemen und das Lösen dieser mit Informatiksystemen ein wesentlicher Bestandteil der Gegenwart und der Zukunft. „Die Darstellung von Algorithmen in grafischer Form und ihre Umsetzung in ein effizientes Programm sollen den Schülerinnen und Schülern einen Einblick in eine wesentliche Phase der Erstellung von Software vermitteln.“[30]

Wie auch im RLP der Sek. II empfohlen, werden im Unterricht einfache dokumentations-unterstützende Techniken wie Struktogramme[31] und Pseudocodes verwendet, um den Ablauf von Algorithmen intuitiv verständlicher darzustellen. So werden auch die logischen Strukturen, anstelle der Syntax und Semantik der jeweiligen Programmiersprache, in den Vordergrund gestellt. „Sinn und Zweck graphischer Darstellungen für Algorithmen im Unterricht wurden bisher nicht nennenswert diskutiert. Allerdings haben Lehrkräfte auch die Erfahrung gemacht, dass Schüler Struktogramme durchweg zur Dokumentation von fertigen Programmen erstellen und nicht für die Entwicklung von Programmen.“[32] Daher

---

[28] Das Sprichwort stammt von Benjamin Franklin und ist 1748 in seinem Buch *Advice to a Young Tradesman* erstmals erschienen.

[29] https://www.python.org/doc/essays/cp4e/ [Zugriff: 20.05.2015]

[30] RLP der Sek. II, Seite 21.

[31] Auch Nassi-Shneiderman-Diagramm genannt, wurden in den 1970er-Jahren von Nassi und Shneiderman vorgeschlagen und sind durch die DIN66261 vereinheitlicht.

[32] SCHUBERT: *Didaktik der Informatik,* Seite 261.

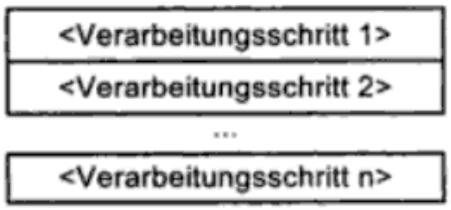
erhalten die Schüler, vor der Implementierung von Bubblesort, die Aufgabe einen Pseudocode und ein Struktogramm zu erstellen.

Die Notation eines Pseudocodes ist textuell, aber besteht im Gegensatz zu einem Programmcode aus stichwortartigen Anweisungen, die umgangssprachlich oder z.B. auch in mathematischer Notation verständlich formuliert sind. Dabei verwendet man Schlüsselwörter wie beispielsweise „solange ... tue“ und Einrückungen, um den Algorithmus näher zu beschreiben. Der Pseudocode ist daher eine nützliche und einfache Technik, um einen Algorithmus zu entwickeln.

Möchte man die algorithmischen Ablaufstrukturen graphisch darstellen, so verwendet man ein Struktogramm. Es setzt sich aus einzelnen rechteckigen Strukturblöcken, den sogenannten Kontrollstrukturen (siehe dafür auch 14.1.5) zusammen, und ist „genauso breit wie sein breitester Strukturblock.“[33]

Wie auch beim Pseudocode sollte dabei keine programmiersprachenspezifische Syntax verwendet werden. Daher kann man sowohl Pseudocodes als auch Struktogramme ohne Programmiererfahrungen erstellen. Dementsprechend haben die Schüler bereits im Vorfeld beide Techniken kennengelernt.

Die folgenden drei Abbildungen[34] zeigen die Umsetzung der einzelnen Strukturelemente in ein Struktogramm.



Sequenzen stellen Programmzeilen dar, deren Anweisungen nacheinander ausgeführt werden.

Abbildung 3: Sequenzen

[33] HEIDERICH: *Technische Probleme lösen mit C/C++: Von der Analyse bis zur Dokumentation,* Seite 104.

[34] HUBWIESER: *Fundamente der Informatik: Ablaufmodellierung, Algorithmen und Datenstrukturen,* Seite 29.

Anweisungen, die nur unter bestimmten Bedingungen ausgeführt werden, stellt man in einem Struktogramm folgendermaßen dar:

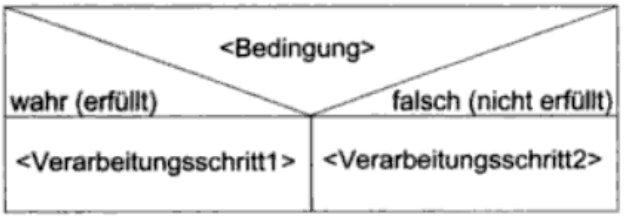
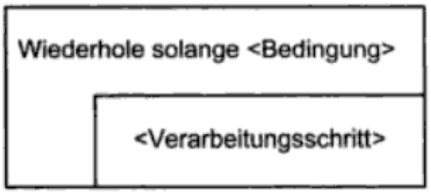


Abbildung 4: Entscheidungen/Alternativen (if)



Eine Schleife wird so lange durchlaufen bis eine Abbruchbedingung erfüllt wird, welche am Anfang der Schleife geprüft wird.

Abbildung 5: Schleifen/Iteration

Sortieralgorithmen gehören zu den „klassischen Themen der praktischen Informatik. An dem Problem des Sortierens lässt sich nämlich gut verdeutlichen, dass Algorithmen sich in ihrer Effizienz unterscheiden.“[35] Es gibt eine Vielzahl an Sortieralgorithmen: Selectionsort, Bubblesort, Quicksort u.v.m.. Beispielsweise sucht der Algorithmus Selectionsort (Sortieren durch Auswählen) aus der unsortierten (Teil-) Liste immer den niedrigsten Wert heraus und sortiert diesen in die sortierte (Teil-) Liste ein.[36] Die Schüler kennen die Vorgehensweise von Selectionsort und Quicksort bereits aus dem Unterricht von Frau L. und haben für ersteres auch ein Struktogramm sowie einen Pseudocode erstellt. Da sie jedoch zu der Zeit keine Programmiersprache beherrschten, haben sie noch keinen Sortieralgorithmus implementiert.

Wie unter anderem auch Karsten Weicker beschreibt, ist Bubblesort „eines der einfachsten Sortierverfahren, das sowohl auf Feldern als auch auf Listen mit gleichem Aufwand arbeitet.“[37] Es eignet sich daher gut, um die gelernten Techniken anzuwenden und zu vertiefen.

---

[35] WEIGEND: *Python 3: Lernen und professionell anwenden,* Seite 208.

[36] SCHOENNINGH: *Informatik 2,* Seite 114.

[37] WEICKER: *Algorithmen und Datenstrukturen*, Seite 56.

„Ein klassisches Gebiet für Visualisierungen sind Sortieralgorithmen. Die Visualisierung hilft, die dynamischen Abläufe während der Ausführung des Algorithmus zu verstehen.“[38] Daher wird als Unterrichtseinstieg eine Animation des Bubblesort Algorithmus gezeigt, bei der Balken der Größe nach sortiert werden. Durch die fragend-entwickelnde Methode werde ich die Schüler hierbei mit konvergenten Wissensfragen dahin lenken, auf wichtige Aspekte in der Visualisierung zu achten und diese zu nennen, um die erste Aufgabe[39] beantworten zu können. Konvergente Wissensfragen eignen sich hier gut, da die Schüler das Wissen erkennen und wiedergeben sollen. In der ersten Aufgabe sollen die Schüler dann die Vorgehensweise anhand der animierten Sortierung beschreiben.

Der Bubblesort Algorithmus durchläuft die Elemente einer Liste von vorne nach hinten und vergleicht jeweils zwei nebeneinander liegende Elemente miteinander. Dabei tauschen zwei Elemente ihren Platz, wenn das erste Element größer ist als das zweite. Bei jedem Durchlauf werden alle Elemente der Liste betrachtet. Der Algorithmus terminiert, sobald (n-1) Durchläufe vollzogen wurden. Ein Pseudocode von Bubblesort könnte beispielsweise folgendermaßen lauten:

```
Wiederhole n-1 mal die folgende Anweisung:
     Wiederhole n-1 mal die folgende Anweisung:
          Vergleiche Element Nr. i mit Element Nr. i+1
               Wenn Element Nr. i größer, dann vertausche beide
```

Der Buchstabe n entspricht der Anzahl der Listenelemente. Es sollen also beispielsweise bei einer Liste mit 5 Elementen insgesamt 5-1 = 4 Durchläufe erfolgen, wobei pro Durchlauf 5-1 = 4 Paare verglichen werden. Der fünfte Durchlauf muss nicht erfolgen, da wir wissen, dass beim (n-1)-ten Durchlauf alle Elemente, bis auf das erste, an der richtigen Position sind und somit auch das erste Element richtig platziert ist.

„Die Laufzeit von Sortierverfahren wird häufig mit der Anzahl der Vergleiche angegeben.“[40] Im besten Fall wäre eine Liste mit n Elementen bereits sortiert und der

---

[38] HARTMANN: *Informatikunterricht planen und durchführen,* Seite 130.

[39] 1. Aufgabe des ersten Arbeitsblattes, siehe Anhang 13.2.1.

[40] WEICKER: *Algorithmen und Datenstrukturen*, Seite 59.

Algorithmus bräuchte aufgrund der Wiederholungen trotzdem (n-1) Durchläufe mit jeweils (n-1) Vergleichen. Im schlechtesten Fall sind alle Elemente der Liste umgekehrt angeordnet, indem das n-te Element an 1. Stelle und das (n-1)-te Element an 2. Stelle usw. positioniert ist. Der Algorithmus bräuchte in diesem Fall, wie in jedem Fall, ebenfalls (n-1) Durchläufe mit jeweils (n-1) Vergleichen, also insgesamt (n-1)*(n-1) Vergleiche und hat daher immer eine Laufzeit von $O(n^2)$[41]. Das bedeutet, dass die Laufzeit quadratisch mit der Anzahl der Listenelemente zunimmt.[42] „Das ist viel - Bubblesort ist ein relativ langsamer Algorithmus.“[43] Da die Schüler dieses Grundkurses die Laufzeit von Algorithmen bislang nicht behandelt haben, müssen sie auf anderem Wege merken, dass der Bubblesort Algorithmus nicht effizient ist und optimiert werden kann. Bei der Visualisierung wird durch farbliche Kennzeichnung erkennbar sein, dass der Algorithmus stets alle Elemente durchläuft, also auch die bereits sortierten Elemente paarweise vergleicht, und dass er nicht terminiert, obwohl bereits alles sortiert ist. Die divergente Wissensfrage „Wie könnte man diesen Sortieralgorithmus optimieren?“ fördert das kritische Denken und die Ideenfindung der Schüler. Sollten die Schüler Hilfe benötigen, kann die zu bejahende konvergente Wissensfrage „Gibt es überflüssige Arbeitsschritte?“ die Schüler anregen, den Algorithmus dahingehend zu verbessern.

Man kann dieses Verfahren optimieren, indem die bereits sortierten Elemente in den darauffolgenden Durchläufen nicht mehr verglichen werden. Der unsortierte Bereich wird somit in jedem Durchlauf um ein Element verkleinert und der Algorithmus bricht ab, sobald keine Vertauschungen mehr notwendig sind. Diesen Unterschied vermerken die Schüler auf ihrem Arbeitsblatt und betrachten dafür als unterstützende Hilfe eine weitere Visualisierung des nun optimierten Bubblesort Algorithmus.

Durch die Optimierung werden im k-ten Durchlauf (n-k) Vergleiche mit k = {1, 2, ..., n-1} durchgeführt und abgebrochen, sobald keine Vertauschungen mehr in einem Durchlauf

---

[41] Für die Beschreibung der Laufzeit von Algorithmen wird die übliche Schreibweise „Groß-Oh-Notation“ (Landau-Symbol) verwendet.

[42] Weiterführende Literatur: Laufzeitanalysen von Algorithmen werden im Buch *Praktische Algorithmik mit Python* von Tobias Häberlein im ersten Kapitel erläutert.

[43] WEIGEND: *Python 3: Lernen und professionell anwenden,* Seite 210.

nötig sind. Summiert man nun die Anzahl der (n-k) Vergleiche mit k = {1, 2, ..., n-1} mit Hilfe der Gauß'schen Summenformel[44] auf, so erhält man 1 + 2 + 3 + ... + (n-1) = (n-1)*(n-1+1)/2 = (n-1)*n/2 = (n^2 - n)/2. Es ergibt sich daher im schlechtesten Fall wieder eine Laufzeit von O(n^2). Für eine bereits sortierte Liste benötigt der optimierte Bubblesort Algorithmus jetzt aber nur noch den ersten Durchlauf mit (n-1) Vergleichen. Da keine Vertauschungen im besten Fall nötig sind, bricht der Algorithmus nach dem ersten Durchlauf ab. Also hat er im besten Fall eine verbesserte Laufzeit von O(n).

J. Bruner unterscheidet die drei Repräsentationsebenen: enaktiv, ikonisch und symbolisch. Der Algorithmus wurde bereits ikonisch betrachtet und sprachlich erläutert. Die enaktive Repräsentation entspricht dem Erfassen von Sachverhalten durch eigenes Tun.[45] Daher erhalten die Schüler von mir jeweils fünf Spielkarten, welche sie beliebig vor sich hinlegen sollen. Sie bekommen dazu den Auftrag den optimierten Bubblesort Algorithmus auf die Karten anzuwenden. „Gemäß Bruner ist dieser Wechsel im Unterricht wichtig: Denkoperationen sollen wenn immer möglich auf mehreren Ebenen durchgespielt werden.“[46] Der Algorithmus soll den Schülern so zugänglicher gemacht werden, sodass dieser sich besser in ihren Gedächtnissen verankert.

Ein Pseudocode zu diesem optimierten Bubblesort Algorithmus entspricht der symbolischen Ebene und ist im Folgenden dargestellt:

```
Setze fertig auf falsch
Setze index auf die Indexzahl des letzten Elements in der Liste
Solange nicht fertig, wiederhole:
      Setze fertig auf wahr
      Für alle i von 0 bis index-1 die folgenden Anweisungen:
            Vergleiche Element Nr. i mit Element Nr. i+1
                  Wenn Element Nr. i größer, dann vertausche beide
                  und setze fertig auf falsch
      Setze index um 1 kleiner
```

---

[44] Die Gauß'sche Summenformel lautet 1 + 2 + 3 + ... + n = n (n+1)/2.

[45] HARTMANN: *Informatikunterricht planen und durchführen,* Seite 116.

[46] HARTMANN: *Informatikunterricht planen und durchführen,* Seite 116.

Da die Schüler noch keinen Pseudocode selbstständig erstellt haben, wird dieser im Plenum durch ein Lehrer-Schüler-Gespräch an der Tafel gemeinsam erarbeitet. Dabei muss beachtet werden, dass die Schüler zuvor noch eine kurze Einführung in Listen brauchen, um im Pseudocode die Indizes der Listenelemente und weiteren Listenoperationen in ihre Überlegungen einbeziehen zu können.[47]

Als zweite Aufgabe ist das Erstellen eines Struktogramms für den optimierten Bubblesort Algorithmus gefordert. Nachfolgend ist eine mögliche Darstellung gegeben:

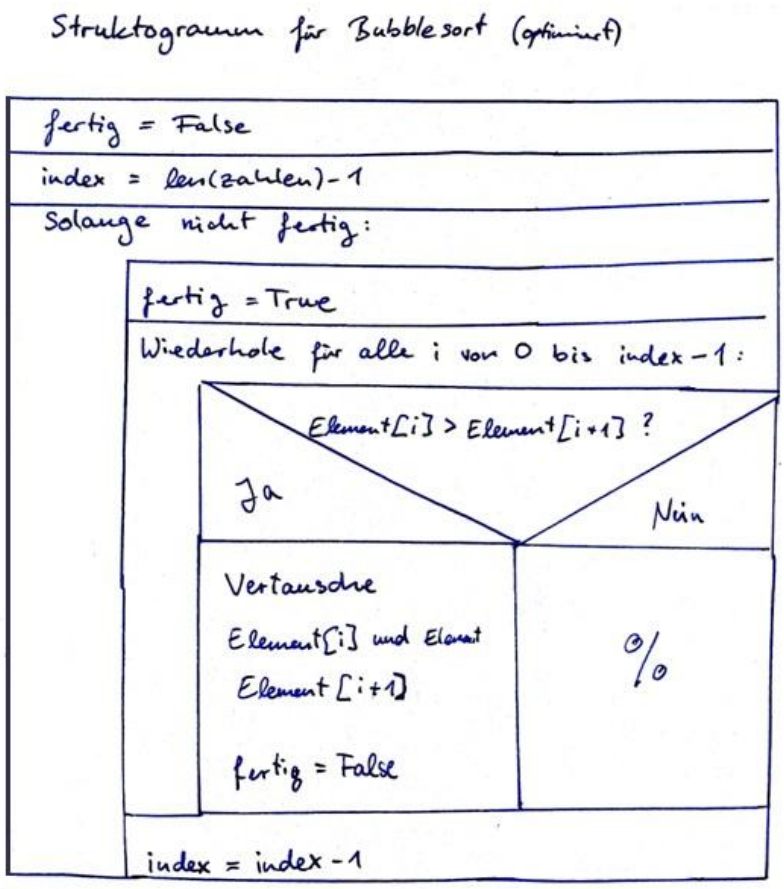


Abbildung 6: Struktogramm für den optimierten Bubblesort Algorithmus

Die grafische (ikonische) Darstellung hilft den Schülern sich ein Bild vom Ablauf des Algorithmus zu machen. Hinzu kommt, dass sie in der dritten Aufgabe die Struktogramme

[47] Dazu im Abschnitt zur Programmiersprache Python mehr.

mit ihrem Partner tauschen und anhand dessen eine Belegungstabelle für eine gegebene Liste erstellen sollen.

7	**2**	**4**	**3**	**9**
2	**7**	4	3	9
2	**4**	**7**	3	9
2	4	**3**	**7**	9
2	4	3	**7**	**9**
2	**4**	3	7	9
2	**3**	**4**	7	9
2	3	**4**	**7**	9
2	**3**	4	7	9
2	**3**	**4**	7	9

Tabelle 8: Belegungstabelle für die Liste [7, 2, 4, 3, 9] mit 4 Vertauschungen und 9 Vergleichen.

Für die Überprüfung der Zielerreichung sollen die Vertauschungen und Vergleiche gezählt und im Plenum verglichen werden. Beim Durchlaufen des Struktogramms auftretende Schwierigkeiten bzw. Fehler sollen vermerkt und anschließend gemeinsam mit dem Partner besprochen und verbessert werden. So üben die Schüler sich auch in Kritikfähigkeit und im respektvollen Umgang miteinander.

Für die Implementierung des optimierten Bubblesort Algorithmus wird die objektorientierte Programmiersprache Python[48] verwendet, welche auch die erste und einzige Programmiersprache ist, die die Schüler (teilweise) beherrschen. In der Fachdidaktik Informatik wird empfohlen, den Unterricht nicht auf maschinenorientierten Sprachen zu gründen. „Es sollten also Sprachen in der Schule gewählt werden, die sich in ihren Konzepten möglichst deutlich von den zugrundeliegenden Kalkülen Registermaschine, λ-Kalkül, Prädikatenkalkül unterscheiden. Dies gilt im imperativen Bereich z.B. für Python und weniger für Java oder C.“[49]

Die Schüler erhalten die Aufgabe in Einzelarbeit mit Hilfe des Struktogramms den Bubblesort Algorithmus in Python zu programmieren. Die Zielerreichung wird

[48] Die verwendete Python-Version ist Python 3.4.

[49] SCHUBERT: *Didaktik der Informatik,* Seite 156.

anschließend zum einen durch das Sortieren von vorgegebenen (unsortierten) Listen überprüft und zum anderen durch mindestens einen Schülervortrag mit der Vorführung und Erläuterung seines Programmcodes gesichert.

Der in Python 3[50] programmierte optimierte Bubblesort Algorithmus ist im Folgenden dargestellt, wobei das Struktogramm zum Vergleich nebenstehend abgebildet ist, um die Parallelen hervorzuheben.

```
##############################################
# Optimierten Bubblesort Algorithmus
# Autor: Mina Ghomi
##############################################

def Bubblesort(zahlen):                          #1
    fertig = False                               #2
    index = len(zahlen)-1                        #3
    while not(fertig) and len(zahlen) > 0:       #4
        fertig = True                            #5
        for i in range(index):                   #6
            if zahlen[i] > zahlen[i+1]:          #7
                temp = zahlen[i]                 #8
                zahlen[i] = zahlen[i+1]          #9
                zahlen[i+1] = temp               #10
                fertig = False                   #11
        index = index-1                          #12
    print(zahlen)                                #13
```

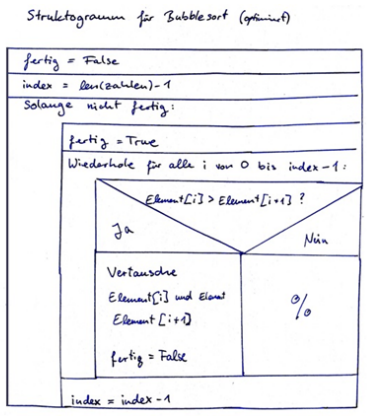


Abbildung 7: Programmcode für Bubblesort

In der ersten Zeile (#1) wird eine Funktion namens `Bubblesort` definiert, der ein Parameter namens `zahlen` übergeben wird. Es ist genaugenommen eine Prozedur, die die übergebene Liste „in place“[51] sortiert, da es keinen Wert zurückgibt. Der Parameter `zahlen` ist eine Liste, also eine veränderbare Sequenz, welche Objekte beliebigen Typs enthalten kann.

---

[50] Python 3 ist eine neue Version, welche mit den Vorgängerversionen nicht mehr kompatibel ist, d.h. „ein Programm, das z.B. in Python 2.5 geschrieben worden ist, läuft in der Regel nicht mehr mit einem Python-3-Interpreter.“ (WEIGEND: *Python 3: Lernen und professionell anwenden*, Seite 21.)

[51] WEIGEND: *Python 3: Lernen und professionell anwenden,* Seite 209.

Auf Listen sind in Python eine Vielzahl von Operationen anwendbar.[52] Die gebräuchlichsten Operationen sind in der folgenden Tabelle zusammengefasst:

Operation	**Ergebnis**
`zahlen = [1,2,3]`	Wertzuweisung
`zahlen[i]`	Das i-te Element der Liste `zahlen`
`x in zahlen`	`True`, wenn Element `x` in der Liste enthalten ist, sonst `False`
`zahlen = zahlen+[e]`	Der Liste wird das Element `e` hinzugefügt.
`a + b`	Konkatenation der beiden Listen `a` und `b`
`n * zahlen`	`n` Kopien der Liste `zahlen`, wobei `n` eine natürliche Zahl ist
`len(zahlen)`	Die Länge der Liste `zahlen`
`min(zahlen)`	Das kleinste Element der Liste `zahlen`
`max(zahlen)`	Das größte Element der Liste `zahlen`

Tabelle 9: Operationen für Listen in Python 3

„Die Elemente einer Liste sind in einer bestimmten Reihenfolge angeordnet und durchnummeriert."[53] Jedes Element in einer Liste hat also einen Index, wobei das erste Element den Index 0 und das n-te Element den Index (n-1) hat. Die Folgende Abbildung stellt diesen Sachverhalt anschaulich an einem Beispiel der Liste `s` dar.

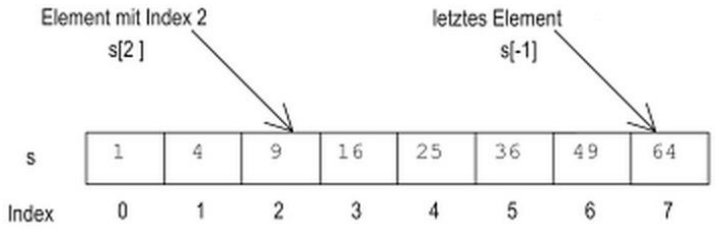


Abbildung 8: Liste mit Index der Elemente[54]

[52] weiterführende Literatur: Tabelle mit Listenoperationen in WEIGEND: *Python 3: Lernen und professionell anwenden,* Seite 199.

[53] WEIGEND: *Python 3: Lernen und professionell anwenden,* Seite 193.

[54] WEIGEND: *Python 3: Lernen und professionell anwenden,* Seite 94.

Die Variable `fertig` wird eingeführt und zu Beginn auf `False` gesetzt (`#2`), um die Terminierung zu sichern. Nur solange `fertig = False` und nur wenn die Liste überhaupt Elemente enthält, also die Länge der Liste größer als Null ist, werden die darauffolgenden Aktionen ausgeführt (`#4`) bzw. beginnt ein Durchlauf. In der Zählschleife (`#6`) erhält die Variable `i` zuerst den Wert 0 und nach jedem Zählschleifendurchlauf wird dieser um 1 erhöht, bis schließlich der Wert der Variable `index` erreicht ist. Die Variable `index` hat im ersten Durchlauf (der while-Schleife) den Wert des Index des letzten Elements in der Liste (`#3`) und wird in jedem Durchlauf um 1 verringert (`#12`), wodurch die bereits sortierten Elemente (am Ende der Liste) bei jedem weiteren Durchlauf nicht mehr verglichen werden. Innerhalb der Zählschleife befindet sich eine einseitige Entscheidung (`#7`). Wenn das linke Element kleiner ist als das benachbarte Element, findet eine Vertauschung statt (`#8-#10`). Bei dieser Vertauschung wird einer Speicherstelle `temp` der linke Wert zugewiesen, dem linken Listenelement der rechte Wert zugewiesen und anschließend dem rechten Listenelement der Wert der Speicherstelle zugewiesen. Somit hat man die Werte der beiden Listenelemente mit Hilfe einer Variablen vertauscht. Die Zeile `#13` dient nur der Ausgabe der sortierten Liste und könnte auch weggelassen werden.

Schüler, die das Programm schnell umsetzen, können zusätzliche Modifizierungsaufgaben lösen. Dabei erweitern sie ihren Programmcode um:

- eine optisch herausstechende sortierte Listenausgabe,
- die Ausgabe der unsortierten Ausgangsliste,
- die Ausgabe von der Anzahl der Vertauschungen, Vergleiche und Durchläufe und
- die Ausgabe von jedem Vergleichsschritt mit der Angabe jedes Durchlaufbeginns

Im Folgenden sind zwei Abbildungen dargestellt. Abbildung 9 stellt den Programmcode des optimierten Bubblesort Algorithmus mit entsprechenden zusätzlichen Modifikationen dar. Die rot hinterlegten und nach dem Doppelkreuz (#) folgenden Kommentare erläutern jeweils die Programmzeile. Ruft man die Prozedur auf, so erhält man nun eine ausführliche Ausgabe. In Abbildung 10 ist die Ausgabe des Beispiel-Aufrufes `BubbleSort(["tisch","Tisch","A","g","a","w"])` dargestellt.

Bei diesem Beispiel sehen die Schüler auch gut, wie die lexikographische Ordnung (A<Aa<a) funktioniert.

```
def BubbleSort(zahlen):

    print(zahlen, "Ausgangsliste")  # Ausgangsliste anzeigen

    fertig = False
    index = len(zahlen)-1    # Index der letzten Zahl in der Liste

    vergleiche = 0          #Zusatz
    vertauschungen = 0      #Zusatz
    durchlaeufe = 0         #Zusatz

    while not(fertig) and len(zahlen) > 0:
        fertig = True

        durchlaeufe = durchlaeufe+1         #Zusatz: Anzahl der Durchläufe zählen
        print(durchlaeufe,". Durchlauf:")

        for i in range(index):

            vergleiche = vergleiche + 1     #Zusatz: Anzahl der Vergleiche zählen

            if zahlen[i] > zahlen[i+1]:
                temp = zahlen[i]
                zahlen[i] = zahlen[i+1]
                zahlen[i+1] = temp
                # Alternative Vertauschung: zahlen[i], zahlen[i+1] = zahlen[i+1], zahlen[i]

                fertig = False

                vertauschungen = vertauschungen + 1  #Zusatz: Anzahl der Vertauschungen zählen

            print(zahlen)   #Zusatz: Jeden Schritt ausgeben

        index = index-1
    #Zusatz: Ausgaben der Anzahlen
    print()
    print("Anzahl der Vertauschungen: ", vertauschungen)
    print("Anzahl der Vergleiche: ", vergleiche)
    print("Anzahl der Durchlaeufe: ", durchlaeufe)
    print("-------------------------------------------------------------------")
    print("Sortierte Liste: ",zahlen)
    print("-------------------------------------------------------------------")
    print()
```

Abbildung 9: Programmcode für Bubblesort Algorithmus mit zusätzlichen Ausgaben

```
['tisch', 'Tisch', 'A', 'g', 'a', 'w'] Ausgangsliste
1 . Durchlauf:
['Tisch', 'tisch', 'A', 'g', 'a', 'w']
['Tisch', 'A', 'tisch', 'g', 'a', 'w']
['Tisch', 'A', 'g', 'tisch', 'a', 'w']
['Tisch', 'A', 'g', 'a', 'tisch', 'w']
['Tisch', 'A', 'g', 'a', 'tisch', 'w']
2 . Durchlauf:
['A', 'Tisch', 'g', 'a', 'tisch', 'w']
['A', 'Tisch', 'g', 'a', 'tisch', 'w']
['A', 'Tisch', 'a', 'g', 'tisch', 'w']
['A', 'Tisch', 'a', 'g', 'tisch', 'w']
3 . Durchlauf:
['A', 'Tisch', 'a', 'g', 'tisch', 'w']
['A', 'Tisch', 'a', 'g', 'tisch', 'w']
['A', 'Tisch', 'a', 'g', 'tisch', 'w']

Anzahl der Vertauschungen:  6
Anzahl der Vergleiche:  12
Anzahl der Durchlaeufe:  3
-------------------------------------------------------------
Sortierte Liste:  ['A', 'Tisch', 'a', 'g', 'tisch', 'w']
-------------------------------------------------------------
```

Abbildung 10: Programm-Ausgabe am Beispiel der Liste ["tisch","Tisch","A","g","a","w"]

„Eine weitere wichtige Problemgruppe sind die Verfahren zum Durchsuchen von Sequenzen, Arrays und Listen.“[55] Die konvergente Denkfrage „Warum sortiert man Daten?“ führt die Schüler selbstständig zum Themenfeld der Suchalgorithmen, wofür sie auch selbstständig Alltagsbeispiele nennen können. Ein typisches Beispiel ist das Suchen eines Eintrages in einem Telefonbuch oder einem Warenbestand.

Durchaus realistisch ist auch die Suche nach einer DVD in einem DVD-Regal. Die vier Schüler erhalten in der Erarbeitungsphase zwanzig unsortierte, in einer Reihe aufgestellte DVDs, denn „die enaktive Repräsentationsform eignet sich besonders für den Einstieg in ein Thema.“[56] Jeder Schüler erhält einen individuellen Suchauftrag für einen speziellen Filmtitel, wobei ein Schüler einen Titel suchen soll, den es im „Regal“ nicht gibt. Die Anzahl der Vergleichsschritte soll beim Suchen gezählt und die jeweilige Vorgehensweise anschließend erläutert werden. Auch hier wird fragend-entwickelnd die Vorgehensweise mit Fragen wie „Wo beginnt die Suche? Was wird verglichen? Wann endet die Suche?“ komplettiert, sodass die Schüler in der Lage sind, die Vorgehensweise der sogenannten linearen Suche auf ihrem Arbeitsblatt zu notieren.

Gegeben sei eine Liste mit n Elementen, in der nach einem Element gesucht werden soll. Bei der linearen Suche, welche auch sequentielle Suche genannt wird, beginnt man mit dem ersten Element der Liste. Stimmt dieses mit dem gesuchten überein, kann die Suche beendet werden. Andernfalls fährt man mit dem nächsten Element fort bis alle n Elemente der Liste betrachtet wurden. Die Suche läuft solange, bis das gesuchte Element erfolgreich gefunden oder bis das Ende der Liste erreicht wurde. Die lineare Suche kann sowohl auf sortierte als auch auf unsortierte Listen angewendet werden, was einen Vorteil dieser Suche gegenüber anderen Suchverfahren darstellt.

Im Gegensatz dazu setzt die binäre Suche eine sortierte Liste voraus und funktioniert „nach dem Motto Teile und Herrsche“[57]. Man wählt ein Element aus der Mitte aus und entscheidet, ob das gesuchte Element kleiner, größer oder gleich dem gewählten Element

---

[55] HUBWIESER: *Fundamente der Informatik: Ablaufmodellierung, Algorithmen und Datenstrukturen,* Seite 198.

[56] HARTMANN: *Informatikunterricht planen und durchführen,* Seite 116.

[57] PERRY: *Jetzt lerne ich Programmieren*, Seite 188.

ist. Wenn das gesuchte Element kleiner ist als das ausgewählte, dann wählt man das mittlere Element des unteren/linken Bereichs. Wenn es größer ist, dann wählt man das mittlere Element des oberen/rechten Bereichs und setzt die Suche fort. Wenn das ausgewählte Element dem gesuchten entspricht, dann ist die Suche erfolgreich beendet. Ebenfalls ist die Suche beendet, wenn es keinen unteren/linken bzw. oberen/rechten Bereich gibt und somit das Element nicht in der Liste enthalten ist.

Auch hier sollen die Schüler aktiv werden und selbstständig die Vorgehensweise der binären Suche erschließen. Dafür erhalten sie in Partnerarbeit einen spielerischen Arbeitsauftrag: „Denk dir eine Zahl zwischen 0 - 1000 und sage bei jedem Rateversuch deines Partners, ob die geratene Zahl zu groß oder zu klein ist. Dein Partner hat maximal 10 Versuche. Wechselt euch ab." Im Plenum werden dann die zum Erfolg geführten Strategien besprochen, sodass die Schüler die allgemeine Vorgehensweise auf ihrem Arbeitsblatt festhalten können.

„Da bei der binären Suche jeder Vergleich bewirkt, dass die Anzahl der noch zu durchsuchenden Elemente im Durchschnitt halbiert wird, ist dieses Vorgehen im Vergleich zur sequentiellen Suche sehr viel effizienter. Ist n die Anzahl der Elemente der Liste, so ist der Aufwand des Algorithmus, der die sequentielle Suche ausführt, von linearer Ordnung, als O(n), während der Algorithmus zur binären Suche nur einen Aufwand logarithmischer Ordnung, also O(log n), bedeutet."[58]

Die Schüler sollen die Effizienz der beiden Suchverfahren anhand des DVD-Regals nachempfinden. Der Schüler mit dem nicht vorhandenen Filmtitel hat bei der linearen Suche bereits gemerkt, dass man im schlimmsten Fall alle n Elemente betrachten muss. Um nun einen Vergleich zwischen den beiden Suchverfahren herzustellen, sollen die Schüler die DVDs mit Hilfe des Bubblesort Algorithmus sortieren und anschließend dieselben Titel nochmal mit dem binären Suchverfahren suchen. Jeder Schüler zählt auch diesmal die Anzahl der Vergleichsschritte. So kann im Plenum zusammengetragen werden, wie viele Vergleichsschritte jeder Schüler bei beiden Verfahren benötigte und wie viele es jeweils im

[58] ABECK: *Kursbuch Informatik I,* Seite 275.

schlimmsten Fall sind. Die Erkenntnis wird sein, dass die binäre Suche gegenüber der linearen Suche effizienter ist.

Wünschenswert wäre jedoch ein Verfahren, das mit konstanter Laufzeit und somit auch unabhängig von n arbeitet. Das Streuspeicherverfahren (Hashing) „ist ein Verfahren, mit dem die Forderung nach einer asymptotischen Laufzeitkomplexität von O(1) unter bestimmten Umständen erfüllt werden kann.“[59] „Beim sogenannten Hashing wird auf den Suchschlüssel eine Berechnungsfunktion (Hashfunktion) angewandt, dessen Ergebnis direkt anzeigt, wo gesucht werden muss.“[60] Anwendung findet dieses Verfahren beispielsweise in Telefonlisten oder bei Kundendaten einer Firma. Dieses Suchverfahren und das Implementieren dieser Verfahren mit Hilfe der unterstützenden Techniken könnte als Fortsetzung der Unterrichtseinheit behandelt werden.

[59] POMBERGER: *Algorithmen und Datenstrukturen: eine systematische Einführung in die Programmierung,* Seite 282.

[60] SCHOENNINGH: *Informatik 2,* Seite 108.

5. Ausführliche Stundenentwürfe

5.1 Verlaufsplanung vom 06.03.2015

Thema: Sortieralgorithmus Bubblesort

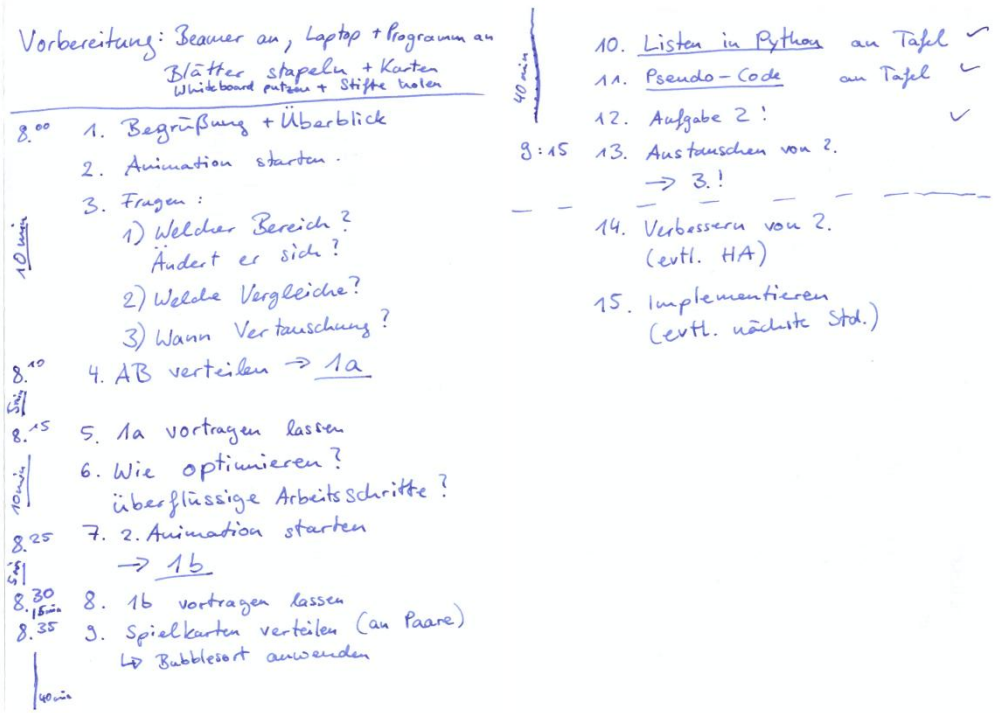
Zeit	**Phasen**	**Unterrichtsschritte/Lehrer-Schülerinteraktion**	**Sozialform & Handlungs-muster**	**Medien**
10 min	Einstieg Einleitung	L. beginnt nach der Begrüßung mit einem Überblick über die heutige Stunde und knüpft dabei an das Ende der letzten Stunde (Betrachtung der Animationen von einigen Sortierverfahren) an. L. startet die erste Animation am Laptop (Beamer). S. betrachten die animierte Sortierung. Um die erste Aufgabe des Arbeitsblattes zu beantworten (Vorgehensweise des Algo. beschreiben) stellt L. gezielte Fragen, deren Antworten dafür nötig sind: L.: „Welcher Bereich wird beim Sortieren in einem Durchlauf bearbeitet? Ändert sich dieser? Wenn ja, wie?“ S.: „Alle Elemente werden in jedem Durchlauf bearbeitet, siehe grüne Linie.“ L.: „Welche zwei Elemente werden jeweils miteinander verglichen?“ S.: „Zwei nebeneinander liegende Elemente werden verglichen, siehe rote Markierungen.“	Plenum Lehrer-Schüler-Gespräch	Beamer, Laptop mit Python 2.7.9

		L.: „Wann findet eine Vertauschung von zwei Elementen statt?“ S.: „Wenn das vordere Element größer ist als das hintere.“ L. verteilt das Arbeitsblatt, mit der Bitte die Aufgabe 1a zu bearbeiten.		AB (siehe Anhang 12.2.1)
5 min	Erarbeitung 1	S. bearbeiten Aufgabe 1a.	Einzelarbeit	AB, Beamer
10 min	Ergebnis-sicherung 1	L. fordert S. auf, die Antwort von 1a vorzutragen. S.: „Der Algorithmus durchläuft die Elemente von vorne nach hinten und vergleicht jeweils zwei nebeneinander liegende Elemente miteinander. Dabei tauschen zwei Elemente ihren Platz, wenn das erste Element größer ist als das zweite. Bei jedem Durchlauf werden alle Elemente betrachtet.“ L.: „Wie könnte man diesen Sortieralgorithmus optimieren?“ Optional: L.: „Gibt es überflüssige Arbeitsschritte?“ S.: „Da der hintere Bereich bereits sortiert ist, ist es unnötig diesen in jedem Durchlauf zu betrachten. Der Algorithmus muss den Bereich in jedem Durchlauf um 1 verkleinern. Wenn keine Vertauschungen notwendig sind, muss der Algorithmus aufhören.“ L. startet die zweite Animation am Laptop (Beamer). S. betrachten die animierte Sortierung.	Plenum Lehrer-Schüler-Gespräch	AB, Beamer, Laptop
5 min	Erarbeitung 2	S. bearbeiten Aufgabe 1b.	Einzelarbeit	AB, Beamer, Laptop
5 min	Ergebnis-sicherung 2	L. fordert S. auf, die Antwort von 1b vorzutragen. S.: „Unterschied: *Der Bereich wird in jedem Durchlauf um 1 verkleinert bzw. der Algo.*	Plenum Lehrer-Schüler-	AB

		bricht ab, sobald keine Vertauschungen mehr notwendig sind.“ L. bestimmt, dass W. mit C., sowie A. mit B. zusammen arbeiten und verteilt jeweils fünf Skat-Spielkarten (mit Zahlen) an die Schülerpaare.	Gespräch	
40 min	Erarbeitung 3	L.: „Wendet den Algorithmus „`BUBBLECLEVER-SORT`“ auf die Spielkarten an.“ S. legen Karten beliebig hin und wenden den cleveren Bubblesort Algo. an. L. führt kurz „Listen in Python“ grob ein und erarbeitet anschließend mit den S. an der Tafel einen Pseudo-Code für den Algorithmus. (S. schreiben mit) L.: „Erstellt in Aufgabe 2 auf einem Extrablatt ein Struktogramm zum cleveren Bubblesort Algorithmus.“ (Anmerken, dass es getauscht wird)	Plenum Lehrervortrag und Lehrer-Schüler-Gespräch Partnerarbeit (Einzelarbeit)	Spielkarten, Whiteboard + Stifte, AB Tafelbild (siehe Anhang 12.2.5)
15 min	Ergebnis-sicherung 3	L.: „Tauscht eure Struktogramme aus und erstellt in Aufgabe 3 eine Belegungstabelle für das Struktogramm eures Partners. Beurteilt das Struktogramm und gebt ggf. Vorschläge für Verbesserungen.“ S. kritisieren gegenseitig ihre Struktogramme und verbessern jeweils ihr eigenes. (Verbesserung ggf. als Hausaufgabe bzw. S. beginnen mit der Implementierung am PC)	Partnerarbeit (Einzelarbeit)	AB, (evtl. PC)

5.1.1 Spickzettel der Stunde vom 06.03.15:

Laut Hilbert Meyer enthält der Spickzettel den schriftlichen Ablaufplan der geplanten Unterrichtsstunde. Es gehört all das auf ihn, was man nicht auswendig behalten kann oder will, aber während der Stunde wissen müsste.[61]



[61] MEYER: *Leitfaden Unterrichtsvorbereitung,* Seite 32-33.

5.2 Verlaufsplanung vom 13.03.2015

Thema: Sortieralgorithmus Bubblesort (und lineare und binäre Suche)

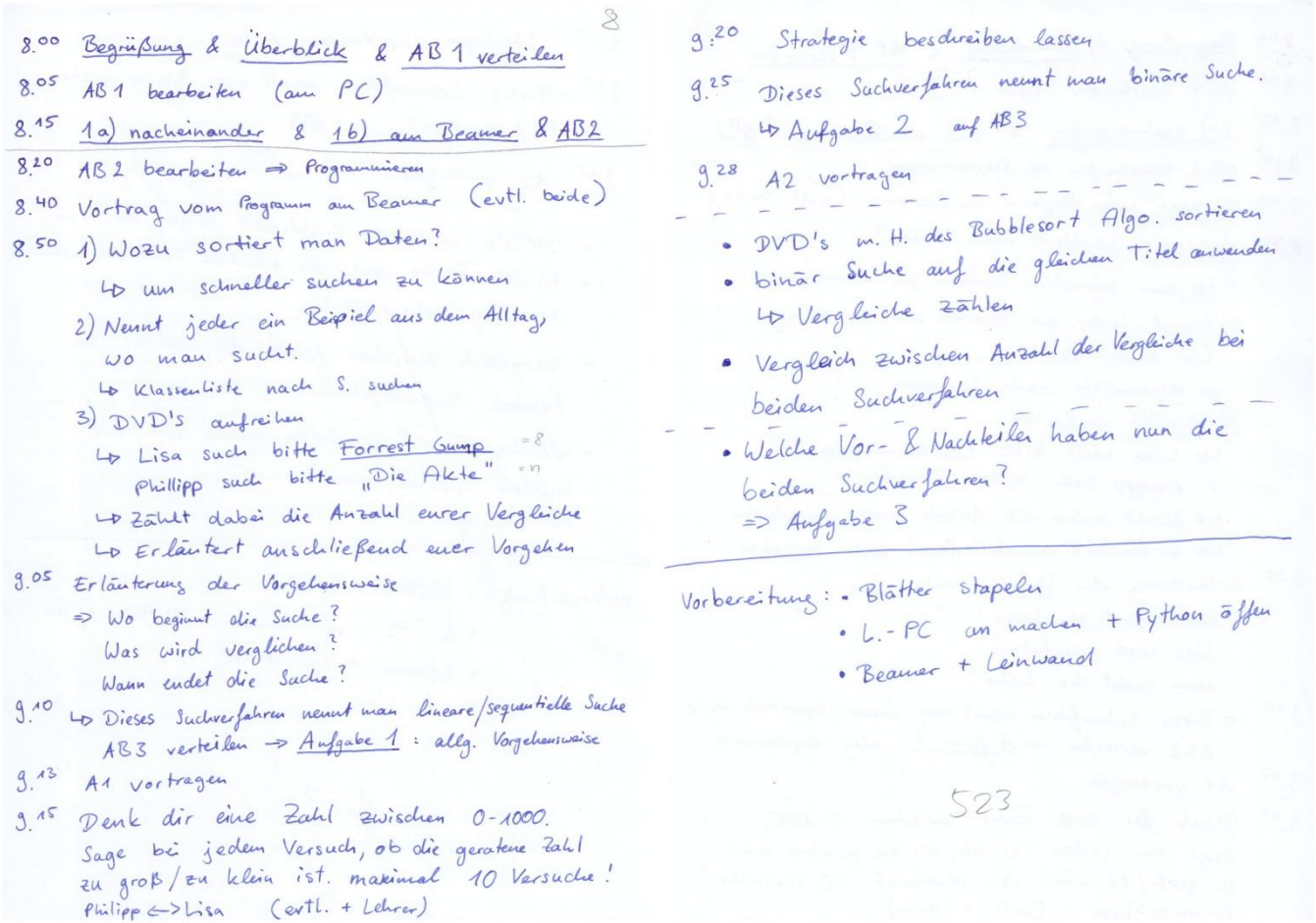
Uhrzeit	**Phasen**	**Unterrichtsschritte/Lehrer-Schülerinteraktion**	**Sozialform & Handlungs-muster**	**Medien**
8:00 – 8:05	Einstieg Einleitung	L. beginnt nach der Begrüßung mit einem Überblick über die heutige Stunde und knüpft dabei an das Ende der letzten Stunde an. L. verteilt AB 1 zur Einführung von Listen in Python.	Plenum	AB 1 (siehe Anhang 12.2.2)
8:05 – 8:15	Erarbeitung 1	S. bearbeiten das erste Arbeitsblatt mit Hilfe des PC's.	Einzelarbeit	AB 1, Schüler-PC
8:15 – 8:20	Ergebnis-sicherung 1	L. forder S. auf, die Tabelleneinträge von 1a) nacheinander vorzutragen. Anschließend führt ein Schüler 1b) am Beamer vor. L. verteilt zweites Arbeitsblatt.	Plenum Schüler-Lehrer-Gespräch	AB1, Beamer, PC, AB 2 (siehe Anhang 12.2.3)
8:20 – 8:40	Erarbeitung 2	S. implementieren Bubblesort. Schnelle bzw. leistungsstarke S. erhalten zusätzlich Arbeitsaufträge auf dem zweiten Arbeitsblatt. Falls Schüler anwesend sind, die letzte Stunde gefehlt haben, erhalten diese das AB der letzten Stunde, um Aufgabe 1 und 2 zu bearbeiten. Video und Pseudo-Code werden dafür bereitgestellt.	Einzelarbeit	AB 2, Schüler-PC, AB + Pseudo-Code + Video (06.03.)

8:40 – 8:50	Ergebnis-sicherung 2	Leistungsstarker Schüler trägt sein Programm am Beamer vor, wobei verschiedene Listen sortiert werden sollen, der Programm-Code erläutert werden soll und (Zusatz:) Vergleiche, Durchläufe und Vertauschungen gezählt werden sollen. S. und L. können daraufhin Fragen stellen. Falls ein anderer Schüler einen anderen Ansatz für die Umsetzung des Algo. hat, wird dieser auch vorgetragen.	Plenum Schüler-Vortrag	Beamer, PC
8:50 – 9:05	Erarbeitung 3	L.: „Wozu sortiert man Daten?“ S.: „Um möglichst schnell auf einzelne Elemente zugreifen zu können bzw. um zu prüfen, ob das Element vorhanden ist. → um schneller suchen zu können“ L.: „Nennt jeder ein Beispiel aus dem Alltag, wo man sucht.“ S.: „In einer Klassenliste, welche nach Nachnamen sortiert ist und verschiedene Informationen (Telefonnummer, E-Mail, Adresse) zu jedem S. aufgelistet sind, nach einem Schüler suchen.“ L. reiht DVDs (mit unsortierten Titeln) auf dem Tisch auf . L.: „W. such bitte den Film „Forrest Gump“ und A. such bitte den Film „Die Akte“. Zählt dabei die Anzahl eurer Schritte/Vergleiche. Erläutert uns anschließend eure jeweilige Vorgehensweise.“ S. suchen ihren Titel, wobei A. seinen nicht finden wird, da dieser nicht existiert.	Plenum Schüler-Lehrer-Gespräch	DVD's

9:05 – 9:10	Zwischen-sicherung 3	Beide erläutern dann ihre Vorgehensweise. L. ergänzt durch gezielte Fragen die Vorgehensweise: Wo beginnt die Suche? Was wird verglichen? Wann endet die Suche? → Hinführung zur sequentiellen Suche: S.: „Man fängt bei der ersten DVD an. Stimmt der Titel mit dem gesuchten Titel überein, kann die Suche beendet werden. Andernfalls fährt man mit dem nächsten Titel fort. Die Suche läuft solange, bis der gesuchte Titel gefunden oder bis das Ende der DVD-Sammlung erreicht wurde.“	Plenum Schüler-Lehrer-Gespräch	
9:10 – 9:13	Erarbeitung 4	L.: „Dieses Suchverfahren nennt man sequentielle oder auch lineare Suche.“ L. verteilt das dritte Arbeitsblatt mit dem Arbeitsauftrag in der ersten Aufgabe die allgemeine Vorgehensweise der linearen/sequentiellen Suche zu erläutern.	Einzelarbeit	AB 3 (siehe Anhang 12.2.4)
9:13 – 9:15	Ergebnis-sicherung 4	Ein S. wird aufgefordert, seine Antwort vorzulesen.	Plenum	
9:15 – 9:20	Erarbeitung 5	L.: „A., denk dir eine Zahl zwischen 0 – 1000. Sage bei jedem Versuch von W., ob die geratene Zahl zu groß oder zu klein ist. W., du hast maximal 10 Versuche, um A.s gedachte Zahl zu ermitteln.“ L.: „Tauscht nun eure Rollen. W. denkt sich eine Zahl und A. ermittelt.“ (Eventuell wiederholen. Falls Strategie nicht erschlossen wurde, macht L. mit)	Partnerarbeit	

9:20 – 9:25	Ergebnis-sicherung 5	L.: „Beschreibt eure Strategie.“ S.: „Man grenzt den Bereich der Zahlen ein, indem man eine Zahl wählt, die in der Mitte der verbleibenden Zahlen liegt. Durch die Aussage „zu groß/zu klein“ fallen die Hälfte der Zahlen weg. Diese Halbierung des jeweiligen Bereichs führt man solange fort, bis man die Zahl gefunden hat.“	Plenum Schüler-Lehrer-Gespräch	
9:25 – 9:28	Erarbeitung 6	L.: „Dieses Suchverfahren nennt man binäre Suche. Beschreibt nun in Aufgabe 2 auf AB 3 die allgemeine Vorgehensweise der binären Suche.“	Einzelarbeit	AB 3
9:28 – 9:30	Ergebnis-sicherung 6	Ein S. wird aufgefordert, seine Antwort vorzulesen. ***** Mögliches Stundenende *****	Plenum	
	Erarbeitung 7	S. werden aufgefordert, die DVD's mit Hilfe des Bubblesort-Algorithmus zu sortieren, um anschließend die binäre Suche darauf anzuwenden. S. sortieren und suchen wieder ihre beiden Titel. Dabei zählen Sie ihre Schritte/Vergleiche.	Einzelarbeit	DVD's
	Ergebnis-sicherung 7	S. berichten, wie viele Schritte sie benötigt haben → Vergleich zur linearen Suche. L.: „Welche Vorteile und Nachteile haben nun die beiden Suchverfahren?“ S.: „Geschwindigkeit der binären Suche kann schneller sein als bei der linearen Suche. Vor allem dann, wenn das Element nicht in der Liste vorhanden ist. Die lineare Suche benötigt jedoch nicht die Voraussetzung, dass die Daten sortiert sein müssen.“ L.: „Haltet diese Vor- und Nachteile in Aufgabe 3 fest.“	Plenum Schüler-Lehrer-Gespräch	

5.2.1 Spickzettel der Stunde vom 13.03.15:



8.00 Begrüßung & Überblick & AB 1 verteilen
8.05 AB 1 bearbeiten (am PC)
8.15 1a) nacheinander & 1b) am Beamer & AB2
8.20 AB 2 bearbeiten ⇒ Programmieren
8.40 Vortrag vom Programm am Beamer (evtl. beide)
8.50 1) Wozu sortiert man Daten?
↳ um schneller suchen zu können
2) Nennt jeder ein Beipiel aus dem Alltag, wo man sucht.
↳ Klassenliste nach S. suchen
3) DVD's aufreihen
↳ Lisa such bitte Forrest Gump
Phillipp such bitte „Die Akte"
↳ Zählt dabei die Anzahl eurer Vergleiche
↳ Erläutert anschließend euer Vorgehen
9.05 Erläuterung der Vorgehensweise
⇒ Wo beginnt die Suche?
Was wird verglichen?
Wann endet die Suche?
9.10 ↳ Dieses Suchverfahren nennt man lineare/sequentielle Suche
AB3 verteilen → Aufgabe 1 : allg. Vorgehensweise
9.13 A1 vortragen
9.15 Denk dir eine Zahl zwischen 0-1000.
Sage bei jedem Versuch, ob die geratene Zahl zu groß / zu klein ist. maximal 10 Versuche!
Philipp ↔ Lisa (evtl. + Lehrer)

9.20 Strategie beschreiben lassen
9.25 Dieses Suchverfahren nennt man binäre Suche.
↳ Aufgabe 2 auf AB3
9.28 A2 vortragen

- DVD's m. H. des Bubblesort Algo. sortieren
- binär Suche auf die gleichen Titel anwenden
  ↳ Vergleiche zählen
- Vergleich zwischen Anzahl der Vergleiche bei beiden Suchverfahren
- Welche Vor- & Nachteile haben nun die beiden Suchverfahren?
  ⇒ Aufgabe 3

Vorbereitung:
- Blätter stapeln
- L.-PC an machen + Python öffnen
- Beamer + Leinwand

523

5.3 Verlaufsplanung vom 18.03.2015

Thema: Sortieralgorithmus Bubblesort sowie die lineare und binäre Suche

Anmerkung: Da wir am 13.03.15 von der zeitlichen Planung abgewichen sind und wir die Verlaufsplanung nur bis zur „Ergebnissicherung 2“ (Schüler-Präsentation des Bubblesort Programms ohne Modifikationen) durchgeführt haben, wird am 18.03. die restliche Planung ausgeführt.

Uhrzeit	**Phasen**	**Unterrichtsschritte/Lehrer-Schülerinteraktion**	**Sozialform & Handlungs-muster**	**Medien**
8:00 – 8:03	Einstieg Einleitung	L. beginnt nach der Begrüßung mit einem Überblick über die heutige Stunde und knüpft dabei an das Ende der letzten Stunde an.	Plenum	
8:03 – 8:20	Erarbeitung 1	S. implementieren Bubblesort (Fortsetzung). Leistungsstarke S. erhalten zusätzlich Arbeitsaufträge auf dem Arbeitsblatt der letzten Unterrichtsstunde.	Einzelarbeit	(evtl. AB), Schüler-PC
8:20 – 8:30	Ergebnis-sicherung 1	Leistungsstarker Schüler trägt sein Programm am Beamer vor, wobei verschiedene Listen sortiert werden sollen, der Programm-Code kurz erläutert werden soll und vor allem (Zusatz:) Vergleiche, Durchläufe und Vertauschungen gezählt werden sollen. S. und L. können daraufhin Fragen stellen.	Plenum Schüler-Vortrag	Beamer, PC
8:30 – 8:45	Erarbeitung 2	L.: „Wozu sortiert man Daten?“ S.: „Um möglichst schnell auf einzelne Elemente zugreifen zu können bzw. um zu prüfen, ob das Element vorhanden ist. → um schneller suchen zu können“	Plenum Schüler-Lehrer-Gespräch (Einzelarbeit	DVD's

		L.: „Nennt jeder ein Beispiel aus dem Alltag, wo man sucht." S.: „In einer Klassenliste, welche nach Nachnamen sortiert ist und verschiedene Informationen (Telefonnummer, E-Mail, Adresse) zu jedem S. aufgelistet sind, nach einem Schüler suchen." L. reiht DVDs (mit unsortierten Titeln) auf dem Tisch auf . L.: „W. such bitte den Film „Forrest Gump" und A. such bitte den Film „Die Akte". Zählt dabei die Anzahl eurer Schritte/Vergleiche. Erläutert uns anschließend eure jeweilige Vorgehensweise." Falls C. & B. anwesend sind, sollen auch sie einen Titel suchen. S. suchen ihren Titel, wobei A. seinen nicht finden wird, da dieser nicht existiert.	beim Suchen)	
8:45 – 8:50	Ergebnis-sicherung 2	S. erläutern dann ihre Vorgehensweise. L. ergänzt durch gezielte Fragen die Vorgehensweise: Wo beginnt die Suche? Was wird verglichen? Wann endet die Suche? → Hinführung zur sequentiellen Suche: S.: „Man fängt bei der ersten DVD an. Stimmt der Titel mit dem gesuchten Titel überein, kann die Suche beendet werden. Andernfalls fährt man mit dem nächsten Titel fort. Die Suche läuft solange, bis der gesuchte Titel gefunden oder bis das Ende der DVD-Sammlung erreicht wurde."	Plenum Schüler-Lehrer-Gespräch	
8:50 – 8:53	Erarbeitung 3	L.: „Dieses Suchverfahren nennt man sequentielle oder auch lineare Suche." L. verteilt das Arbeitsblatt „Suchverfahren" mit dem Arbeitsauftrag in der ersten Aufgabe die allgemeine Vorgehensweise der linearen Suche zu erläutern.	Einzelarbeit	AB Suchver-fahren

8:53 – 8:55	Ergebnis-sicherung 3	Ein S. wird aufgefordert, seine Antwort vorzulesen.	Plenum	
8:55 – 9:00	Erarbeitung 4	L.: „A., denk dir eine Zahl zwischen 0 – 1000. Sage bei jedem Versuch von W., ob die geratene Zahl zu groß oder zu klein ist. W., du hast maximal 10 Versuche, um A.s gedachte Zahl zu ermitteln. Tauscht anschließend eure Rollen. “ Falls C. & B. anwesend sind, sollen auch sie das machen. Falls nur einer von den beiden anwesend ist, rät dieser mit dem L. L.: „Tauscht nun eure Rollen. W. denkt sich eine Zahl und A. ermittelt.“ (Eventuell wiederholen. Falls Strategie nicht erschlossen wurde, macht L. mit)	Partnerarbeit	
9:00 – 9:05	Ergebnis-sicherung 4	L.: „Beschreibt eure Strategie.“ S.: „Man grenzt den Bereich der Zahlen ein, indem man eine Zahl wählt, die in der Mitte der verbleibenden Zahlen liegt. Durch die Aussage „zu groß/zu klein“ fallen die Hälfte der Zahlen weg. Diese Halbierung des jeweiligen Bereichs führt man solange fort, bis man die Zahl gefunden hat.“	Plenum Schüler-Lehrer-Gespräch	
9:05 – 9:08	Erarbeitung 5	L.: „Dieses Suchverfahren nennt man binäre Suche. Beschreibt nun in Aufgabe 2 auf dem AB die allgemeine Vorgehensweise der binären Suche.“	Einzelarbeit	AB Suchver-fahren
9:08 – 9:10	Ergebnis-sicherung 5	Ein S. wird aufgefordert, seine Antwort vorzulesen.	Plenum	

9:10 – 9:20	Erarbeitung 6	S. werden aufgefordert, die DVD's mit Hilfe des Bubblesort-Algorithmus zu sortieren, um anschließend die binäre Suche darauf anzuwenden. S. sortieren und suchen wieder ihre beiden Titel. Dabei zählen Sie wieder ihre Schritte/Vergleiche.	Einzelarbeit	DVD's
9:20 – 9:30	Ergebnis-sicherung 6	S. berichten, wie viele Schritte sie benötigt haben → Vergleich zur linearen Suche. L.: „Welche Vorteile und Nachteile haben nun die beiden Suchverfahren?“ S.: „Geschwindigkeit der binären Suche kann schneller sein als bei der linearen Suche. Vor allem dann, wenn das Element nicht in der Liste vorhanden ist. Die lineare Suche benötigt jedoch nicht die Voraussetzung, dass die Daten sortiert sein müssen.“ L.: „Haltet diese Vor- und Nachteile in Aufgabe 3 fest.“	Plenum Schüler-Lehrer-Gespräch	AB Suchver-fahren

6. Darstellung der durchgeführten Tests

Ich habe die Schüler insgesamt zwei Tests schreiben lassen und diese auch benotet. Die beiden Tests sind im Anhang 12.3 enthalten und bestehen beide aus zwei A4 Seiten.

Der erste Test ist zum Thema „Einführung in die Programmierung“ und dient der Erfassung der (notwendigen) Vorkenntnisse. Aufgrund der geringen Schüleranzahl brauchte ich keine Maßnahmen, wie z.B. das Erstellen von zwei Varianten des Tests (Gruppe A und B), gegen Täuschungsversuche vornehmen.

In der ersten Aufgabe sollen die Schüler auswählen, welche gegebenen Definitionen und Eigenschaften zum Algorithmusbegriff gehören (2 Punkte). In Aufgabe 2 werden die in Python zulässigen Bezeichner ausgewählt (6 Punkte). Dafür müssen die Schüler wissen, dass keine Schlüsselwörter wie „if“ zulässig sind und ein Bezeichner in Python stets mit einem Buchstaben (des englischen Alphabets) oder einem Unterstrich beginnt und anschließend nur Buchstaben, Unterstriche oder Zahlen folgen. Die dritte Aufgabe besteht aus einer zu füllenden Tabelle (8 Punkte), in der eine Spalte mit Python-Ausdrücken gegeben ist. Die dazugehörige Ausgabe, sowie eine kurze Erklärung zu jeder Eingabezeile, ist gesucht. Beispiel: Eingabe: `20 // 6`, Ausgabe: `3`, Erklärung: ganzzahlige Division. Um die Aufgabe lösen zu können, sollten die Schüler arithmetische und logische Ausdrücke bzw. die Operatoren kennen.

Die beiden Multiple Choice Aufgaben, zur Definition vom Algorithmusbegriff und zu gültigen Bezeichnern in Python, würden ausreichen, um den Test zu bestehen. Löst ein Schüler die drei ersten Aufgaben komplett mit Erfolg, so hat er 16 Punkte und die Note 2 erreicht. Um eine bessere Note zu erhalten, muss der Schüler die letzte Aufgabe auch lösen. In dieser Aufgabe ist ein Programm gegeben, in dem sieben Fehler enthalten sind. Der Schüler soll mindestens vier Fehler finden und korrigieren. Er erhält pro gefundenen Fehler 0,5 Punkte und pro Korrektur weitere 0,5 Punkte (maximal 4 Punkte). Dafür muss der Schüler beispielsweise wissen, dass bei der Eingabe von `stunden = input("Stunden: ")` die Stundenzahl als Zeichenketten eingelesen wird. Die Korrektur wäre also `stunden = eval(input("Stunden: "))`.

Der Anforderungsbereich steigt von Aufgabe zu Aufgabe, sodass der Schwierigkeitsgrad der letzten Aufgabe am höchsten ist.

Dieser Test wurde unangekündigt am Freitag, den 20.02.15, von A. und W. von 8:00 Uhr bis 8:15 Uhr geschrieben. C. und B. waren nicht anwesend. A. hat 11 von 20 Punkten erreicht und somit eine 4+ als Note erhalten. Seine größten Schwierigkeiten hatte er in Aufgabe 4, wo er keine Punkte erhalten hat. W. hat mit 17 von 20 Punkten eine 2+ als Note bekommen. In Aufgabe 2 fehlten ihr drei Punkte, weil sie nicht alle gültigen Bezeichner ausgewählt hatte.

Beim zweiten Test war die Struktur mit 4 Aufgaben und 20 Punkten ähnlich dem ersten Test. Die erste Aufgabe ist ebenfalls eine Multiple Choice Aufgabe zu den Voraussetzungen und dem Vorteil der binären Suche gegenüber der linearen Suche (2 Punkte). In Aufgabe 2 soll nun die in den letzten drei Unterrichtsblöcken häufig thematisierte Vorgehensweise des optimierten Bubblesort Algorithmus beschrieben werden (5 Punkte), bevor er dann an einer Belegungstabelle Anwendung findet, indem die Buchstaben b|A|a|C|B sortiert werden sollen (5 Punkte). Die Schüler müssen dafür wissen, dass die lexikographische Sortierung zur Folge hat, dass die Großbuchstaben vor den Kleinbuchstaben positioniert werden und somit in der letzten Zeile A|B|C|a|b stehen müssten. Für das Zählen der Vergleiche und Vertauschungen in der Belegungstabelle erhält man dann weitere 2 Punkte. In der letzten Aufgabe sind drei Teilaufgaben mit je 2 zu erreichenden Punkten. In der Teilaufgabe 4a) muss der Schüler die Listenoperation `len(a)` kennen. In der zweiten Teilaufgabe ist die Konkatenation von Listenelementen gefordert, wobei hier die Schwierigkeit durch die Angabe der richtigen Indizes erhöht wird. Abschließend sollte der Schüler in der letzten Teilaufgabe eine dreifache Kopie einer gegebenen Liste, sowie eine anschließende Konkatenation mit einer weiteren Liste durchführen und angeben.

Diesen Test zu Sortier- und Suchalgorithmen haben diesmal A., W. und C. von 8:00 - 8:30 Uhr am Freitag, den 20.03.2015, geschrieben. Vor allem aufgrund der Belegungstabelle habe ich die Zeit von 15 Minuten (beim ersten Test) auf 30 Minuten erhöht. Die Belegungstabelle war ebenfalls Grund für die nicht erreichten Punkte der drei Schüler. Besonders die Bemühungen von C. beim Lösen der Testaufgaben sind mir (trotz

„schlechtester“ Note) positiv aufgefallen. Die Noten waren wie folgt: A. hat eine 2-, W. eine 3+ und C. eine 3- erhalten. Entgegen meinen Erwartungen hatte die sonst leistungsstärkste W. die größten Schwierigkeiten beim Ausfüllen der Belegungstabelle, weil sie nicht für jeden Vergleich eine Tabellenzeile verwendet hatte, sondern nur für jede Vertauschung. Doch auch die beiden anderen Schüler hatten ähnliche Schwierigkeiten beim Ausfüllen der Tabelle.

Da A. sich von einer 4+ im ersten Test zu eine 2- im zweiten Test verbessert hat, scheint das Anforderungsniveau der Tests adäquat gewesen zu sein. Der Leistungsabfall bei W. von einer 2+ zu einer 3+ zeigt mir auch, wie anspruchsvoll die Tests inhaltlich waren.

Die Erreichung des Hauptziels meiner Unterrichtseinheit (siehe 3.1) kann in einem solchen Test nicht vollständig überprüft werden. Doch ich kann mit Sicherheit sagen, dass alle drei Schüler die Vorgehensweise des Bubblesort Algorithmus beschreiben können (Aufgabe 2 Test 2) und diesen auch anwenden können (Aufgabe 3). Außerdem haben sie mindestens eine Vorstellung davon wie die Herangehensweise (mit möglichen unterstützenden Techniken) an die Implementierung eines Algorithmus ist.

.

7. Hospitation

Am 18.03.2015 durfte ich im Unterricht von meinem Kommilitonen S. hospitieren. Er machte zeitgleich ein Praktikum am K.-Gymnasium in Berlin. Dieses Gymnasium bietet „ein breites Angebot, das über ihr Profil als Schule mit besonderer Betonung der Mathematik, den Naturwissenschaften und der Informatik deutlich hinausgeht.“[62] In seiner Unterrichtsstunde waren 16 Schüler aus drei verschiedenen Informatik-Wahlpflichtkursen der 9. Klassenstufe anwesend. Darunter waren auch Schüler, die er nicht kannte. Daher hatte S. erschwerte Bedingungen eine Bedingungsanalyse zu erstellen und, wie man glauben würde, eine innere Differenzierung z.B. „zur Herstellung arbeitsfähiger Teilgruppen“[63] vorzunehmen.

Mit Hilfe von den drei leicht modifizierten Beobachtungsbogen von Hilbert Meyer habe ich einen differenzierenden Blick auf S.s Unterricht bekommen. Ich habe die Bogen nach dem 90 minütigen Unterrichtsblock ausgefüllt und mir während des Unterrichts Notizen gemacht. Wie auch schon H. Meyer möchte ich betonen, dass meine Aussagen und zusammenfassenden Einschätzungen zu S.s Unterricht subjektive Urteile auf der Grundlage meiner Interpretation der Stunde sind.[64] Im Anhang sind die ausgefüllten Bogen gegeben, welche ich nun kommentieren werde.

S. hat sich an den von H. Meyer[65] beschriebenen methodischen Grundrhythmus gehalten. Seine Unterrichtsstunde gliedert sich gemäß dem Dreischritt in Einstieg, Erarbeitung und Ergebnissicherung. Angelehnt an den Grundrhythmus der Freiarbeit[66] gab er den Schülern im Einstieg eine Orientierung über die Stundenplanung und verteilte Aufgabenstellungen. Er ließ die Schüler dann selbstständig eine Wandzeitung erarbeiten und anschließend in Kleingruppen präsentieren und im Plenum reflektieren. Als Sozialform wählte er dabei die

---

62 http://kaethe-kollwitz-gymnasium.de/kkos15/ [letzter Zugriff: 29.05.2015]

63 MEYER: *Leitfaden Unterrichtsvorbereitung,* Seite 72.

64 MEYER: *Leitfaden Unterrichtsvorbereitung,* Seite 236.

65 MEYER: *Was ist guter Unterricht?,* Seite 27.

66 MEYER: *Leitfaden Unterrichtsvorbereitung,* Seite 73.

Gruppenarbeit in der Erarbeitungsphase und in der Ergebnissicherung. Den Plenumsunterricht nutze er als Einstieg sowie als zweite Ergebnissicherung. Seine gewählten Handlungsmuster waren das Unterrichtsgespräch beim Plenumsunterricht und eine themenzentrierte Selbstdarstellung in Form von einer Wandzeitung bei der Erarbeitungsphase. In der ersten Ergebnissicherung wurde dann ein sogenannter Gallery Walk durchgeführt, wobei die Gruppenkonstellation sich (wie beim Gruppenpuzzle) änderte, sodass sich in jeder neuen Gruppe mindestens ein Schüler aus jeder alten Gruppe befand. Seine beiden im Kurzentwurf genannten Lernziele: „S. erstellen eine Wandzeitung zu einem Themenvorschlag „Roboter im Alltag“.“ und „S. führen einen Gallery Walk durch.“ sind eher als Aufgabenstellungen formuliert. Zutreffender wäre meiner Meinung nach zum Beispiel das Lernziel „Die S. sollen verschiedene Einsatzmöglichkeiten von Robotern im Alltag nennen können.“

S. hatte keinerlei Schwierigkeiten mit den ihm unbekannten Schülern umzugehen. Auch für eine hohe innere Differenzierung konnte er sorgen, indem er die Schüler in der Einstiegsphase selbst entscheiden ließ, welches Thema sie jeweils bearbeiten wollten. So konnten sich leistungsstarke Schüler auch individuell mit schwierigen Inhalten vertieft auseinandersetzen. Die Methodenwahl passte sehr gut zu den Inhalten. Zur Auswahl waren die Themen: Roboter im Weltraum, Roboter in der Medizin, Roboter im Alltag, Roboter in der Industrie und Roboter im Straßenverkehr. Je nach Interesse haben die Schüler ein Thema gewählt und in der Gruppe selbstständig bearbeitet. Als Beobachter merkte man, dass die Schüler beim selbstregulierten Lernen interessiert arbeiteten und motiviert mitmachten. Sie halfen sich auch meist untereinander und beteiligten sich überwiegend an Diskussionen.

Der Arbeitsauftrag war bei allen Gruppen derselbe: „Informiert euch über das Thema. Verwendet als Einstieg die unten genannten Quellenvorschläge. Konzentriert euch auf die inhaltlichen Schwerpunkte. Haltet eure Ergebnisse auf einer Wandzeitung fest.“ Die Arbeitsaufträge waren nicht für alle Schüler verständlich. Die meisten Schüler kannten eine sogenannte Wandzeitung nicht und suchten erst mal im Internet danach. Mein Vorschlag wäre, einfach ein Beispiel einer Wandzeitung mitbringen und zeigen. Zwei bis drei Internet-Links wurden pro Arbeitsblatt als Quellenvorschläge angegeben, welche jedoch Zeichen für Zeichen von den Schülern abgetippt werden mussten, da die Links nur auf den

Blättern gegeben waren. Die triviale Lösung wäre es, den Schülern das Blatt oder zumindest die Linkliste als PDF Dokument bereitzustellen, sodass sie den Link sofort anklicken können. Ein weitaus größeres Manko war die nicht für einen Gallery Walk vorbereitete Umgebung. S. sollte sich in der Vorbereitung bereits überlegen, wo die Plakate positioniert und wie diese (mit Tesafilm, Magneten o.ä.) angebracht werden sollen. Damit vermeidet man unnötige Unruhen bei der Übergangsphase und sorgt dafür, dass alle Gruppenteilnehmer das jeweilige Plakat von vorne lesen können.

Hilbert Meyer berichtet über Forschungsergebnissen zum Gruppenunterricht: „Das größte Hindernis für erfolgreichen Gruppenunterricht ist die Belehrungs- und Kontrollsucht der Lehrer. [...] Gerade dort, wo der Lehrer nicht in die Gruppenprozesse eingreift, lassen sich größere Lernerfolge nachweisen.“[67] Aus diesem Grund war das ständige Laufen von S. von Gruppe zu Gruppe und das Eingreifen in den Prozess eventuell etwas nachteilig. Jedoch konnte er dadurch auch feststellen, wie weit die jeweiligen Gruppenarbeiten fortgeschritten waren. So passte er seine Zeitplanung dem Leistungsvermögen der Schüler entsprechend an. Er verlangsamte das Unterrichtstempo bei der Erarbeitung und zog das Tempo an, als die Schüler bei der Ergebnissicherung nicht so viel Zeit wie geplant benötigten.

Alles in allem war es eine erfolgreiche schülerzentrierte Unterrichtsstunde, bei der neben fachlichem Wissen auch vor allem Methoden- und Sozialkompetenzen an die Schüler vermittelt wurden.

---

[67] MEYER: *Was ist guter Unterricht?,* Seite 81.

8. Zusammenfassung der fachdidaktisch-methodischen Erfahrungen und theoretischen Einsichten

Bei meiner Unterrichtseinheit vom 06.03.15 waren, wie schon erwartet, nur A. und W. anwesend. Da die beiden Schüler das nötige Vorwissen haben und zu den leistungsstärkeren Schülern gehörten, konnten manche Unterrichtsphasen, wie z.B. die fragend-entwickelnde Einstiegsphase anhand der Animation, schneller als geplant durchgeführt werden. In der dritten Erarbeitungsphase sehe ich nun im Nachhinein einige optimierbare „Schwachstellen". Beispielsweise haben die Schüler sich wenig über die enaktive Phase gefreut, was ich eigentlich erwartet hatte, als ich die Spielkarten zum Sortieren einplante. Hinzu kommt, dass die Vermittlung von neuem Wissen über „Listen in Python" vom roten Faden der Stunde abgewichen ist und somit nicht in die handlungsorientierte Unterrichtsstunde über Bubblesort passte. Die dritte Erarbeitungsphase beinhaltete auch das Erstellen des Pseudocodes im Lehrer-Schüler-Gespräch an der Tafel. Rückblickend ist die Tafelarbeit mit einem direkten Schüler-Lehrer-Gespräch eine sehr hohe Anforderung gewesen. Die Schüler hatten nicht viel Zeit zu überlegen, mussten mir direkt antworten und hatten eventuell Scheu falsche Antworten zu geben.[68] Die darauffolgende Partnerarbeit bei der Erstellung und Prüfung der Struktogramme ist bei der dazugehörigen dritten Ergebnissicherung gut aufgenommen und durchgeführt worden. Die Sozialkompetenzen der beiden Schüler wurden, wie im Hauptziel der Unterrichtseinheit formuliert, in Hinblick auf angemessene und respektvolle Kritikäußerungen geschult.

In der zweiten Unterrichtsstunde (am 13.03.15) erschien erfreulicherweise C. das erste Mal nach zwei Monaten wieder im Unterricht. Er traute sich selbst wenig zu und dachte er sei störend bzw. würde den Unterricht nur unnötig aufhalten. Jedoch hatte ich (ohne wirklich

[68] Dazu mehr im Alternativentwurf.

damit zu rechnen) für den Fall, dass C. und B. kommen würden, alles vorbereitet, sodass sie an jeder Phase teilnehmen hätten können. Der erste Dreischritt (Einstieg 1, Erarbeitung 1 und Ergebnissicherung 1) war zum Thema „Listen in Python". Das Arbeitsblatt war so aufbereitet, dass auch C. ohne jedwede Programmiererfahrung die Ausdrücke in der IDLE eingeben und die Ausgaben sowie Ansätze einer Erläuterung aufschreiben konnte. In der zweiten Erarbeitungsphase haben dann A. und W. die vorherige Stunde fortgesetzt, indem sie mit Hilfe ihres Struktogramms den Bubblesort Algorithmus in Python implementiert haben. Für den Fall, dass einer der beiden schneller fertig ist, hatte ich auf einem Arbeitsblatt zusätzliche Teilaufgaben als Ergänzung gegeben. C., der weder den Algorithmus, noch die Programmiersprache Python kannte, hat parallel dazu das Arbeitsblatt und die beiden Animationen der letzten Stunde erhalten. So konnte er, wie zuvor W. und A., die Vorgehensweise des Algorithmus anhand der Animation beschreiben, überflüssige Schritte ausfindig machen und den optimierten Bubblesort Algorithmus so entwickeln. Den Pseudocode sind wir gemeinsam durchgegangen. Mit Hilfe des Arbeitsblattes zu Struktogrammen von Frau L. (siehe Anhang 12.1.5) hat C. es geschafft, ein Struktogramm anhand des Pseudocodes für Bubblesort zu erstellen. Die Stunde endete mit der Ergebnissicherung 2, in der dann A. sein Programm (ohne die zusätzlichen Aufgaben) am Beamer vorstellte, mehrere Beispiel-Listen sortierte und wir im Plenum offene Fragen klärten. Im Anschluss stellte ich noch Kontrollfragen zum Programm und zum Algorithmus an alle drei Schüler.

Da dieser Grundkurs keine schriftlichen oder mündlichen Abiturprüfungen im Fach Informatik haben wird, ist es durchaus zulässig, das Hauptaugenmerk daraufzulegen, dass alle Schüler individuell gefördert werden. Ich kann mir jedoch vorstellen, dass in einem Kurs, in dem auch Schüler sind, die Abiturprüfungen schreiben werden, der fehlende Inhalt selbstständig nachgeholt werden muss und man als Lehrer gezwungen ist, gewisse Themen in einer vorgegebenen Zeit „abzuarbeiten".

In der dritten und letzten Stunde waren A. und C. anwesend. W. fehlte entschuldigt. Ich hatte mich bereits beim Planen schon darauf gefreut, diese Stunde zu halten. Denn die Methodenvielfalt, die diesmal vor allem durch Knobelaufgaben und enaktive Phasen geprägt war, würde den Schülern meiner Meinung nach viel Spaß bereiten. Und so war es auch. Die Stunde zu Suchalgorithmen hat allen Beteiligten sehr gut gefallen und verlief

besser als geplant. C. setze sich zu A. an den PC und sie setzten gemeinsam die zusätzlichen Modifikationsaufgaben zum Bubblesort Programm um. A. hat ausführlich und mit viel Geduld seinem Mitschüler erklärt, warum und wie er welche Ausgabe so programmiert. Vor allem bei der Erarbeitungsphase 4 mit der Aufgabe „Denk dir eine Zahl zwischen 0 – 1000..." für die Hinführung zur binären Suche hat die Schüler der Ehrgeiz gepackt. Nach insgesamt sieben Durchläufen haben sie es geschafft die Zahl des anderen zu erraten und so die Vorgehensweise des Algorithmus zu beschreiben.

Die Vorbereitungen auf meinen Unterricht haben zugegebenermaßen mehr Zeit in Anspruch genommen, als ich anfangs erwartet hatte. Doch wenn man erst einmal die Arbeitsblätter, Lösungsvorschläge, Konzepte, Materialien und Spickzettel erstellt und zusammengetragen hat, fühlt man sich bereit und selbstsicher auch auf unerwartete Situationen zu reagieren. Außerdem wollte ich unbedingt die Motivation der Schüler steigern und Spaß am Fach vermitteln, sodass mir das allgegenwärtige Überlegen, Ideenfinden und Recherchieren auch außerhalb der Praktikumszeit begleitet hat. Die Theorie, welche man in der Universität lernt, scheint mir nun nur ein Bruchteil dessen zu sein, was ich im Praktikum an Praxis erlernt habe. Hauptsächlich konnte ich mein Wissen aus den Didaktik-Kursen anwenden. Als Lehramtsstudent würde ich mir von Seiten der Universität mehr Kurse zu schulischen Fachthemen aus dem RLP wünschen, wobei in Informatik noch eher nötiges Schulwissen in den Vorlesungen vermittelt wird, als in meinem Erstfach Mathematik.

Alles in allem hat mir mein Praktikum an der A.-F.-Schule und meine Betreuung durch Frau L. sehr gut gefallen. Meine eigene Freude am Unterrichten und das gute Feedback von meinen Schülern und meiner Betreuerin hat mich darin bestärkt, die richtige Berufswahl getroffen zu haben. Frau L., mit der ich weiterhin in Kontakt bleibe, hat mir berichtet, dass C. seit meinem Praktikum weiterhin regelmäßig zum Unterricht erscheint. A. wird das Fach Informatik als 5. Prüfungskomponente wählen. W. hat ihre Abiturprüfungen zuversichtlich geschrieben und somit keinen Unterricht mehr an der Schule. Von B. ist zu erwarten, dass er die Schule offiziell abbricht, da er seit Januar nicht mehr regelmäßig die Schule besucht hat. Frau L. hat mir Anfang Mai berichtet: „Wir haben mit Arduino angefangen. Die ersten Schaltungen haben erfolgreich eine LED blinken lassen."

9. Abbildungsverzeichnis

10. Tabellenverzeichnis

11. Literaturverzeichnis

Abeck, Sebastian: *Kursbuch Informatik I: Formale Grundlagen der Informatik und Programmierkonzepte am Beispiel von Java,* Universitätsverlag Karlsruhe, 2005.

Astrachan, Owen: *Bubble Sort: An Archaeological Algorithmic Analysis,* Duke University, 2003.

Häberlein, T.: *Praktische Algorithmik mit Python,* Oldenbourg Wissenschaftsverlag, 2012.

Hartmann, W./ Näf, M./ Reichert, R.: *Informatikunterricht planen und durchführen,* Springer-Verlag Berlin Heidelberg, 2006.

Hattenhauer, Rainer: *Informatik für Schule und Ausbildung,* Pearson Schule, 2010.

Heiderich, N./ Meyer, W.: *Technische Probleme lösen mit C/C++: von der Analyse bis zur Dokumentation*, 2. Auflage, Hanser Verlag, 2014.

Hubwieser, P./Aiglstorfer, G.: *Fundamente der Informatik: Ablaufmodellierung, Algorithmen und Datenstrukturen,* Oldenbourg Wissenschaftsverlag, 2004.

Humbert, Ludger: *Didaktik der Informatik mit praxiserprobtem Unterrichtsmaterial*, 2. Auflage, Teubner Verlag Wiesbaden, 2006.

Lutz, M./Ascher, D.: *Einführung in Python: Moderne OO-Programmierung,* 2. Auflage, O'Reilly Verlag, 2007.

Meyer, Hilbert: *Leitfaden Unterrichtsvorbereitung*, 7. Auflage, Cornelsen Schulverlage, 2014.

Meyer, Hilbert: *Leitfaden zur Unterrichtsvorbereitung*, 12. Auflage, Cornelsen Scriptor, 1993.

Meyer, Hilbert: *Was ist guter Unterricht?*, 7. Auflage, Cornelsen Verlag Scriptor, 2011.

Meyer, Hilbert: *Unterrichts-Methoden II: Praxisband*, 14. Auflage, Cornelsen Verlag Scriptor, 2008.

Perry, Greg: *Jetzt lerne ich Programmieren,* Markt+Technik Verlag, 2002.

Pomberger, G./Dobler, H.: *Algorithmen und Datenstrukturen: einer systematische Einführung in die Programmierung,* Pearson Studium, 2008.

Schoenningh Lehrwerk: *Informatik 2: Modellierung, Datenstrukturen und Algorithmen,* Band 2, 2012.

Schubert, S./ Schwill, A.: *Didaktik der Informatik,* 2. Auflage, Spektrum Verlag Heidelberg, 2011.

Weicker, K./Weicker, N.: *Algorithmen und Datenstrukturen,* Springer Vieweg, 2013.

Weigend, M.: *Python 3: Lernen und professionell anwenden,* 5. Auflage, Hüthig Jehle Rehm Verlag, 2013.

Berliner Rahmenlehrplan der Informatik für die gymnasiale Oberstufe: (1. Auflage 2006) http://www.berlin.de/imperia/md/content/sen-bildung/unterricht/lehrplaene/sek2_informatik.pdf?start&ts=1429785405&file=sek2_informatik.pdf [letzter Zugriff: 28.05.2015].

12. Anhang

12.1 Arbeitsblätter von Frau L.

12.1.1 Arbeitsblatt für die Einführung in Python

Einführung in die Programmiersprache Python

Erklären Sie die Begriffe Hardware und Software. Was bedeutet „Programmieren“?

(EVA-Prinzip: Eingabe – Verarbeitung – Ausgabe)

Es gibt mehrere Hundert Programmiersprachen.[1] Nennen Sie einige Programmiersprachen:

Man unterscheidet je nach Arbeitsweise zwischen **Compilern** und **Interpretern**.

Compiler (Übersetzer): das Quellprogramm wird als Ganzes in Maschinensprache übersetzt, danach wird das Maschinenprogramm ausgeführt.

Interpreter: das Quellprogramm wird zeilenweise analysiert und jede Zeile sofort ausgeführt.

Python ist eine Interpretersprache, die Anfang der 1990er Jahre von Guido van Rossum am *Centrum Wiskunde & Informatica* in Amsterdam mit dem Ziel entworfen wurde, möglichst einfach und übersichtlich zu sein.[2]

Python als Taschenrechner

Geben Sie in der IDLE die folgenden Ausdrücke ein und notieren Sie die Ausgaben:

Eingabe	Ausgabe und Erklärung
41 + 6	
25 - 18	
3 * 7	
3 * 7.0	
25 / 6	
25 // 6	
25 % 6	
2 ** 3	
3 ^ 3	
17 < 20	
17 > 21	
a = 12 a a + 5 a + 5.0	
b = 'b' b b + 5 b + a b + 'a'	

1 http://de.wikipedia.org/wiki/Liste_der_Programmiersprachen

2 http://de.wikipedia.org/wiki/Python_%28Programmiersprache%29

12.1.2 Arbeitsblatt für die Einführung in Python

Ausdrücke:

- besteht aus Operanden und Operatoren, zusammengesetzt nach bestimmten Regeln
- ist eine Verarbeitungsvorschrift, die ein Ergebnis zurück gibt.
- Ausdruck (arithmetisch): 7+(3-4)*2 liefert das Ergebnis „5“ (ganze Zahl)
- Ausdruck (logisch): 17 < 21 liefert das Ergebnis „True“ (Wahrheitswert)

Operatoren:

- Arithmetische Operatoren berechnen Werte aus ein oder zwei Operanden.
- Vergleichsoperatoren vergleichen zwei Werte.
- Logische Operatoren verknüpfen verschiedene Ausdrücke.
- Vergleichsoperatoren: „==“ „<“ „>“ „<=“ „>=“ „!=“
- arithmetische Operatoren
  Priorität der arithmetischen Operatoren in der Reihenfolge ihrer Priorität:

Operator	*Erläuterung*
**	Potenzieren
-, +	Vorzeichen (unär)
*, /, //, %	Multiplikation, Division, ganzzahlige Division (div), Rest (mod)
+, -	Addition, Subtraktion

Bezeichner:

- Namen für Variablen, Funktionen u.ä.
  bei der Wahl der Namen für Variablen, Funktionen usw. ist zu beachten:
- keine Schlüsselwörter verwenden!
  Schlüsselwörter in Python 3:

False	as	continue	else	for	import	nonlocal	raise	with
None	assert	def	except	from	in	not	return	yield
True	break	del	exec	global	is	or	try	
and	class	elif	finally	if	lambda	pass	while	

- erlaubt sind nur Buchstaben des englischen Alphabets, Ziffern oder Unterstrich
- unterscheidet zwischen großen und kleinen Buchstaben!
- erstes Zeichen eines Bezeichners muss ein (großer oder kleiner) Buchstabe des (englischen) Alphabets oder ein Unterstrich sein ('_').
- der Rest des Bezeichners kann aus (großen oder kleinen englischen) Buchstaben, Unterstrichen ('_') oder Ziffern (0-9) bestehen

zulässig	summe	Kaese	i	a23b7	name	__mein_name
unzulässig	1summe	käse	if	4+3	name-1	mein name

- wichtig: sogenannte „sprechende“ Bezeichner verwenden!
  (summe, summe_1, anfangszeit, laenge, tag, stunde1,...)

Wertzuweisung:

a = 22 Der Wert der Variable auf der linken Seite des Operators wird verändert, der Variable a wird der ganzzahlige Wert 22 zugewiesen.

Vergleich:

a == 22 Die Werte der Ausdrücke links und rechts vom Operator werden verglichen; das Ergebnis des Vergleichs ist True (bei Gleichheit) oder
False (bei Ungleichheit)

12.1.3 Arbeitsblatt zu Ein- und Ausgaben in Python

Eingaben:

a = input()	Der über die Tastatur eingegebene Wert wird der Variable a zugewiesen **Achtung**: Ziffern werden als Zeichen gelesen und müssen noch in Zahlen umgewandelt werden! (siehe unten, Typumwandlungen)
a = input(„a= ")	Die Ausgabe „a= " erscheint auf dem Bildschirm, das ist benutzerfreundlicher.
a = eval(input(„a= "))	Der eingegebene Wert wird ausgewertet, statt der Zeichenkette '2' wird die Zahl 2 zugewiesen

Ausgaben:

print(a)	Ausgabe des Wertes der Variable a
print(„Zahl ist ", a)	Verketten von Ausgaben
print(a, end =")	kein Zeilenumbruch nach der Ausgabe von a

Kommentare:

- Zu einem guten Programmierstil gehört, dass ein Programmtext kommentiert wird.
  (*Eagleson's Law*. dies sagt aus, dass jeder selbstgeschriebene Code, den man mehr als sechs Monate nicht betrachtet hat, genauso aussieht, als hätte ihn jemand anderes geschrieben.)
- Ein Kommentar beginnt mit dem Zeichen „#"
- Kommentar wird nicht ausgeführt, dient der besseren Lesbarkeit des Quellcodes
- Kommentar beginnt entweder am Zeilenanfang oder nach einer Anweisung und gilt bis zum Zeilenende
- wenn Kommentar deutsche Sonderzeichen enthalten soll, muss die erste Programmzeile (je nach verwendeter Codierung) lauten:

```
# -*- coding: iso-8859-1 -*-
```

bzw.

```
# -*- coding: utf-8 -*-
```

Beispiel für einen Programmkopf:

```
# -*- coding: iso-8859-1 -*-
#-------------------------------------------------
# Dateiname: Datumsrechner.py
#
# Version: 2.4
# Beschreibung: Berechnung der Schaltjahre eingefügt
# Wilhelm Busch, 20.03.2011
#-------------------------------------------------
```

Typumwandlungen:

- Eingabe einer Zeichenfolge z.B. „23" und Zuweisung zur Variable „a": a=input("a=")
- Umwandlung von „23" in ganze Zahl 23: x = int(a)
- Umwandlung von „23" in Dezimalzahl 23.0: y = float(a)

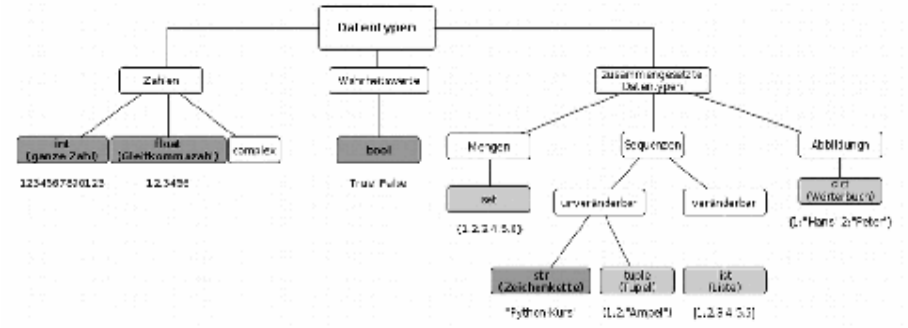
Aufgaben:

1. Schreiben Sie ein Programm, das Ihren Namen abfragt und eine Begrüßung ausgibt.
2. Schreiben Sie ein Programm, das die Seitenlängen eines Rechtecks abfragt und die Fläche ausgibt.
3. Schreiben Sie ein Programm, das eine Sekundenzahl in Stunden, Minuten und Sekunden umrechnet.
4. Schreiben Sie ein Programm, das zwei Zeiten (stunden, Minuten und Sekunden) abfragt, die Zeiten addiert und das Ergebnis wieder in Stunden, Minuten und Sekunden ausgibt.
5. Führen Sie die folgenden Berechnungen bzw. Typumwandlungen durch und schließen Sie auf die Genauigkeit der Zahlendarstellungen:
   int(1e30)
   float(123456789**10)
   123456789**10

12.1.4 Arbeitsblatt zu Datentypen

Wichtige Datentypen in Python3

- Python ist eine *objektorientierte* Sprache, Daten werden durch *Objekte* repräsentiert.
- Jedes Objekt besitzt eine Identität, einen Wert und einen Typ.
- *Identität*: dient der Identifizierung eines Objekts und wird durch eine ganze Zahl repräsentiert
  Abfrage mit *id(Objektname)*
- *Wert*: „Inhalt" des Objekts, z.B. „123", „Python-Kurs", „1.5"
- *Typ*: bestimmt, wie das Objekt im Python-Skript verarbeitet wird (z.B. welche Operationen erlaubt sind)
  Abfrage des Typs einer Variable oder eines Ausdrucks mit
  type(Variablenname) oder *type(Ausdruck)*



Ganze Zahlen

- können beliebig lang sein
- Darstellung in anderen Zahlensystemen:

	binär	oktal	dezimal	hexadezimal
Eingabe	`a = 0b10110`	`a = 0o1235`	`a = 234`	`a = 0x2F`
a (dezimal)	22	669	234	47

Gleitkommazahlen

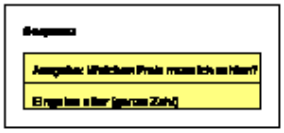
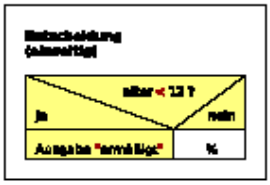
- Darstellung mit Dezimalpunkt (kein Komma!)
  `1.234`
- Null vor dem Dezimalpunkt kann man weglassen
  `.234` `0.01223`
- bei sehr großen und sehr kleinen Zahlen wird die Exponentialschreibweise verwendet
  `5.4e-07` entspricht `0.00000054`
  `1e2` entspricht `100`
  `.234e6` entspricht `234000.0`
  `17e0` entspricht `17.0`
- Genauigkeit der internen Darstellung ist auf eine feste Anzahl von Stellen begrenzt
  Gibt man eine längere Ziffernfolge ein, so werden die letzten Stellen abgetrennt!
  Beispiel: aus `1.2345678901234567890` wird `1.2345678901234567`

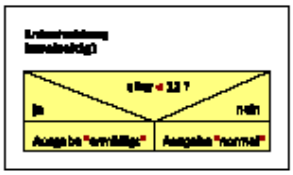
Zeichenketten (`Strings`)

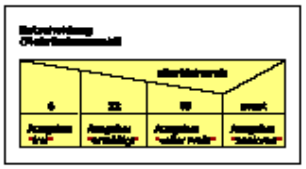
- bestehen aus einzelnen Zeichen, eingeschlossen in Hochkomma oder Anführungszeichen (`„Py"`, `'Py'`)
- `str(123)` wandelt die Zahl 123 um in die Zeichenkette „123"
- `chr(65)` gibt das Zeichen mit dem ASCII-Code 65 aus („A")
- `ord(„A")` gibt den ASCII-Code des Zeichens „A" aus (65)

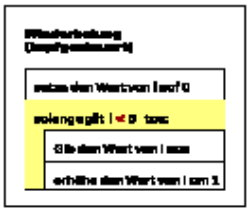
12.1.5 Arbeitsblatt zu Struktogrammen

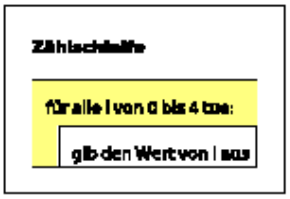
Kontrollstrukturen (Nassi-Shneiderman-Diagramm)

Erläuterung	Programmbeispiel	Struktogramm
alle Anweisungen werden nacheinander ausgeführt	`print('Welchen Preis muss ich zahlen?')` `alter = eval(input('wie alt: '))`	
Die Anweisungen werden nur unter bestimmten Bedingungen ausgeführt.	`if alter < 12:` `  print('ermäßigt')`	

Es sind Anweisungen vorhanden, die ausgeführt werden, wenn die Bedingung erfüllt ist und andere, wenn die Bedingung nicht erfüllt ist.	`if alter < 12:`
	`  print('ermäßigt')`
	`else:`
	`  print('normal')`
		

Es gibt mehr als 2 Fälle;
für jeden Fall gibt es gesonderte Anweisungen.
 	
Die Mehrfachauswahl vereinfacht geschachtelte Entscheidungen.
	`if alter < 6:`
	`  print('frei')`
	`elif alter < 12:`
	`  print('ermäßigt')`
	`elif alter < 65:`
	`  print('voller Preis')`
	`else:`
	`  print('Senioren')`
		

Die Schleife wird so lange durchlaufen, bis eine Abbruchbedingung erfüllt wird. Diese wird am Anfang der Schleife geprüft.
 	
Solange die Bedingung erfüllt ist, führe die Aktion aus.
	`i = 0`
	
	`while i < 5:`
	`  print(i)`
	`  i = i + 1`
		

Die Schleife wird so lange durchlaufen, bis eine Abbruchbedingung erfüllt wird. Diese wird am Anfang der Schleife geprüft. Zähle von Min bis Max.	`for i in range(5):` `  print(i)`	

12.1.6 Arbeitsblatt mit Übungsaufgaben

wichtig:

- jede Anweisung in einzelne Zeile , Kommentar am Zeilenende ist möglich
- Blöcke werden durch *einheitliche Einrückung* gekennzeichnet! (Tabulator verwenden)
- *Doppelpunkt* nach `if, else, elif, while` nicht vergessen!

Aufgaben:

Lösen Sie die folgenden Probleme zuerst mit Hilfe eines Struktogramms und setzen Sie den Algorithmus danach in ein Python-Programm um.

6 Nach der Eingabe zweier Zahlen gibt das Programm die Ergebnisse der ganzzahligen Division sowie den Rest bei der ganzzahligen Division aus.

7 Nach der Eingabe einer Zeichenkette wird diese
a) 5 mal nebeneinander ausgegeben
b) 3 mal nebeneinander ausgegeben, dazwischen Sternchen

8 Beim Schlussverkauf werden Rabatte von bis zu 70% auf den früheren Preis gewährt.
a) Nach Eingabe des alten Preises und des Rabatts ist der neue Preis auszugeben.
b) Nach Eingabe des alten und neuen Preises wird der Rabatt berechnet und ausgegeben.

9 Der Notendurchschnitt einer Klassenarbeit (Noten von 1 bis 6) ist zu berechnen und auszugeben.
Überlegen Sie anfangs genau, welche Eingaben zu tätigen sind.

10 Eine ganze Zahl ist einzugeben und die Quadrate der Zahlen von 1 bis zu dieser Zahl sind zu berechnen und auszugeben.

11 Eine ganze Zahl ist einzugeben, ausgegeben werden
a) alle geraden Zahlen zwischen 1 und der eingegebenen Zahl
b) alle Teiler dieser Zahl, die größer als 1 sind

12 Eine Zahl ist einzugeben, ausgegeben wird das kleine (bzw. das große) Einmaleins dieser Zahl.
(z.B. Zahl sei 9: 1*9=9 usw.)

13 Numerische Ausgaben kann man in Tabellenform darstellen. Die Anweisung

```
print ("%3d * 7 = %6d"  %  (i,i*7.0))
```

gibt eine Zeichenkette aus, die zwei Variablen enthält .Der Wert der ersten Variable wird mit einer Breite von 3 Zeichen ausgegeben und der Wert der zweiten Variablen mit einer Breite von 7 Zeichen. Die beiden Variablen werden den Angaben in der runden Klammer entnommen.
Stellen Sie das Einmaleins aus Aufgabe 11 in Tabellenform dar.

14 Nach Eingabe einer ganzen Zahl sollen alle Primzahlen ausgegeben werden, die kleiner als diese Zahl sind.

15 Es werden zwei Zahlen eingegeben und das Programm gibt aus, welche der Zahlen die Größere bzw. die Kleinere ist.
Was geschieht, wenn beide Zahlen gleich groß sind?

16 Es werden drei Zahlen eingegeben und das Programm gibt aus, welche der Zahlen die Größte bzw. die Kleinste ist.
Was geschieht, wenn zwei Zahlen gleich groß sind?

12.1.7 Arbeitsblatt zu Funktionen in Python

Unterprogramme (Funktionen)

- Unterprogramme können vom Aufrufer Werte übergeben bekommen.
- Unterprogramme können einen Wert an das aufrufende Programm *zurückgeben*.
- Wenn ein Unterprogramm einen Wert an das aufrufende Programm zurück gibt, wird das Unterprogramm als *Funktion* bezeichnet.
- Wenn ein Unterprogramm keinen Wert an das aufrufende Programm zurück gibt, wird das Unterprogramm als *Prozedur* bezeichnet.
- Jedes Unterprogramm hat einen eindeutigen *Namen*. Mit Hilfe dieses Namens wird die Funktion *aufgerufen*.

Vorteile bei der Verwendung von Funktionen

- Die Aufgabenstellung wird in kleinere unabhängige Module eingeteilt und somit auch strukturiert
- Der Quellcode lässt sich besser lesen.
- Der Code einer Funktion kann in anderen Programmen wiederverwendet werden
- Fehler lassen sich schneller finden, weil der Code nur an einer Stelle bearbeitet werden muss
- Veränderungen lassen sich einfacher vornehmen und testen, weil nur Codefragmente betroffen sind

Ablauf

- Die Funktion wird mit ihrem Namen aufgerufen. Der Aufrufer weiß nur die Bezeichnung der Funktion sowie Anzahl, Reihenfolge und Bedeutung der Argumente.
- Nach dem Aufruf löst die Funktion mit Hilfe von Anweisungen die gestellte Aufgabe. Der Entwickler weiß, wie das Problem gelöst wird. Der Aufrufer muss es nicht wissen.
- An den Aufrufer kann die Funktion am Ende einen Wert zurückgeben. Standardmäßig gibt eine Funktion « None » zurück.

Funktionen bestehen aus einem *Kopf* und einem *Rumpf*.

Funktionskopf

- ist die „Blaupause" für den Aufruf und ist die Schnittstelle nach außen
- Beispiel: `def Eingabe():`
- Eine Funktion wird mit dem Schlüsselwort « def » eingeleitet.
- Dem Schlüsselwort folgt der *Name* der Funktion. Mit Hilfe des Namens wird die Funktion *aufgerufen*.
- Dem Namen folgt die Liste mit den zu übergebenden *Parametern*.

Parameterliste

- Beispiel: `def zinsen(kapital, zinssatz, jahre):`
- wird durch die runden Klammern zusammengefasst
- enthält Platzhalter für an die Funktion zu übergebende Werte
- kann beliebig viele *Parameter* haben
- nutzt als Trennzeichen zwischen den Parametern ein Komma
- ist leer, wenn keine Argumente an die Funktion übergeben werden

Funktionsrumpf

- beschreibt die programmierte Aufgabe und fasst den Code für eine Aktion zusammen
- beginnt nach dem Doppelpunkt im Funktionskopf, der gesamte Block wird eingerückt
- kann auch nur aus der Anweisung « `pass` » bestehen. Dieses Schlüsselwort symbolisiert eine leere Anweisung

Rückgabewert einer Funktion

- Mit Hilfe der Anweisung « `return` » werden Werte an den Aufrufer zurückgegeben und die Funktion beendet.
- Standardmäßig wird « None » zurückgegeben. Die Funktion besitzt keinen Rückgabewert.
- Als *Rückgabewert* kann jeder beliebige Datentyp genutzt werden.
- Der Rückgabewert kann in einer Variablen *gespeichert* werden. Die Variable nimmt den Typ des Rückgabewertes an.
- Rückgabe eines Wertes erfolgt in der Form (s.u.) `f = fakultaet(x)`
- Rückgabe mehrerer Werte erfolgt z.B. über (s.u. Aufg.20) `std,minu,sek = zeiteingabe()`

12.1.8 Arbeitsblatt und Aufgaben zu Funktionen in Python

Aufruf der Funktion

- Beispiel:

```
zahl = eingabe()
print(zinsen(2000, 4.5, 10))
print(zinsen(mein_kapital, 4.5, dauer))
```

- Eine Funktion wird immer mit ihrem Namen aufgerufen. Falls eine Funktion nicht definiert ist, wird eine Fehlermeldung „NameError" ausgegeben.
- Dem Funktionsnamen folgt die *Argumentliste*. Falls keine Argumente übergeben werden, sind die Klammern leer. Die Anzahl und der Typ der zu übergebenden Argumente ist abhängig von der Parameterliste des Funktionskopfes.

Beispiel:

```
def fakultaet(x):                    # Funktionskopf mit Parameter
    f = 1                            #
    for i in range(1,x):             #
        f = f * i                    #
    f = f * x                        #
    return f                         # Rückgabe des Ergebnisses

print('n! = ',fakultaet(n))          # Aufruf und Verarbeitung der Funktion
```

Lokale und globale Variablen

- *Eine lokale Variable ist nur in dem Block existent und sichtbar, in dem sie definiert wurde.*
  Sie kann deshalb nur innerhalb dieses Blocks genutzt werden.
- *Globale Variablen* können im gesamten Programm und allen Unterprogrammen genutzt werden.
  Sie können aber durch lokale Variablen mit einer identischen Bezeichnung überdeckt werden. In dem Block sind die globalen Variablen dann nicht sichtbar.
  Auf globale Variablen sollte verzichtet werden!

Docstrings

- unterhalb des Funktionskopfes kann ein Kommentar (Zeichenkette) eingefügt werden
- dieser *Docstring* sollte in *dreifachen* Anführungszeichen stehen und folgende Angaben enthalten:
  - kurze Beschreibung der Funktion
  - Eigenschaften der zu übergebenden Parameter
  - Eigenschaften der zurückgelieferten Objekte
  - Verwendung globaler Variablen?
- Beispiel:

```
def zeitaddition(st1,m1,s1,st2,m2,s2):
    """Addition von zwei Zeiten,

       Eingabe in Stunden, Minuten und Sekunden
       Rückgabe der Zeit in Stunden, Minuten und Sekunden
       """
```

- im interaktiven Modus (IDLE) kann der Docstring sichtbar gemacht werden durch die *help*-Funktion:

```
help(zeitaddition)
```

Aufgaben:

Lösen Sie die folgenden Aufgaben mit Hilfe von geeigneten Funktionen.

17. Schreiben Sie ein Programm, das einen Begrüßungstext mit Hilfe der Prozedur `begruessung()` ausgibt.
18. Schreiben Sie ein Programm, das mit Hilfe der Funktion `zeit_in_sek(h,m,s)` eine Angabe von Stunden, Minuten und Sekunden in eine Sekundenzahl umrechnet und das Ergebnis zurückgibt.
19. Schreiben Sie ein Programm, das mit Hilfe der Funktion `sek_in_zeit(s)` eine Sekundenzahl in Stunden, Minuten und Sekunden umrechnet und zurückgibt.
20. Schreiben Sie ein Programm, das mit Hilfe von geeigneten Funktionen die Aufgabe 4 löst.

Zusatzaufgabe: Verwenden Sie Docstrings zur Beschreibung.

12.2 Materialien zum ausführlichen Stundenentwurf

12.2.1 Erstes Arbeitsblatt: Sortieralgorithmus Bubblesort

Sortieralgorithmus
Bubblesort

1. Aufgabe
a) Beschreiben Sie die Vorgehensweise des Bubblesort Algorithmus anhand der animierten Sortierung durch „BUBBLE-SORT“.

_____
_____
_____
_____
_____

b) Beschreiben Sie den Unterschied der Vorgehensweise des optimierten Bubblesort-Algorithmus zum vorherigen anhand der animierten Sortierung durch „`BUBBLECLEVER-SORT`“.

_____
_____
_____
_____
_____

2. Aufgabe
Erstellen Sie ein Struktogramm für den optimierten Bubblesort Algorithmus.

3. Aufgabe
Tauschen Sie nun mit ihrem Partner die Struktogramme. Erstellen Sie anhand des Struktogramms eine Belegungstabelle für folgende Elemente:

7 2 4 3 9

Zählen Sie dabei die Anzahl der Vertauschungen und Vergleiche.

Beurteilen Sie das Struktogramm ihres Partners. Stellt es den optimierten Bubblesort Algorithmus korrekt dar? Schreiben Sie ggf. Verbesserungsvorschläge in das Struktogramm und besprechen Sie diese anschließend.

Lösungsvorschlag

1. Aufgabe

a) Beschreiben Sie die Vorgehensweise des Bubblesort-Algorithmus anhand der animierten Sortierung durch „`BUBBLE-SORT`“.

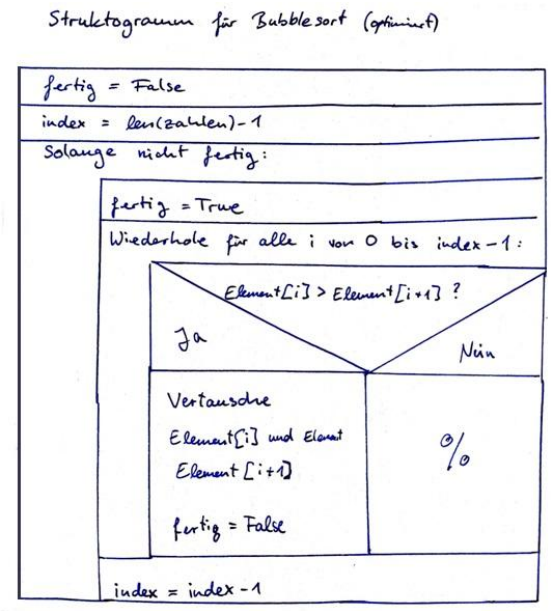
Der Algorithmus durchläuft die Elemente von vorne nach hinten und vergleicht jeweils zwei nebeneinander liegende Elemente miteinander. Dabei tauschen zwei Elemente ihren Platz, wenn das erste Element größer ist als das zweite. Bei jedem Durchlauf werden alle Elemente betrachtet.

b) Beschreiben Sie den Unterschied der Vorgehensweise des optimierten Bubblesort-Algorithmus zum vorherigen anhand der animierten Sortierung durch „`BUBBLECLEVER-SORT`“.

Unterschied: Betrachtet wird bei jedem weiteren Durchlauf nur noch der unsortierte, vordere Bereich. Der Bereich wird in jedem Durchlauf um 1 verkleinert bzw. der Algo. bricht ab, sobald keine Vertauschungen mehr notwendig sind.

2. Aufgabe

Erstellen Sie ein Struktogramm für den optimierten Bubblesort Algorithmus.



3. Aufgabe

Tauschen Sie nun mit ihrem Partner die Struktogramme. Erstellen Sie anhand des Struktogramms eine Belegungstabelle für folgende Elemente:

7	**2**	**4**	**3**	**9**
2	**7**	4	3	9
2	**4**	**7**	3	9
2	4	**3**	**7**	9
2	4	3	**7**	**9**
2	**4**	3	7	9
2	**3**	**4**	7	9
2	3	**4**	**7**	9
2	**3**	4	7	9
2	**3**	**4**	7	9

Zählen Sie dabei die Anzahl der Vertauschungen und Vergleiche.

Vertauschungen: 4

Vergleiche: 9

Beurteilen Sie das Struktogramm ihres Partners. Stellt es den optimierten Bubblesort Algorithmus korrekt dar? Schreiben Sie ggf. Verbesserungsvorschläge in das Struktogramm und besprechen Sie diese anschließend.

→ *Individuell zu lösen*

12.2.2 Zweites Arbeitsblatt: Übungen zu Listen in Python

Übungen zu Listen in Python

1. Aufgabe

a) Geben Sie in der IDLE die folgenden Ausdrücke nacheinander ein und notieren Sie die Ausgaben, sowie die Erklärung:

Eingabe	**Ausgabe**	**Erklärung**
a = [12, 24, 35, -1] a		
b = ['a', 'H', 'l', 'o'] b		
b[3]		
a = a + [0.5] a		
a[3] == 4		
a[3] = 4 a		
4 in a		
a + b		
a * b		
2 * b		
len(a)		
len(a) * len(b)		
max(a)		
min(b)		

b) Was muss man in der IDLE eingeben, um mit den Elementen der Liste b = ['a', 'H', 'l', 'o'] das Wort 'Hallo' zu schreiben?

Eingabe:_____

Ausgabe:_____

Lösungsvorschlag

1. Aufgabe

a) Geben Sie in der IDLE die folgenden Ausdrücke nacheinander ein und notieren Sie die Ausgaben, sowie die Erklärung:

Eingabe	**Ausgabe**	**Erklärung**
a = [12, 24, 35, -1] a	[12, 24, 35, -1]	Wertzuweisung
b = ['a', 'H', 'l', 'o'] b	['a', 'H', 'l', 'o']	Wertzuweisung
b[3]	'o'	Viertes Listenelement
a = a + [0.5] a	[12, 24, 35, -1, 0.5]	Der Liste wird ein weiteres Element hinzugefügt
a[3] == 4	False	Test auf Gleichheit
a[3] = 4 a	[12, 24, 35, 4, 0.5]	Listenelement mit Index 3 wird durch Wertzuweisung 4 ersetzt
4 in a	True	Prüft, ob das Element 4 in der Liste a enthalten ist
a + b	[12, 24, 35, 4, 0.5, 'a', 'H', 'l', 'o']	Verkettung von Listen (Konkatenation)
a * b	Error	
2 * b	['a', 'H', 'l', 'o', 'a', 'H', 'l', 'o']	Zwei Kopien von b aneinander hängen
len(a)	5	Ermitteln der Länge der Liste a
len(a) * len(b)	20	Multiplikation der Längen der Listen a und b
max(a)	35	Größtes Element der Liste
min(b)	'H'	Kleinstes Element der Liste

b) Was muss man in der IDLE eingeben, um mit den Elementen der Liste b = ['a', 'H', 'l', 'o'] das Wort 'Hallo' zu schreiben?

Eingabe: b[1]+b[0]+b[2]+b[2]+b[3]

Ausgabe: 'Hallo'

12.2.3 Drittes Arbeitsblatt: Sortieralgorithmus Bubblesort

Sortieralgorithmus
Bubblesort Teil 2

1. Aufgabe

a) Schreiben Sie mit Hilfe ihres Struktogramms den Bubblesort-Algorithmus in Python.

b) Sortieren Sie mit Hilfe ihres Programms folgende Listen und notieren Sie die Ausgaben:

1) [7, 2, 4, 3, 9]
   Ausgabe:_____
2) ["tisch", "Tisch", "A", "g", "a", "w"]
   Ausgabe:_____
3) [1000.001, 1000.0001, -0.3, 3/4, 1.2, -5]
   Ausgabe:_____
4) [-2, "A", 0, 2]
   Ausgabe:_____

Zusatz:

a) Modifizieren Sie die Ausgabe folgendermaßen:

```
----------------------------------------------------------------------
Sortierte Liste: [...]
----------------------------------------------------------------------
```

b) Lassen Sie zu Beginn die Ausgangsliste ausgeben:

```
[...] Ausgangsliste
```

c) Lassen Sie am Ende auch die Anzahl der Vertauschungen, Vergleiche und Durchläufe ausgeben:

```
Anzahl der Vertauschungen:  ...
Anzahl der Vergleiche:  ...
Anzahl der Durchlaeufe: ...
```

d) Lassen Sie jeden Vergleichsschritt (wie in einer Belegungstabelle) ausgeben. Machen Sie dabei kenntlich, welcher Durchlauf jeweils angezeigt wird. Beispiel:

```
[3, 1, 2] Ausgangsliste
1 . Durchlauf:
[1, 3, 2]
[1, 2, 3]
2 . Durchlauf:
[1, 2, 3]
...
```

Lösungsvorschlag

1. Aufgabe

a) Setzen Sie mit Hilfe ihres Struktogramms den Bubblesort-Algorithmus in Python um.

```
def BubbleSort(zahlen):
    fertig = False
    index = len(zahlen)-1
    while not(fertig) and len(zahlen) > 0:
        fertig = True
        for i in range(index):
            if zahlen[i] > zahlen[i+1]:
                temp = zahlen[i]
                zahlen[i] = zahlen[i+1]
                zahlen[i+1] = temp
                # Alternative Vertauschung:
               # zahlen[i], zahlen[i+1] = zahlen[i+1], zahlen[i]
                fertig = False
        index = index-1
    print(zahlen)
```

b) Sortieren Sie mit Hilfe ihres Programms folgende Listen und notieren Sie die Ausgaben:

1) [7, 2, 4, 3, 9]
   Ausgabe: `[2, 3, 4, 7, 9]`

2) ["tisch", "Tisch", "A", "g", "a", "w"]
   Ausgabe: `['A', 'Tisch', 'a', 'g', 'tisch', 'w']`

3) [1000.001, 1000.0001, -0.3, 3/4, 1.2, -5]
   Ausgabe: `[-5, -0.3, 0.75, 1.2, 1000.0001, 1000.001]`

4) [-2, "A", 0, 2]
   Ausgabe: `TypeError: unorderable types: int() > str()`

Zusatz:

```
def BubbleSort(zahlen):

    print(zahlen, "Ausgangsliste")  #Zusatz: Ausgangsliste anzeigen

    fertig = False
    index = len(zahlen)-1

    vergleiche = 0          #Zusatz
    vertauschungen = 0      #Zusatz
    durchlaeufe = 0         #Zusatz

    while not(fertig) and len(zahlen) > 0:
        fertig = True

        durchlaeufe = durchlaeufe+1   #Zusatz: Anzahl der Durchläufe zählen
        print(durchlaeufe,". Durchlauf:")

        for i in range(index):

            vergleiche = vergleiche + 1  #Zusatz: Anzahl der Vergleiche zählen

            if zahlen[i] > zahlen[i+1]:
                temp = zahlen[i]
                zahlen[i] = zahlen[i+1]
                zahlen[i+1] = temp
                # Alternative Vertauschung:
                # zahlen[i], zahlen[i+1] = zahlen[i+1], zahlen[i]

                fertig = False

               #Zusatz: Anzahl der Vertauschungen zählen:
                vertauschungen = vertauschungen + 1

            print(zahlen)   #Zusatz: Jeden Schritt ausgeben

        index = index-1
    #Zusatz: Ausgaben der Anzahlen und der sortierten Liste
    print()
    print("Anzahl der Vertauschungen: ", vertauschungen)
    print("Anzahl der Vergleiche: ", vergleiche)
    print("Anzahl der Durchlaeufe: ", durchlaeufe)
    print("-----------------------------------------------------------------")
    print("Sortierte Liste: ",zahlen)
    print("-----------------------------------------------------------------")
    print()

#Aufrufe:
zahlen = [7,2,4,3,9]
BubbleSort(zahlen)

BubbleSort(["tisch","Tisch","A","g","a","w"])     #lexikographisch geordnet
BubbleSort([1000.001,1000.0001,-0.3,3/4,1.2,-5])
BubbleSort([-2, "A", 0, 2])
```

12.2.4 Viertes Arbeitsblatt: Suchverfahren

Suchverfahren

Aufgabe 1
Beschreiben Sie die allgemeine Vorgehensweise bei der sequentiellen Suche. Müssen für die Anwendung dieses Suchverfahrens Voraussetzungen gelten? Wenn ja, welche?

_____

_____

_____

_____

_____

_____

_____

Aufgabe 2
Beschreiben Sie die allgemeine Vorgehensweise bei der binären Suche. Müssen für die Anwendung dieses Suchverfahrens Voraussetzungen gelten? Wenn ja, welche?

_____

_____

_____

_____

_____

_____

_____

Aufgabe 3
Nennen Sie für beide Suchverfahren einen Vorteil und einen Nachteil:

<u>Sequentielle Suche:</u>
Vorteil:_____
Nachteil:_____

<u>Binäre Suche:</u>
Vorteil:_____
Nachteil:_____

Lösungsvorschlag

Aufgabe 1
Beschreiben Sie die allgemeine Vorgehensweise bei der sequentiellen Suche. Müssen für die Anwendung dieses Suchverfahrens Voraussetzungen gelten? Wenn ja, welche?

Man beginnt mit dem ersten Element. Stimmt das Element mit dem gesuchten überein, kann die Suche beendet werden. Andernfalls fährt man mit dem nächsten Element fort. Die Suche läuft solange, bis das gesuchte Element gefunden oder bis das Ende der DVD-Sammlung erreicht wurde. Das Suchverfahren kann sowohl auf sortierten als auch auf unsortierte Daten angewendet werden.

Aufgabe 2
Beschreiben Sie die allgemeine Vorgehensweise bei der binären Suche. Müssen für die Anwendung dieses Suchverfahrens Voraussetzungen gelten? Wenn ja, welche?

Die Elemente, die durchsucht werden, müssen sortiert sein. Man wählt ein Element aus der Mitte aus und entscheidet, ob das gesuchte Element kleiner, größer oder gleich dem gewählten Element ist.
1. Fall: Ist das gesuchte Element kleiner, so fährt man mit dem unteren Bereich fort.
2. Fall: Ist das gesuchte Element größer, so fährt man mit dem oberen Bereich fort.
3. Fall: Ist das Element gleich dem gewähltem, so ist die Suche (erfolgreich) beendet.
Spätestens wenn der Suchbereich auf ein einzelnes Element geschrumpft ist, ist die Suche beendet. Dieses eine Element ist entweder das gesuchte Element, oder das gesuchte Element existiert nicht in der Liste.

Aufgabe 3
Nennen Sie für beide Suchverfahren einen Vorteil und einen Nachteil:

<u>Sequentielle Suche:</u>
Vorteil: *auf unsortierte und sortierte Daten anwendbar*
Nachteil: *Geschwindigkeit (im schlimmsten Fall müssen alle n Elemente durchlaufen werden)*

<u>Binäre Suche:</u>
Vorteil: *Geschwindigkeit*
Nachteil: *nur auf sortierte Daten anwendbar*

12.2.5 Geplantes Tafelbild vom 06.03.2015

	Pseudo-Code für Bubblesort Setze *fertig* auf falsch Setze *index* auf die Indexzahl des letzten Elements in der Liste Solange nicht fertig, tue: Setze *fertig* auf wahr Wiederhole für alle *i* von 0 bis *index*-1 die folg. Anweisungen: Vergleiche Element Nr i mit Element Nr. i+1 Wenn Element Nr. i größer, dann vertausche beide Setze *fertig* auf falsch Setze *index* um 1 kleiner	Listen in Python: Wertzuweisung: a = [1, 2, 3] (Index der Elemente veranschaulichen) Auf Elemente Zugreifen: a[0] Ausgabe von print(a[2]): 3 Länge einer Liste: len(a)

12.3 Tests

1. Test zur Einführung in die Programmierung

Name: Datum:

1. Welche Aussagen zum Algorithmusbegriff sind wahr? Kreuzen Sie die wahren Aussagen an.

a) Definieren Sie den Algortihmusbegriff. (1 Punkt)

- ☐ Ein Algorithmus ist die eindeutige Beschreibung eines Verfahrens zur Lösung von gleichartigen Problemen.
- ☐ Ein Algorithmus ist die mehrdeutige Beschreibung eines Verfahrens zur Lösung von unterschiedlichen Problemen.
- ☐ Ein Algorithmus ist die eindeutige Beschreibung eines Verfahrens zur Lösung von unterschiedlichen Problemen.
- ☐ Ein Algorithmus ist die mehrdeutige Beschreibung eines Verfahrens zur Lösung von gleichartigen Problemen.

b) Welche Eigenschaften hat ein Algorithmus? (1 Punkt)

- ☐ Deutlichkeit, Ausführbarkeit, Mehrdeutigkeit und Betagtheit
- ☐ Allgemeingültigkeit, Betagtheit, Deutlichkeit und Eindeutigkeit
- ☐ Allgemeingültigkeit, Ausführbarkeit, Endlichkeit und Eindeutigkeit
- ☐ Gewissheit, Ausführbarkeit, Endlichkeit und Mehrdeutigkeit

2. Bezeichner sind Namen für Variablen, Funktionen u.ä.. In Python 3 muss man einige Regeln bei der Vergabe von Bezeichnern beachten. Entscheiden Sie, welche der folgenden Bezeichner zulässig sind und kreuzen Sie diese an. (6 Punkte)

- ☐ Uhrzeit
- ☐ übertrag
- ☐ __123
- ☐ if
- ☐ 1_Name
- ☐ Name_1
- ☐ bUcHsTaBeN
- ☐ true
- ☐ Tim&Struppi
- ☐ Tim-Struppi
- ☐ E5e1
- ☐ differenz

3. Die folgenden Ausdrücke wurden in der IDLE eingegeben. Welche Ausgabe folgt auf die jeweilige Eingabe? Notieren Sie zeilenweise die Ausgaben und erklären Sie diese stichpunktartig. (8 Punkte)

Eingabe	Ausgabe	Erklärung
`3 * 5.0`		
`6**2`		
`8 > 9`		
`20 // 6`		
`20 % 6`		

`x = 10`
`x`
`x / 5`		

`y = "Zahl"`
`y`
`y + 2`
`y + 'en'`		

4. Gegeben ist ein Programm, welches zwei Zeiten abfragt, die Zeiten addiert und das Ergebnis wieder in Stunden, Minuten und Sekunden ausgibt. Finden Sie mindestens vier Fehler im Programmcode. Unterstreichen Sie die Fehler und korrigieren Sie diese am rechten Rand der jeweiligen Zeile. (4 Punkte)

```
# -*- coding: iso-8859-1 -*-
####################################
# Dateiname: zeitaddition-e
#
#            Addition zweier Zeiten
# Version:   Funktion zur Berechnung
####################################

def zeitabfrage():
    stunden  = input("Stunden: ")
    minuten  = input("Minuten: ")
    sekunden = input("Sekunden: ")
    return minuten, sekunden, stunden

def zeitaddition(h1,m1,s1,h2,m2,s2)
    summe = s1 + s2
    sek = summe%60
    ueb = summe/60

    summe = m1 + m2 + ueb
    min  = summe%60
    ueb   = summe//60

    std = h1 + h2 + ueb
    return std, minu, sek

print("erste Zeitangabe:")
std1, min1, sek1 = zeitabfrage()

print("\nzweite Zeitangabe:")
std2, min2, sek2 = zeitabfrage()

print(\nSumme:)
h, m, s = zeitaddition(std1, min1, sek1, std2, min2, sek2)
print(h, "h", m, "m", s, "s")
```

Viel Erfolg!

Note	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Punkte	20	19	18	17	16	15	14	13	12	11	10	<10	< 8	< 6	< 4	< 2

Erreichte Punktzahl: _____ von 20 Punkten. Note:

2. Test zu Sortier- und Suchalgorithmen

Name: Datum:

1. Welche Aussagen zu Suchalgorithmen sind wahr? Kreuzen Sie die wahren Aussagen an.

a) Welchen Vorteil hat die binäre Suche gegenüber der linearen Suche? (1 Punkt)

- ☐ Die binäre Suche ist sowohl auf sortierte als auch auf unsortierte Daten anwendbar.
- ☐ Die binäre Suche kann man im Gegensatz zur linearen Suche programmieren.
- ☐ Die binäre Suche ist deutlich schneller als die lineare Suche.
- ☐ Der Suchaufwand der binären Suche wächst linear mit der Anzahl der Elemente.

b) Welche Voraussetzungen müssen für die Anwendung der binären Suche gelten? (1 Punkt)

- ☐ Die Elemente, die durchsucht werden, müssen unsortiert sein.
- ☐ Die Elemente, die durchsucht werden, müssen sortiert sein.
- ☐ Das gesuchte Element muss in der Liste enthalten sein.
- ☐ Es gibt keine Voraussetzungen.

2. Beschreiben Sie die Vorgehensweise des optimierten Bubblesort-Algorithmus. (5 Punkte)

_____

_____

_____

_____

_____

_____

3. Vervollständigen Sie die Belegungstabelle, indem Sie den optimierten Bubblesort-Algorithmus auf die Elemente anwenden. Notieren Sie jeden Vergleich bzw. jede Vertauschung. (5 + 2 Punkte)

b	**A**	**a**	**C**	**B**

Geben Sie anschließend die Anzahl der Vergleiche und Vertauschungen an:

Anzahl der Vergleiche:_____

Anzahl der Vertauschungen:_____

4. Notieren Sie jeweils die Eingabe bzw. die Ausgabe. (2 + 2 + 2 Punkte)

a) Was muss man in der IDLE eingeben, um die Länge der Liste `a = [1, 2, 3]` zu ermitteln?

Eingabe: _____

b) Was muss man in der IDLE eingeben, um das Wort `'Bubble'` mit den Elementen der Liste `b = ['u', 'b', 'e', 'l','B']` zu schreiben?

Eingabe: _____

c) Was wird in der IDLE ausgegeben, wenn man nach `c = [11,22]` direkt `3*c + [33, 44]` eingibt?
Ausgabe:_____

Viel Erfolg!

Note	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Punkte	20	19	18	17	16	15	14	13	12	11	10	<10	< 8	< 6	< 4	< 2

Erreichte Punktzahl: _____ von 20 Punkten. Note:

12.4 Beobachtungsbogen

Bogen 1: Führen

Merkmalsausprägungen

Merkmal		**Beobachtungsfragen**	trifft klar zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu
(1) Klare Strukturierung des Unterrichts	1.	Die Schüler werden über den geplanten Stundenverlauf informiert.		X		
	2.	Das Klassenmanagement funktioniert (Fragen und Drannehmen, Aufgabenverteilung, Einsatz von Ritualen usw.).		X		
	3.	Im Stundenverlauf ist ein roter Faden zu erkennen.	X			
	4.	Die Lehrersprache ist verständlich und präzise.			X	
(2) hoher Anteil echter Lernzeit	5.	Das Unterrichtstempo ist dem Leistungsvermögen angepasst.	X			
	6.	Die vorhandene Zeit wird effektiv genutzt.		X		
	7.	Störungen werden zügig behoben.			X	
	8.	Die Zeitplanung hat gestimmt oder konnte korrigiert werden.	X			
(3) Inhaltliche Klarheit	9.	Die Arbeitsaufträge sind verständlich.			X	
	10.	Die Beiträge der Lehrerin sind fachlich korrekt.	X			
	11.	Die Lehrerin geht konstruktiv auf Schülerfehler ein.	X			
	12.	Die Schüler verknüpfen die Unterrichtsinhalte mit bereits Bekanntem (vernetztes Lernen).	X			
(4) transparente Leistungserwartungen	13.	Die Lehrerin orientiert sich an den Kernlehrplänen und Bildungsstandards.	X			
	14.	Die Lehrerin bespricht ihre Leistungserwartungen mit den Schülern.			X	
	15.	Die Lehrerin gibt den Schülern zügig Leistungsrückmeldungen.	X			
	16.	Verschiedene Formen der Leistungsdokumentation werden benutzt.		X		

(5) Methodentiefe	17.	Die Inhalte werden auf mindestens zwei Repräsentationsebenen dargestellt.	X			
	18.	Die Methoden passen zu den Inhalten.	X			
	19.	Die Methoden werden handwerklich korrekt eingesetzt.	X			
	20.	Es werden mindestens zwei Unterrichtskonzepte angewendet.	X			

Bogen 2: Fördern

Merkmalsausprägungen

Merkmal		**Beobachtungsfragen**	trifft klar zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu
(6) Lernfreundliches Klima	1.	Der Umgangston ist wertschätzend und respektvoll.	X			
	2.	Vereinbarte Regeln werden eingehalten.	X			
	3.	Der Lärmpegel entspricht dem Arbeitsprozess.	X			
	4.	Die Lehrerin lobt und ermutigt die Schüler aufgabenbezogen.		X		
(7) Sinnstiftendes Kommunizieren	5.	Die Lehrerin erläutert den Sinn von Aufgaben.		X		
	6.	Die Lehrerin geht auf Schülerinteressen ein.	X			
	7.	Die Schüler stellen Verständnisfragen.	X			
	8.	Die Schüler stellen kritische und weiterführende Fragen.	X			
(8) Individuelles Fördern	9.	Die Lehrerin gibt differenzierte Arbeitsaufträge.	X			
	10.	Die Lehrerin kümmert sich um einzelne Schüler.	X			
	11.	Leistungsstarke Schüler können sich aus Routineaufgaben ausklinken und an eigenen Schwerpunkten arbeiten.	X			
	12.	Schüler mit Sprachdefiziten oder anderen Handicaps erhalten Hilfen.				
(9) Intelligentes Üben		*Nur in Stunden mit Übungsphasen ausfüllen:*				
	13.	Die Übungsaufgaben passen zur Zielstellung der Stunde.				

	14. 15. 16.	Die Übungsmaterialien sind ansprechend und verständlich gestaltet. Es gibt ausreichend Zeit für Übungsphasen. Die Lehrerin hilft den Schülern, geeignete Lernstrategien zu finden.				
(10) Schüleraktivität	17. 18. 19. 20.	Die Schüler arbeiten interessiert und motiviert mit. Die Schüler beteiligen sich an Diskussionen. Die Schüler lernen selbstreguliert. Die Schüler helfen ihren Mitschülern.	X X X X			

Bogen 3: Selbstständiges Arbeiten

Merkmalsausprägungen

Merkmal		**Beobachtungsfragen**	trifft klar zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu
(1) Vorbereitete Umgebung	1.	Der Klassenraum ist für offene Lernformen hergerichtet (Flächen für gemeinsamen Unterricht, Funktionsecken, Lernstationen).			X	
	2.	Es gibt nach Interessen und Leistungsvermögen differenzierte Lernmaterialien.	X			
	3.	Spielregeln für selbstständiges Arbeiten werden eingehalten.		X		
	4.	Die Schüler kümmern sich selbst um Ordnung im Klassenraum.		X		
(2) Lernfreundliches Klima	5.	Die Schüler werden zum selbstständigen Arbeiten ermutigt.	X			
	6.	Die Schüler helfen sich gegenseitig beim Lernen.	X			
	7.	Die Schüler achten bei der Tandem- und Gruppenbildung darauf, dass niemand übrig bleibt.		X		
	8.	Die Schüler akzeptieren unterschiedliche Lerntempi und Leistungsvermögen ihrer Mitschüler.		X		
(3) Aufgaben-klarheit	9.	Die Schüler sind an der Aufgabenplanung beteiligt.	X			
	10.	Die Schüler setzen sich selbst Aufgaben.	X			
	11.	Die Schüler können den Schwierigkeitsgrad der Aufgaben einschätzen.	X			
	12.	Die Schüler sprechen über ihren Lernprozess.	X			

(4) Kompetenzorientierung	13. 14. 15. 16.	Die Lehrerin nimmt sich Zeit, die Schüler bei der Arbeit zu beobachten. Die Aufgabenstellungen können von Schülern auf unterschiedlicher Kompetenzstufe bearbeitet werden. Die Sozial- und Selbstkompetenzen der Schüler werden gefördert. Die Lehrerin bewertet die individuellen Lernergebnisse kompetenzstufenbezogen.	X X X			
(5) Selbstregulation	17. 18. 19. 20.	Die Schüler regeln selbstständig, was sie wann, wie und mit wem bearbeiten wollen. Die Schüler nutzen Lernstrategien, die der Aufgabenstellung angemessen sind. Bei Unklarheiten wenden sich die Schüler zuerst an die Mitschüler, erst danach an die Lehrerin. Die Schüler führen selbstständig Leistungskontrollen durch.	X X X X			
(6) Individuelles Fördern	21. 22. 23. 24.	Schüler helfen sich gegenseitig. Es gibt Lernmaterialien für unterschiedliche Kompetenzniveaus. Für Schüler mit sonderpädagogischem Förderbedarf ist eine Lernstandsdiagnose erstellt worden. Es gibt gezielte Förderungen auf Grundlage der Lernstandsdiagnose.	X X			