

Einreichung Innovative Lehre 2020¹

EINREICHENDE/R
Vor- und Zuname (inkl. akad. Grad/e): Dr. Franz-Karl Skala
Department: Management
Akad. Einheit/Serviceeinrichtung: Wirtschaftspädagogik
Durchwahl: 4854 E-Mail: franz-karl.skala@wu.ac.at
Ggfs. Zusammensetzung der Arbeitsgruppe²:
ALLGEMEINE ANGABEN ZUR EINREICHUNG und LV
LV wird abgehalten im Master LV-Nummer: 2235 Semester: WS 19/20 ECTS: 4 LV-Titel: Digital Business - Programmieren
Rahmenbedingungen der LV: <p>Die Lehrveranstaltung ist als Pflichtwahlfach im Studienplan des Masterstudiums Wirtschaftspädagogik verankert. Sie kann im dritten Semester belegt werden und ist eine von zwei Lehrveranstaltungen aus „Digital Business“. Während sich diese Lehrveranstaltung mit dem Erwerb von Computational Thinking im engeren beschäftigt, zielt die zweite Lehrveranstaltung auf die Entwicklung, Reflexion und Promotion von Digitalisierungsprojekten ab. Das Ziel der LV ist also, die Studierenden mit Computational Thinking vertraut zu machen, die Mensch-Computer-Interaktion auf Programmebene für sie sichtbar zu machen und die Studierenden in die Lage zu versetzen, Programme und Algorithmen in einer höheren Programmiersprache zu interpretieren, adaptieren und auch konzipieren zu können.</p> <p>Da Wirtschaftspädagogen auch die Lehrbefähigung für Unterricht aus Wirtschaftsinformatik an berufsbildenden höheren Schulen erwerben, soll diese PI bei den Studierenden einerseits ein hohes Maß an Interesse und Begeisterung wecken und ihnen andererseits implizit natürlich auch ein methodisches Rüstzeug für die mögliche didaktische Umsetzung der Inhalte im Regelunterricht an BMHS mitgegeben werden.</p> <p>Zu beachten ist dabei, dass die Studierenden aus vorgelagerten Lehrveranstaltungen zwar über profunde Kenntnisse im Umgang mit Tabellenkalkulationsprogrammen und Datenbanksystemen (Modellierung, SQL) verfügen, darüber hinaus jedoch keinerlei Vorkenntnisse mitbringen. Es wird daher in diesem Kontext von absoluten Novizen ausgegangen.</p>
Links in die Online-Umgebung der LV: <ul style="list-style-type: none"> • Die gesamten Kursmaterialien stehen hier zur Verfügung: https://learn.wu.ac.at/dotlrn/classes/pool/2235.19w • Ausgewählte Beispiele zu allen drei Ansätzen sowie Demonstrationen von Studierendenprojekten stehen unter folgendem Link zur Verfügung.

¹ Im Jahr 2019 (SS 2019, WS 2019/20) abgehaltene Lehrveranstaltungen können für den Lehrpreis eingereicht werden. Lehrveranstaltungen, welche über zwei Semester andauern, (WS 2018/19-SS 2019) können ebenfalls eingereicht werden.

² Bitte nennen Sie hier alle Personen, die an der Entwicklung des LV-Konzepts beteiligt waren. Die hier genannten Personen werden im Falle einer Prämierung mit ausgezeichnet.

1. KURZBESCHREIBUNG DES LEHRVERANSTALTUNGSDESIGNS (max. 180 Wörter)

Die hier verfasste Kurzbeschreibung wird im Falle einer Prämierung gemeinsam mit dem eingereichten Formular auf der Website der WU veröffentlicht.

Die Antwort auf die Frage, ob Bildungsinstitutionen in ihrer Lehre die Digitalisierung der Lebenswelt thematisieren sollen ist weltweit wohl unbestritten. Das zeigen unter anderem aktuelle Aktionen wie der Digitalpakt Schule in Deutschlandⁱ oder der Masterplan für Digitalisierung im Bildungswesen des BMBWFⁱⁱ. Die Antwort auf die Frage, wie „Digitale Kompetenz“ erworben werden kann, ist jedoch höchst umstritten. Eine Seite des Meinungsspektrums moniert, dass das Bedienen grafischer Benutzeroberflächen „Digitale Kompetenz“ nur unzureichend abbildet und daher „Coding“ als Basisfähigkeit Digitalkompetenter unerlässlich sei. Auf der anderen Seite wird argumentiert, dass der Mensch dem Computer in Zeiten künstlicher Intelligenz durch Geist und Fantasie überlegen sei. Programmieren und „Coding“ sei daher keinesfalls als Grundkompetenz in Lehrpläne aufzunehmen, sondern es gelte ein möglichst humanistisch geprägtes Bildungsideal anzustreben. Man könne ja auch Autofahren, ohne selbst einen Motor gebaut zu haben.ⁱⁱⁱ

In den Diskussionen um Digitalisierung, künstliche Intelligenz, Datenanalyse und Datenschutz sowie Robotik und schlussendlich digitaler Ethik zeigt sich oftmals, dass nicht nur unter Lernenden immer wieder bemerkenswerte Fehlvorstellungen von der „Blackbox Computersystem“ (die App, die Cloud, der Roboter) vorherrschen. Diese Fehlvorstellungen, ließen sich möglicherweise durch einen Blick unter die „Motorhaube“ relativ leicht korrigieren. Das Ziel der PI ist daher, dass Studierende der Wirtschaftspädagogik im Sinne eines weiter gefassten Konzepts des „*Computational Thinkings*“ bessere Vorstellung von der Funktionsweise von Computersystemen erlangen, dadurch Neugierde an der Thematik geweckt wird und sie vielleicht sogar Spaß daran haben, sich später eigenständig tiefer in die Thematik einzuarbeiten. Einerseits profitieren sie davon unmittelbar in ihrer beruflichen und privaten Lebenswelt. Andererseits können sie selbst als künftige Lehrerinnen und Lehrer Fehlvorstellungen von Schülerinnen und Schülern vermutlich besser auflösen.

Um das Ziel der Lehrveranstaltung zu erreichen, wurde ein 3-Phasen-Modell als Trilogie entwickelt. Diese Phasen bilden unterschiedliche Zugänge zum Erlangen von Computational Thinkings ab, konkurrieren jedoch nicht, sondern greifen logisch ineinander. Diese sind:

1. Phase: der **spielerische Zugang** mit dem Ziel, Interesse und Neugierde wecken, spielerisch Algorithmen zu entwickeln und durch das Erzielen möglichst greifbarer Ergebnisse eine positive Grundmotivation herzustellen.
2. Phase: der **fachpraktische Zugang** mit dem Ziel, die spielerisch erlernten Grundelemente in einer höheren Programmiersprache den semantischen und syntaktischen Regeln entsprechend anzuwenden.
3. Phase: der **anwendungsorientierte Zugang** mit dem Ziel, anhand einer höheren Programmiersprache kleine, aber reale, konkrete und sinnvolle IoT-Projekte zu entwickeln und umzusetzen.

Die Rückmeldungen nach Erprobung des Konzepts der Lehrveranstaltung durch die Studierenden und die erbrachten Leistungen sind vielversprechend. Trotz hohem zugeschriebenem Workload und hoher kognitiver Anforderungen herrscht eine hohe Grundzufriedenheit. Die PI hilft den Studenten lt. Evaluierungsergebnissen dabei, sich selbstständig in die Inhalte zu vertiefen und sich kritisch mit den Inhalten auseinanderzusetzen.

2. AUSFÜHRLICHE DARSTELLUNG DES LEHRVERANSTALTUNGSDESIGNS

2a.) Überblick

- Welche Learning Outcomes sollen die Studierenden erreichen?
- Wie ist die LV inhaltlich und strukturell aufgebaut?
- Aus welchen Elementen setzt sich die Endnote zusammen?

Welche Learning Outcomes sollen die Studierenden erreichen?

Das Ziel der Lehrveranstaltung besteht darin, dass die Studierenden nach Abschluss der Lehrveranstaltung in der Lage sind:

- LO A: Algorithmen zur Problemlösung zu **strukturieren**.
- LO B: Algorithmen in einer bildungsorientierten **visuellen Programmiersprache umzusetzen**.
- LO C: Algorithmen in einer **höheren Programmiersprache umzusetzen**.
- LO D: Einfache IoT-Projekte zu **konzipieren** und zu **realisieren**.
- LO E: Grenzen und Reichweiten von IoT Projekten zu **reflektieren**.

Wie ist die LV inhaltlich und strukturell aufgebaut?

Die Lehrveranstaltung ist inhaltlich, aber auch methodisch als **Trilogie** gestaltet. Der Gedanke der Trilogie bildet sich **inhaltlich** logisch in der Struktur der Präsenzphasen ab. Die **Methode** folgt dem gewählten Zugang. Die Präsenzphasen werden durch „Out-of-Class“-Übungen flankiert, die je nach inhaltlicher und methodischer Ausrichtung wiederum unterschiedliche Kompetenzdimensionen fördern sollen.

Nach Auseinandersetzung mit dem state of the art zu „Computational Thinking“ (siehe dazu 2b), der Reflexion des Ansatzes in die intendierten Learning Outcomes und der Analyse der Eingangsvoraussetzungen, wurden die folgenden drei Phasen konzipiert:

- **Phase 1:** Der **spielerische Zugang**: Nach einem organisatorischen und kurzen generell inhaltlichen Teil, startet die Informationsvermittlung in den ersten beiden Lehrveranstaltungseinheiten nach dem Ansatz „Lernen durch Probieren und Reflektieren“. Anhand der visuellen Programmiersprache Scratch sollen Programmabläufe spielerisch kennengelernt werden. Im Blickfeld stehen die Ebenen Denkprozess, Abstraktion, Dekomposition, algorithmisches Design und Evaluation. Jedoch nicht linearer, sondern als iterativer Prozess zwischen Dekomposition und Evaluation. Scratch wurde aufgrund der lerntheoretischen Begründung gewählt, der guten Usability und der hohen Fehlertoleranz. Es müssen keine Code-Zeilen geschrieben werden, sondern Programme können aus vorgefertigten Blöcken, welche zum Beispiel Anweisungen darstellen, per Drag&Drop im Baukastensystem zusammengestellt werden. Dies erlaubt ein hohes Maß an Flexibilität, vor allem deshalb, da die Befehle nicht erst kompiliert werden müssen und die Änderungen am Algorithmus unmittelbar sichtbar werden. Weiters ist keine Installation notwendig und Projekte können online zwischen den Studierenden und dem LV-Leiter ausgetauscht und *remixed* werden.
- **Phase 2:** Der **fachpraktische Zugang**: Nachdem die grundlegende Logik der Erstellung eines einfachen Computerprogramms auf spielerische Weise erprobt wurde, erfolgt die Einführung in eine Hochsprache. Hier wurde Python gewählt, da sie durch ihre klare Struktur und reduzierten Syntax einen gut lesbaren und knappen Programmierstil fördert. Da Python-Programme zur Laufzeit interpretiert werden, müssen sie nicht kompiliert werden. Das erleichtert das Adaptieren oder Erweitern des Programms bzw. auch das Beheben von Fehlern und nochmalige Ausprobieren enorm. Python ist nicht an spezifische Betriebssysteme gebunden und kann daher unter Windows, MacOS, Linux, Android usw. ausgeführt werden. Obwohl Python als sehr einfach zu erlernen gilt, sind die Befehle nun jetzt in der exakten Syntax anzuschreiben. Ein fehlender Doppelpunkt oder ein falscher Datentyp bei einer Variablen führt unweigerlich zu einer Fehlermeldung. Die Überleitung von Phase 1 zu Phase 2 kann methodisch über das Tool EduBlocks^{iv} begünstigt werden. Dieses Tool wandelt Blöcke, wie man es aus Scratch gewohnt ist, in Python-Code um.
- **Phase 3:** Der **anwendungsorientierte Zugang**: Ein wesentliches Problem beim Einstieg in eine höhere (objektorientierte) Programmiersprache ist, dass es einen fundierten Sockel an Fähigkeiten bedarf, um einfache, über das Banale hinausgehende, Algorithmen zu realisieren. Zwar werden im Rahmen der Lehrveranstaltung brauchbare Anwendungsbeispiele erstellt (Prüfen von IBANS, Visualisieren von Schulstandorten, Berechnung von Flächen bzw. Umfängen etc.). Aus Sicht der Lernenden ist es zwar motivierend, wenn ein Programm funktioniert und ein nachvollziehbarer Output

erreicht wird. Das Ergebnis wirkt aber oft abstrakt. Der Anspruch der Phase 3 ist aus diesem Grund, das Ergebnis eines Programms genauso leicht fassen zu können wie in Phase 1, jedoch unter Einsatz eines regelgeleiteten Vorgehens von Phase 2. Um dieses Ziel zu erreichen wird ein Einplatinencomputer (Raspberry Pi) an die Studierenden ausgegeben. Über Python-Programme kann auf dessen GPIO-Pins zugegriffen werden. Daran können Komponenten wie Sensoren (Luftdruck, Luftfeuchte, Temperatur, Gyroskop und Beschleunigungssensor), Motoren, Bewegungssensor, Distanzmesser, Monitor uvm. angeschlossen werden und somit einfache IoT-Projekte realisiert werden.

Daraus ergibt sich folgende Grundstruktur:

EH	Didaktische Überlegung	Methodische Vorgangsweise	Überwiegend adressierte Ebenen des CT	Zentrale Frage (Adressierte Learning Outcomes)
Phase 1: Der spielerische Zugang				
EH1	Erfahrungslernen; Programmabläufe spielerisch kennenlernen; rasches erzielen sichtbarer Ergebnisse	Probieren, Reflektieren, Probieren, ...	Denkprozess Abstraktion Dekomposition Algorithmisches Design Evaluation Generalisierung Automation	Wie können die Grundkonzepte der Programmierung (Variablen, Verzweigungen, Schleifen, Objekte und Klassen, Methoden, Funktionen) spielerisch erlebt und begreifbar gemacht werden? (LO A, LO B)
EH2				
Phase 2: Der fachpraktische Zugang				
EH3	Erlernen und Erproben eines strukturierten, regelgeleiteten Vorgehens. Programmierstil erlernen. Dokumentieren des Vorgehens.	Strukturieren, Realisieren, Probieren, Reflektieren, Optimieren	Denkprozess Abstraktion Dekomposition Algorithmisches Design Evaluation Generalisierung Automation	Wie können Algorithmen und Programme in einer höheren Programmiersprache umgesetzt werden? (LO A, LO C)
EH4				
EH5				
Phase 3: Der anwendungsorientierte Zugang				
EH6	Verschränkung des regelgeleiteten fachtheoretischen eher abstrakten Zugangs mit Elementen der realen Welt. Erzielen sichtbarer Ergebnisse.	Konzipieren, Strukturieren, Realisieren, Probieren, Reflektieren, Optimieren, Dokumentieren	Denkprozess Abstraktion Dekomposition Algorithmisches Design Evaluation Generalisierung Automation	Wie können mit Hilfe der Kenntnisse einer höheren Programmiersprache einfache Internet-of-Things (IoT)-Projekte geplant, realisiert, dokumentiert und deren Ergebnisse evaluiert werden? Wie können Messdaten der realen Welt (Umgebungswerte, Eingaben) erfasst, verarbeitet, analysiert und gespeichert werden? (LO A, LO D)
EH7				
Präsentation der Projektarbeiten & Erprobung				
EH8	Erprobung der Projekte und Reflexion des Entstehungsprozesses. Diskussion der User Experience und möglicher Erweiterungsmöglichkeiten.	Probieren, Reflektieren, Optimieren, ggf. Konzipieren	Denkprozess Abstraktion Dekomposition Algorithmisches Design Evaluation Generalisierung Automation	Reflexion des Erstellungsprozesses. Festhalten der Lessons Learned. Reflexion des Produkts. Optimierungsmöglichkeiten, Grenzen & Reichweiten. Positives und negatives Potenzial. (LO D, LO E)

2b.) „Lehrmethoden“

- Welche Lehrmethoden und mediendidaktischen Elemente setzen Sie ein, um die Studierenden beim Erreichen der Learning Outcomes zu unterstützen? Welche Rolle spielt die Online-Unterstützung dabei?
- Aus welchem Grund haben Sie sich genau für diese Methode(n) entschieden? Was möchten Sie damit erzielen? Welche besonderen Vorteile sehen Sie im Einsatz der Methode(n)?
- Wie greifen Sie die mediendidaktischen Methoden im weiteren Lehrveranstaltungsverlauf auf?
- Inwieweit profitieren die Studierenden von den mediendidaktischen Methoden in der Lehrveranstaltung?

Grundlegend muss festgehalten werden, dass der Antragsteller der Auffassung ist, dass ein Grundverständnis der Funktionsweise von Computerprogrammen Fehlvorstellungen auflösen kann. Der Katalysator hierfür ist das Konzipieren von Programmen und Algorithmen, sowie das Erlernen der Grundlagen einer höheren Programmiersprache. Das Erlernen einer Programmiersprache kann spannend gestaltet werden, auch ohne den Anspruch zu haben, Programmierer auszubilden.

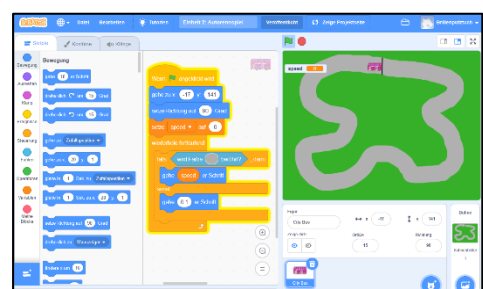
In der Literatur gibt es jedoch keinen common sense wie das Gelingen könnte. Fest dürfte nur stehen: „programming is hard to learn, and hard to teach. These problems are frequently acknowledged within the computing education community and are confirmed in several studies (Bennedson 2008, Berglund/Lister 2007, Pears et al 2007, Robins/Rountree/Rountree 2003 zit. nach Berglund/Lister 2010). Dementsprechend vage sind auch die Vorstellungen, wie Computational Thinking (CT) verstanden bzw. dann auch vermittelt werden kann: „there is little agreement about what computational thinking encompasses, and even less agreement about strategies for assessing the development of computational thinking in young people“ (Brennan/Resnick 2012: 1).

In der Literatur können dafür unterschiedliche Denkschulen identifiziert werden. Dem semiotischen maschinennahen Ansatz (Koffman 1986, Liebenau/Backhouse 1990) stehen idealisiert problemlösungsorientierte Ansätze (Sorva 2013, Monk 1981) gegenüber. Idealisiert problemlösungsorientierten Ansätzen wird entgegengebracht, „this does not always lead to a fruitful understanding of the machine, as it is based on an abstraction of its capabilities presented by the chosen programming language – and programming languages differ in their design“. (Spangsberg/Brynskov 2018: 742f). Gegen den semiotischen maschinennahen Ansatz führen Berglund und Lister (2010) ins Treffen: „we argue that many computing educators and computing education researchers operate from within narrower views of the didactic triangle. For example, computing educators often teach programming based on how they relate to the computer, and not how the students relate to the computer“ (2010: 35). Das führt auf motivationalen Ebene mitunter dazu „that frustration and engagement do vary according to the type of task given to the students“ (ebd: 40), sofern diese als nicht realitätsnahe bzw. sinnbehaftet von den Studierenden wahrgenommen werden.

Sie schlagen daher vor, die Didaktik der Programmierung, und somit wohl auch CT durch das didaktische Dreieck nach Kansanen (1999) zu sehen, also neben der Inhalts-Lehrer-Beziehung auch die Inhalts-Studenten-Beziehung sowie die Lehrer-Studenten-Beziehung vor dem jeweiligen Kontext zu berücksichtigen. Ähnlich argumentieren auch Spangsberg und Brynskov (2018), die eine semiotische Sichtweise auf CT vorschlagen, welche sowohl die maschinenorientierte (subface) als auch die anwenderorientierte (surface) Sichtweise vereint. Als Kontext für die Vermittlung von Computational Thinking kann das gemeinsame Grundverständnis von CT herangezogen werden, das Selby und Woollard (2013) wie folgt aus der Literatur abgeleitet haben: 1) A thought process, 2) Abstraction, 3) Decomposition, 4) Algorithmic design, 5) Evaluation, 6) Generalisation und 7) Automation (Selby/Woollard 2014). Je nachdem welche Phase des CT adressiert wird, ergeben sich im Zusammenspiel mit dem didaktischen Dreieck somit logischerweise unterschiedliche methodische Überlegungen, die im 3-Phasen-Modell reflektiert werden.

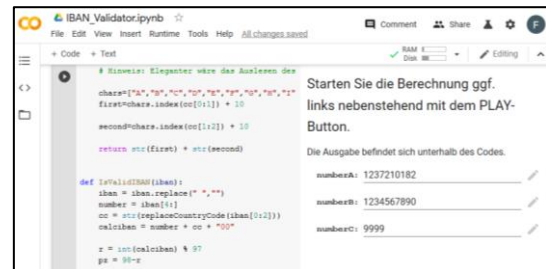
Welche Lehrmethoden und mediendidaktischen Elemente setzen Sie ein, um die Studierenden beim Erreichen der Learning Outcomes zu unterstützen? Welche Rolle spielt die Online-Unterstützung dabei?

In der **1. Phase, dem spielerischen Zugang**, steht eindeutig das **Probieren** und **Reflektieren** durch die Studierenden im Vordergrund. Es soll der eigenständige Denkprozess, Abstraktion sowie Dekomposition gefördert werden. Den Studierenden werden zwar die grundlegenden Konzepte einer Programmiersprache nähergebracht. Die Erprobung ebendieser erfolgt jedoch vorwiegend durch die Studierenden zuhause im Selbststudium anhand der Programmiersprache SCRATCH. Die Hausübungen bestehen im Wesentlichen aus einem **Video**,



welches die **fertige Lösung** zeigt. Unter Zuhilfenahme der Folien aus den Präsenzeinheiten sollen die ersten vier Phasen des CT, insbesondere die **Dekomposition**, gefördert werden. Über das **Forum** auf LEARN können sie sich austauschen. Der Prozess der Erstellung muss **reflektiert** und die zentralen Probleme und Herausforderungen werden festgehalten (vgl. exemplarisch im Anhang). Diese werden dann, genauso wie ausgewählte studentische Lösungen, in der darauffolgenden Lehrveranstaltungseinheit thematisiert. Im Fokus der Beurteilung steht nicht unmittelbar die Qualität des fertigen Programms, sondern die **Darstellung der Vorgangsweise**. Schließlich sind in der Programmierung mehrere Lösungen zulässig. Manche davon eleganter, manche weniger. Der Lehrende agiert hier eher als **Coach**, der bei Fragen zur Verfügung steht und ansonsten die unterschiedlichen Vorgangsweisen und Ergebnisse thematisiert und diskutiert.

In der **2. Phase, dem fachpraktischen Zugang**, steht das Strukturieren, Realisieren und Optimieren von Programmen in einer höheren Programmiersprache im Fokus. Das Augenmerk liegt hier stark auf dem **Content** und auf dem Umsetzen von Algorithmen in der Hochsprache Python. Für Novizen kann das fehlerhafte Setzen einer Klammer oder eines Anführungszeichens zu einer erheblichen Arbeitszeitverlängerung führen. Zwar ist das Identifizieren und Beheben von Syntax- und Logikfehlern in der Programmierung eine wesentliche Kompetenz. Im Rahmen des Lernprozesses kann dies aber schnell in **Frustration** umschlagen, da es ja bei der Umsetzung eines Algorithmus in einer Hochsprache zu überhaupt keinen Programmablauf mehr kommt (im Gegensatz zu Scratch). In der Out-of-Class-Phase werden daher vorwiegend **repetitive Beispiele** bearbeitet, welche das Erlernte festigen sollen. Am Ende der Hausübungen werden jedoch noch gezielt schwierigere Aufgabenstellungen – tw. auch fakultativ – angeboten, wo es um die Vernetzung unterschiedlicher Konzepte geht. Diese werden dann zu Beginn der folgenden Lehrveranstaltung im Plenum besprochen bzw. erprobt. In dieser Phase werden ausgewählte Inhalte auch für das Selbststudium mittels **Lecturecasts** ausgelagert. Der Lehrende agiert hier vor allem als **Inhaltsvermittler**.



```
# Hinweis: Eleganter wäre das Auslesen des
iban = iban.replace(" ", "")
first4 = iban.index(cc[0:1]) + 10
second4 = iban.index(cc[1:2]) + 10
# iban = iban[first4:second4]

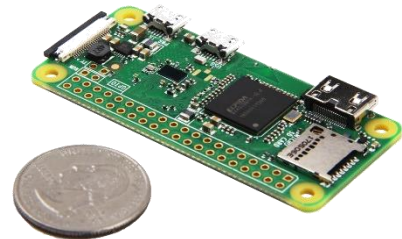
def isValidIBAN(iban):
    iban = iban.replace(" ", "")
    number = iban[4:]
    cc = iban.replaceCountryCode(iban[0:2])
    calciban = number + cc + "00"
    z = int(calciban) % 97
    ga = 98-z
```

Starten Sie die Berechnung ggf. links nebenstehend mit dem PLAY-Button.

Die Ausgabe befindet sich unterhalb des Codes.

numberA:	1237210182
numberB:	1234567890
numberC:	9999

In der **3. Phase, dem anwendungsorientierten Zugang**, steht das Sichtbarmachen des Ergebnisses eines Programms im Vordergrund. Da der Output von Python-Programmen auf dem Niveau Novizen für Interessierte in der Regel eher abstrakt ist und vielmehr der Erstellungsprozess im Vordergrund steht, ist das Ziel der dritten Phase, Programme zu erstellen, die mit der **direkten Umwelt interagieren**. Durch das Bewegen von Motoren, das Messen von Distanzen, erfassen von Bewegungen, Temperatur, Feuchte, Helligkeit uvm. können einfache IoT-Projekt nachgebaut bzw. auch konzipiert werden. Da der verwendete Einplatinencomputer (Raspberry PI zero) de facto ein kleiner vollwertiger Computer ist, der über eine WLAN bzw. Bluetooth-Anbindung verfügt, ist es problemlos möglich, auch einen Webserver, ein (einfaches) Datenbankmanagementsystem über ein USB-Akkupack zu betreiben. Realisiert wurden unter anderem Prototypen für einen **Abfahrtsmonitor** für die Wiener Linien für den Schreibtisch, eine **Roboterarmsteuerung** per Gyroskop, eine einfache **Wetterstation** mit Bewegungssensor, eine **Zufriedenheitsabfrage** für den Hörsaal oder auch eine **Gangschaltung** für Elektromotoren. Je nach zeitlichen Dispositionen für die Projektarbeiten, kann der Schwierigkeitsgrad variiert werden. Die Studierenden sollen und müssen aber jedenfalls: Konzipieren, Strukturieren, Realisieren, Probieren, Reflektieren, Optimieren, Dokumentieren. Der Lehrende agiert hier zunächst in den Inputphasen als **Inhaltsvermittler** und in den Out-of-Class-Phasen als **Coach**, der in eigenen Sprechstundenslots bzw. im Forum oder auch telefonisch bzw. per E-Mail zur Verfügung steht.



Während des **Präsenzunterrichts** wird in allen drei Phasen als mediendidaktische Methode das Smartboard verwendet. In der dritten Phase wird außerdem der Tisch des Vortragenden mit Kamera gefilmt und kann über Blizz (www.blizz.com) von den Studierenden auf deren Rechner gestreamt werden.

In den **Out-of-Class-Phasen** werden die bewährten Tools von LEARN (Lernmaterialien, Aufgabentool, Forum und Gruppenmail) eingesetzt. Außerdem werden Videos gezielt in den Angaben der Hausübungen eingebunden (meist direkt in PPTX-Dateien) und andererseits auch als Vorbereitung für folgende Lehrveranstaltungseinheiten bzw. als Hilfestellung für Projektarbeiten über **LEARN-Lecturecasts** zur Verfügung gestellt. So entstanden in Summe ca. knapp **30 Videos** im Umfang von jeweils ca. 5 bis max. 20 Minuten, die Studierende zur Vorbereitung oder Nachbereitung bzw. als Hilfestellung bei Projektarbeiten nutzen können.

Aus welchem Grund haben Sie sich genau für diese Methode(n) entschieden? Was möchten Sie damit erzielen? Welche besonderen Vorteile sehen Sie im Einsatz der Methode(n)?

Je nach Phase variiert der Einsatz mediendidaktischer Tools. Während Videos in der zweiten Phase eher als Informationsvermittlung zur Vorbereitung auf die Folgeeinheit zur Verfügung gestellt wurden (ähnlich des Konzepts eines „flipped classrooms“), werden sie in der dritten Phase eher als Unterstützung bei Projektarbeiten bzw. als Nachbereitung der Lehrveranstaltungseinheiten eingesetzt. Somit könnte das vorliegende Lehrveranstaltungskonzept salopp auch als „mixed classroom“ bezeichnet werden.

Da diese Lehrveranstaltung erstmalig durchgeführt wurde, und der Antragsteller über den Sommer an der WU nicht beschäftigt war, war zunächst nicht angedacht, das Lehrveranstaltungskonzept systematisch durch Videos zu unterstützen. Während der Lehrveranstaltung wurde jedoch offensichtlich, dass vor allem aufgrund technischer Probleme, Tippfehler oder andere Unaufmerksamkeiten zu viel wertvolle Präsenzeinheit verloren gehen kann. Da das Lehrveranstaltungskonzept jedoch immer davon ausging, die Input-Sessions wenn möglich mit 15 Minuten zu beschränken und danach sofort zu üben, bot sich die Erprobung von Inhaltsvideos an, die dann auch gut angenommen wurden. Die Entscheidung für Videos für die Projektarbeiten erfolgte ebenfalls erst zur Laufzeit der Lehrveranstaltung, da das Konzept der Verkabelung von Komponenten auf einem Breadboard für Novizen offensichtlich herausfordernder sein kann als ursprünglich angenommen. Diese Videos wurden ebenfalls sehr gut angenommen.

Wie greifen Sie die mediendidaktischen Methoden im weiteren Lehrveranstaltungsverlauf auf?

Während der Lehrveranstaltungseinheiten werden die Lecturecasts sowie die Videos aus den Hausübungen als Start für die Präsenzeinheiten aufgegriffen. Es erfolgt in der Regel eine kurze Wiederholung durch das Stellen von Wissensfragen bzw. durch studentische Erklärungen. Danach folgt die Erprobung erster Beispiele, wobei dann ziemlich schnell klar wird, ob und in welcher Tiefe sich die Studierenden mit den Videos auseinandergesetzt hatten.

In Phase 3 wird neben dem aktiven Einsatz des Smartboards auch der Vortragenden-Tisch bei Bedarf (Aufbau der Schaltungen am Breadboard) mittels Videokonferenzsoftware (Blizz) gestreamt. Studierenden können daher den Aufbau direkt auf ihrem Monitor verfolgen und nachbauen, ohne ausschließlich auf die Folien angewiesen zu sein.

Inwieweit profitieren die Studierenden von den mediendidaktischen Methoden in der Lehrveranstaltung?

Vom Einsatz der Lecturecasts und Videos profitieren die Studierenden deshalb, da eine Individualisierung des Lerntempos möglich wird. So fällt der implizite soziale Druck weg, wenn etwas nicht sofort verstanden wurde. Die Videos können nicht nur zur Vorbereitung auf den Unterricht, sondern auch zur Nachbereitung angesehen werden. Auch bei der Erstellung der Projektarbeiten kann darauf wieder zurückgegriffen werden.

Das Streamen des Tisches des Vortragenden in Phase 3 hat sich bewährt, da das Schritt-für-Schritt-Vorgehen des Aufbaus des Breadboards aber auch des Codes nachvollziehbarer ist, als das mit einem Schema am Foliensatz möglich wäre. In einem Hörsaal mit zwei Monitoren (wie im D2.0.031) könnte auf einem Monitor in Zukunft immer der Monitor des Vortragenden-PCs gestreamt werden. Gerade in den letzten Reihen fällt es nämlich oftmals schwer, den Blick immer zwischen Beamer und dem eigenen Rechner schweifen zu lassen.

2c.) *Innovativer Charakter der LV*

- Innerhalb welcher Dimension (welche in der Ausschreibung unter Kapitel 2 angeführt sind) siedeln Sie Ihre Einreichung an?
- Welche didaktischen Elemente Ihres Konzepts erachten Sie als besonders innovativ im Hinblick auf das Schwerpunktthema „Lernprozesse online unterstützen“?
- Transferfunktion: Inwiefern ist Ihr LV-Design auf andere Lehrveranstaltungen übertragbar? Welche Lehrmethoden und mediendidaktischen Elemente könnten auch in anderen Veranstaltungen an der WU zum Einsatz kommen?
- Welche Lehrmethoden und mediendidaktischen Elemente können für eine neuerliche Abhaltung der LV noch verbessert/überdacht werden?

Innerhalb welcher Dimension (welche in der Ausschreibung unter Kapitel 2 angeführt sind) siedeln Sie Ihre Einreichung an?

„Programming is a rewarding and yet demanding field in the ICT labor market, but it is considered a challenging and difficult area of learning for significant numbers of novice programmers“ (Malik 2018: 637). Herausfordernd ist, dass die Studierenden nicht nur eine spezifische Syntax und Semantik einer Programmiersprache erlernen müssen, sondern auch entsprechende Problemlösungskompetenzen entwickeln müssen. Diese Problemlösungskompetenz kann am besten über einen ganzheitlichen Ansatz des „Computational Thinkings“ (Spangenberg/Brynskov: 2018) entwickelt werden. „Computational thinking is the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent“ (Wing 2017). Möglichst realitätsnahe, nachvollziehbare Beispiele unterstützen diesen Prozess: „Examples are important teaching tools. Research in cognitive science confirms that examples appear to play a central role in the early phases of cognitive skill acquisition“. (Börstler/Caspersen/Nordström 2015: 232) Das trifft insbesondere beim Erlernen einer Programmiersprache zu, da „Students generalize from examples and use them as templates for their own work“. (ebd) Sie müssen jedoch einfach zu generalisieren sein und im Einklang mit den zugrundeliegenden Lernzielen stehen. Insofern ist es notwendig, Beispiele zu entwickeln: (ebd: 233):

- die vom Computer tatsächlich ausgeführt werden können, da sie in Maschinensprache übersetzt werden,
- gleichzeitig jedoch auch von den Studierenden verstanden werden können, indem sie auf deren reale Lebenswelt und deren Eingangsvoraussetzungen adressieren
- und die konsistent kommunizieren, welche zugrundeliegenden Konzepte damit verbunden sind.

Anspruch des didaktischen Konzepts ist also, durchgängig möglichst realitätsnahe und nachvollziehbare Beispiele zu entwickeln, die in Maschinensprache übersetzt werden und zu tatsächlichem, sichtbarem Output führen. Sie müssen über die rein idealisierte „notional machine“ (Sorva 2013) und das Entwickeln von „Pseudo-Code“ hinausgehen, ohne die Novizen eingangs jedoch mit den syntaktischen Details einer höheren Programmiersprache zu überfordern. Gleichzeitig soll aber auch der Überlegungsprozess gefördert werden, da das Erlernen von Computational Thinking (CT) nur durch intensive Nachdenk- und Reflexionsphasen möglich ist. Mit dem vorliegenden Konzept wird versucht, die einzelnen Schritte des CT in unterschiedlichen Phasen sowohl konzeptiv als auch mediendidaktisch spezifisch zu fördern. Daraus kann gefolgert werden, dass die Einreichung am besten der Dimension „*LV-Designs, in welchen interaktive und multimediale Formen der Wissensvermittlung eingesetzt werden, um von Studierenden als schwierig wahrgenommene Inhalte zu vermitteln*“ zugeordnet werden.

Welche didaktischen Elemente Ihres Konzepts erachten Sie als besonders innovativ im Hinblick auf das Schwerpunktthema „Lernprozesse online unterstützen“?

Bevor nochmals auf die mediendidaktische Umsetzung eingegangen wird, muss zunächst noch einmal auf die Konzeptionsphase zurückgegriffen werden, da das Schwerpunktthema „Lernprozesse online unterstützen“ bereits bei der Auswahl der Inhalte (mit)berücksichtigt wurde. Bei **Phase 1** (der spielerische Zugang) fiel die Wahl unter anderem Deshalb auf Scratch, weil die visuelle Programmiersprache nicht nur lerntheoretisch begründet wurde, sondern auch, weil sie in der neuesten Version 3.0 auch kostenlos online verfügbar und ausführbar ist. Zahlreiche Materialien finden sich somit nicht nur online, sondern können auch bei Interesse direkt online ausprobiert werden. Somit können die Studierenden andere Projekte auf der Plattform scratch.mit.edu aufrufen und online ansehen, ausprobieren und remixen. Diese Überlegung wurde auch bei **Phase 2** und **3** mitgedacht.

Für das Erlernen der semantischen und syntaktischen Grundlagen der Programmiersprache Python gibt es online zahlreiche gute Dokumentation und viele Hilfeforen als Nachschlagewerke. Die Überleitung von Phase 1 auf Phase 2 kann gut über das Tool EduBlocks erfolgen, da hier die gewohnten Blöcke aus Scratch als Python-Code dargestellt werden können. Die Überleitung bzw. Umsetzung von Python in IoT-Projekten erfolgt dann durch das Digitalisieren von einfachen Sachverhalten der realen Welt. Die Wahl fiel deshalb auf den Raspberry Pi, da hier ebenfalls zahlreiche Online-Ressourcen vorliegen und dieser ursprünglich für den Einsatz im Unterricht konzipiert wurde.

Hinsichtlich des konkreten Lernprozesses folgt die Lehrveranstaltung keinem bestimmten Ansatz bzw. Paradigmas wie zB: einem „Inverted Classroom“. Das wäre bei den Schritten des **Computational Thinkings**: *Denkprozess, Abstraktion, Dekomposition, Algorithmisches Design, Evaluation, Generalisierung, Automation* auch nicht sinnvoll. Online-Phasen werden vielmehr punktuell, einerseits als Vorbereitung, andererseits als Nachbereitung und dann wiederum als Nachschlagemöglichkeit genutzt. Somit kann das Lehrveranstaltungskonzept eher als „Mixed classroom“ umschrieben werden. In den Online-Phasen wird ein Mix mediendidaktischer Elemente, vom Forum bis hin zum Lecturecast eingesetzt.

Transferfunktion: Inwiefern ist Ihr LV-Design auf andere Lehrveranstaltungen übertragbar? Welche Lehrmethoden und mediendidaktischen Elemente könnten auch in anderen Veranstaltungen an der WU zum Einsatz kommen?

Die Grundintention des Lehrveranstaltungskonzepts ist, absoluten Novizen Neugierde und Freude am Erlernen eines generell als kompliziert oder schwierig zu erlernendem Sachverhalt mitzugeben. Das LV-Design verfolgt dabei einen stark anwendungsorientierten Ansatz. In allen Phasen der Konzeption – von der ersten Idee, über das Grobkonzept bis hin zur Detailplanung lag ein Vorgehen nach Berücksichtigung des didaktischen Dreiecks nach Kansanen (Vortragender – Thema – Studierende) im Vordergrund. Dabei wurden die Sachverhalte und konkreten Beispiele jeweils auch nach dem Ansatz des „algorithmic sign“ gedacht. Der Ansatz ist der Versuch, den traditionellen maschinenorientierten Ansatz des Computational Thinkings mit dem idealisierten problemlösungsorientierten Ansatz gegenüber zu stellen, um beide Denkrichtungen abzudecken. Einerseits, die kausale Interpretation des Sachverhalts durch die Maschine und andererseits die intendierte Interpretation des Anwenders (Brynskov 2018). Dadurch kann die Relevanz des Gelernten zum einen aus informationstheoretischer Sicht beleuchtet werden und zum anderen die Auswirkung auf für die Zielgruppe greifbare Sachverhalte thematisiert werden. Die Stärke des Lehrveranstaltungskonzepts ist meines Erachtens somit der Einsatz sehr realitätsnahe Beispiele und Anwendungen. Und das obwohl die Eingangsvoraussetzungen der Zielgruppe sehr niedrig sind. Dennoch ist es gut gelungen – wie auch die Rückmeldungen zeigen – so etwas wie Freude und tw. Begeisterung an der Thematik zu wecken.

Die mediendidaktischen Methoden und Tools auf LEARN sind selbstverständlich auch auf zahlreiche andere Lehrveranstaltungen gut zu übertragen und werden dort natürlich auch schon vielseitig eingesetzt. Einzig das Streamen des Vortragenden-PCs auf die PCs der Teilnehmerinnen und Teilnehmer ist – in Hörsälen ohne LC-Installation – über das Tool Blizz (oder natürlich auch alternative Anwendungen) gerade für Lehrveranstaltungen in Schulungsräumen oder bei BYOD-Settings durchaus eine Überlegung.

Welche Lehrmethoden und mediendidaktischen Elemente können für eine neuerliche Abhaltung der LV noch verbessert/überdacht werden?

Wie bereits erwähnt war die Erstellung einer Vielzahl von Videos und Lecturecasts ursprünglich so nicht geplant. Da die Lehrveranstaltung erstmalig angeboten wurde, lagen keine Referenzwerte hinsichtlich Eingangsvoraussetzungen, Motivation und Interesse der Studierenden vor. Das Angebot an Lecturecasts ist daher sukzessive anlassbezogen gewachsen und könnte stärker systematisch strukturiert werden.

Lecturecasts, die als Vorbereitung für die folgenden Präsenzveranstaltungen dienen, müssen zwingend um gezielte kleine Aufgabenstellungen ergänzt werden. Nur dadurch kann sichergestellt werden, dass die Videos aktiv verfolgt und nicht nur „nebenbei abgespielt“ werden. Die Umsetzung der Wissensfragen bzw. Aufgabenstellungen könnte dabei entweder anhand von interaktiven Videos (zB: via H5P) oder durch adaptive Lernstränge erfolgen.

Die Umsetzung der Phase 3 war ein eher mutiges Projekt, da es hier sehr leicht zu technischen Problemen kommen kann, die sich nur schwer eingrenzen lassen. Beispielsweise ist hier das Verbinden der Rechner in das EDUROAM-Netzwerk zu nennen. Potenzielle Fehlerquellen sind hier nicht nur falsche Benutzernamen und Passwörter (WLAN-Passwort \neq Powernet-Kennung), sondern auch syntaktisch inkorrekt gesetzte Konfigurationsdateien bis hin zu Konfigurationsdateien, die mit dem falschen Zeichensatz (zB: UTF8 statt ANSI) abgespeichert wurden. Das Equipment wurde daher bereits ergänzt (zB: Ankauf von USB-TTL-Kabeln für den seriellen Anschluss oder HDMI-Mini-HDMI-Adapter udgl.) und die Anleitungen spezifiziert, um bei neuerlichen Durchläufen auf allfällige Probleme besser vorbereitet zu sein.

Für das Wintersemester 2021 ist schließlich angedacht, spezifisch noch Breakout-Termine zu planen, in denen die Studierenden ihre Hausübungen (der Phase 2 und 3) bzw. dann die Projektarbeiten bei Bedarf gemeinsam, bzw. auch unter Anwesenheit des LV-Leiters in der Rolle als Coach, lösen können. Zwar fanden sich bei diesem Durchgang einige Studierende in selbst gebildete Lern- und Projektgruppen zusammen. Für berufstätige Studierende oder Pendler würde eine frühzeitige Kommunikation solcher Breakout-Termine möglicherweise unterstützend wirken.

Anhänge

- Evaluierungsergebnisse
- Schaubild: Blitzlichter aus der Lehrveranstaltung
- Beispiel: Reflexion einer Hausübung aus Phase 1

Quellen

- ⁱ <https://www.zeit.de/gesellschaft/schule/2019-02/digitalpakt-schulen-digitalisierung-bildung-bund-laender>
- ⁱⁱ <https://www.bmbwf.gv.at/Themen/schule/zrp/dibi.html>
- ⁱⁱⁱ Vgl. hierzu: <https://www.spiegel.de/lebenundlernen/schule/lehrergestaendnis-warum-wir-informatik-als-pflichtfach-brauchen-a-1294827.html> bzw. <https://www.zeit.de/gesellschaft/schule/2019-05/digitalisierung-schulen-informatik-unterricht-programmieren-digitalpakt> oder <https://www.faz.net/aktuell/wirtschaft/wirtschaftspolitik/gegen-programmier-unterricht-in-der-grundschule-15055414.html>
- ^{iv} <https://edublocks.org>

Literatur

- Berglund, A., & Lister, R. (2010). Introductory Programming and the Didactic Triangle. *12th Australasian Computing Education Conference (ACE 2010)*, (S. 35-44). Brisbane, Australien.
- Börstler, J., Caspersen, M. E., & Nordström, M. (Vol. 24 2016). Beauty and the Beast: on the readability of object-oriented example programs. *Software Qual J*, S. 231-246.
- Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. *Annual American Educational Research Association*, (S. 1-25). Vancouver, Canada.
- Malik, S. I. (2017). Improvements in Introductory Programming Course: Action Research Insights and Outcomes. *Journal of Educational Computing Research*.
- Selby, C., & Woollard, J. (2013). *Computational thinking: the developing definition*. University of Southampton.
- Sorva, J. (2013). *Notional Machines and Introductory Programming Education*.
- Spangsberg, H., & Brynskov, M. (No. 10. Vol. 8 2018). The Nature of Computational Thinking in Computing Education. *International Journal of Information and Education Technology*, S. 742-746.
- Wing, J. M. (2010). *Computational Thinking: What and Why?*

WU-Lehrveranstaltungsevaluierung: Ergebnisse

Lehrveranstaltung	2235 - Wahlfach - DigiBiz - Programmieren
LV-Leiter/innen	Dr. Franz-Karl Skala
Semester	Winter 2019/2020
Anmeldungen	0
Abgegebene Fragebögen	16

Kontextdaten

Studienrichtung		Geschlecht	
Meine bisherige Anwesenheit in den LV-Einheiten	Die inhaltlichen Anforderungen der LV erscheinen mir	Das Arbeitspensum (Workload) für diese Lehrveranstaltung erscheint mir	

Evaluierungsdaten

Didaktische Aufbereitung

MW
Med.

Die LV hilft mir, die Inhalte selbständig zu vertiefen			1.38 1.23
Die bearbeiteten Beispiele helfen mir beim Verständnis der Inhalte			1.44 1.30
Ich halte die Form dieser LV für geeignet, die Inhalte zu vermitteln			1.38 1.17
Summenindex: 1.23			

Lehrveranstaltungs-konzept

MW
Med.

Die Ziele und Anforderungen der LV waren mir von Anfang an bekannt			1.81 1.23
Der Inhalt der LV entspricht den anfänglich definierten Lehrzielen			1.62 1.39
Der Zusammenhang zu anderen LVs der Studienrichtung ist mir klar			2.00 1.50
Summenindex: 1.37			

Lehrveranstaltungsleiter/in (übungsorientierte LV)

MW
Med.

Der/die LV-Leiter/in geht auf Verständnisfragen und Anmerkungen ein			1.12 1.07
Die Studierenden erhalten ausreichend Gelegenheit selbst zu Wort zu kommen			1.06 1.03
Der/die LV-Leiter/in strukturiert die einzelnen LV-Einheiten in der Regel sehr gut			1.44 1.30
Summenindex: 1.13			

Selbstevaluierung Studierende (übungsorientierte LV)

MW
Med.

Ich bin in den LV-Einheiten immer gut vorbereitet			1.81 1.75
Ich beteilige mich aktiv an den Diskussionen in der LV			1.50 1.23
Bei Unklarheiten frage ich nach oder versuche sie zu klären			1.19 1.12
Summenindex: 1.37			

Rahmenbedingungen

MW
Med.

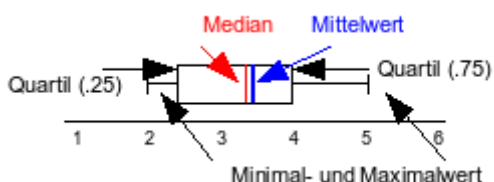
Die Veranstaltungstermine liegen für mich sehr günstig			1.62 1.23
Der Veranstaltungsraum ist für diese LV sehr gut geeignet			1.00 1.00
Die Raumausstattung ist für diese LV sehr gut geeignet			1.06 1.03
Summenindex: 1.09			

Gesamteindruck

MW
Med.

Diese LV regt mich zur kritischen Auseinandersetzung mit den Inhalten an			1.44 1.23
Der/die LV-Leiter/in bemüht sich sehr um den Lernerfolg der Studierenden			1.12 1.07
Insgesamt bin ich mit dieser LV sehr zufrieden			1.38 1.30
Insgesamt lerne ich in dieser LV sehr viel			1.25 1.17
Summenindex: 1.19			

Erläuterung der Boxplots:



LEARN / LV-Evaluierung

Was mir an der LV besonders gut gefällt

Offene Frage - "Was mir an der LV besonders gut gefällt"

Spannende neue Inhalte

Hr. Skala hat für jedes Problem eine Lösung

Viel Aufwand aber OK.

verschiedene Phasen der LV um StudentInnen an Thematik heranzuführen

guter Professor

Bestes Wahlfach im Studium, sollte unbedingt beibehalten werden

- > Es macht Spaß, kleine Programme zu programmieren. Bis jetzt konnte ich das überhaupt nicht.
- > Ich verstehe jetzt die Funktionsweise von technischen Geräten viel, viel besser
- > kleinere Schritte bzw. mehr auf 2 verschiedenen Ebenen erklären

▷ Verständlich + strukturiert "Programmieren" erlebbar
→ spielerischen Aufbau

Wo ich Verbesserungspotentiale sehe

Offene Frage - "Wo ich Verbesserungspotentiale sehe"

mehr Materialen

Hausübung zu schwer

Kurseinheiten wöchentlich abhalten, damit Wissen nicht zu schnell verloren geht

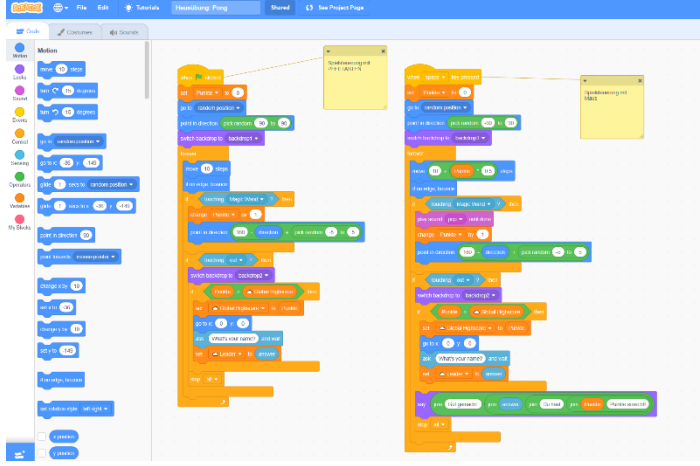
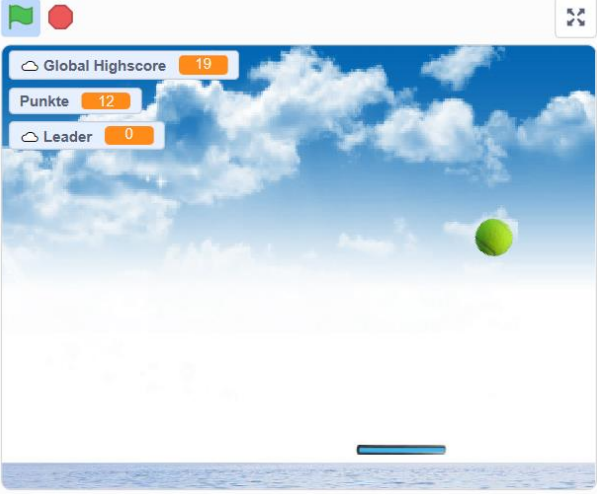
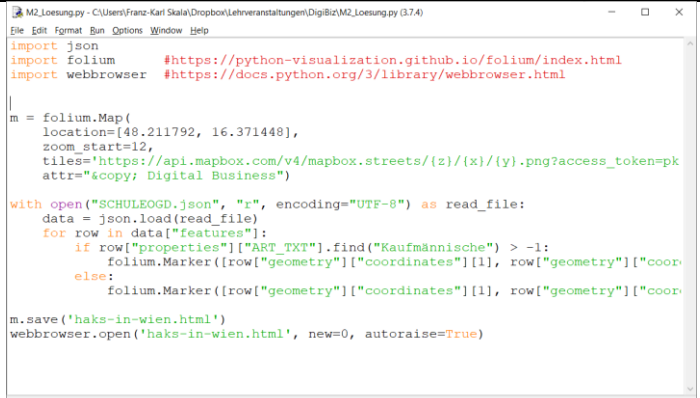

- > Sucht ist ein tolles Programm, eine zusätzliche Einheit zu Python 3 würde sich mir noch wünschen
- > mehr, kurze Klausurstellungenstermine bzw. engere (Wiederholung ist die Mutter des Lernens)
- > zeitintensivere Hausübungen

→ Rückmeldung für 1 Hausübung wäre hilfreich

→ "Blanko" Code installieren ist (nur bedingt) hilfreich → eigen
Zeile n bei
PI erstellen

Digital Business - Programmieren

Blitzlichter aus den unterschiedlichen Phasen des Trilogie-Ansatzes

Phase 1: Der spielerische Zugang		
Ansatz	Adressierte Learning Outcomes	Ziel
Die Studierenden mit den grundlegenden Elementen zur Algorithmbildung vertraut zu machen. Spielerisch Konzepte wie Verzweigungen, Schleifen, Variablen, Objekte und Ereignisse kennenzulernen. Konzipieren und Umsetzen eigener Algorithmen.	<ul style="list-style-type: none"> Algorithmen zur Problemlösung zu strukturieren. Algorithmen in einer bildungsorientierten visuellen Programmiersprache umzusetzen. 	Das Erlernen der grundlegenden Systematik einer Programmiersprache. Kennenlernen des prozeduralen Ablaufs von Programmen. Kennenlernen der Logik objektorientierter Ansätze.
Sichtweise: Subface		Sichtweise: Surface
		
Phase 2: Der fachpraktische Zugang		
Ansatz	Adressierte Learning Outcomes	Ziel
Die Studierenden mit den semantischen und syntaktischen Grundlagen einer höheren Programmiersprache vertraut machen. Regelgeleitetes Umsetzen und Adaption von bestehenden Algorithmen. Programme in einer Hochsprache lesen und ggf. manipulieren können.	<ul style="list-style-type: none"> Algorithmen zur Problemlösung zu strukturieren. Algorithmen in einer höheren Programmiersprache umzusetzen. 	Umsetzen einfacher Algorithmen in einer höheren Programmiersprache. Lesen und adaptieren von Algorithmen in einer Hochsprache. Fähigkeit, bestehende Programme um Funktionen zu erweitern.
Sichtweise: Subface		Sichtweise: Surface
Beispiel: Anzeige aller Schulstandorte in Wien unter Hervorhebung der Handelsakademien		
		

Beispiel: Programm zur Überprüfung der Validität von IBANs

```

def replaceCountryCode(iban):
    # Wie Funktion bekommt einen Parameter mit zwei Buchstaben übergeben (zB: AT, DE usw)
    # Wie soll für die Buchstaben A den Wert 10, für den Buchstaben B den Wert 11 usw. zurückgegeben (zB=5)
    # Wird DE übergeben, ist das Ergebnis 124, AT wäre 129 usw.
    # Lege deshalb eine Liste mit 26 Buchstaben an (Index 0 bis 25)
    characters = "A B C D E F G H I J K L M N O P Q R S T U V W X Y Z"
    firstchars.index(iban[0:2]) + 10 # Suche jetzt den ersten Buchstaben in diesem Index und gib diese Index-Nr. zurück
    secondchars.index(iban[2:4]) + 10 # Suche jetzt den zweiten Buchstaben in diesem Index und gib diese Index-Nr. zurück
    # zB: 18 für F, addiere jetzt die Zahl 10 zu dem Index, somit wird für F 28 zurückgegeben.
    # gib jetzt die Folge der ermittelten Zahlen als String zurück. Hier darf nicht addiert werden!
    return str(first) + str(second)

def isValidIBAN(iban):
    # Entferne von der IBAN alle Leerzeichen
    iban = iban.replace(" ", "")
    number = iban[4:] # Lasse nur die Ziffern ab der fünften Stelle aus (BLZ+Kontonummer)
    cc = str(replaceCountryCode(iban[0:2])) # Übergib die ersten zwei Stellen (ländercode) an die Funktion replaceCountryCode()
    deliban = number + cc + "00" # Setze die Restzahlitalienitalien zusammen aus der Nummer (BLZ+Kontonummer), dem Buchstabenwert der Funktion replaceCountryCode und zwei folgenden 00
    r = int(calculiban) % 97 # Berechne den Rest der Division mit 97 modulo 97
    pr = "98r" # Nachkommastellen von 98 das Ergebnis der Modulof- Berechnung = Prüfziffer
    # Wenn die errechnete Prüfziffer pr gleich den Stellen 3+4 der übergebenen iban
    # dann handelt es sich um eine gültige IBAN
    # sonst
    # handelt es sich um eine ungültige IBAN
    return True
    return False

print(isValidIBAN("AT2134567890123456789012345678901234567890"))
    
```

```

Python 3.7.4 Shell
File Edit Shell Debug Options Window Help

Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22)
[MSC v.1916 32 bit (Intel)] on win32
Type "help()", "copyright()", "credits()" or "license()" for more information.
>>>
RESTART: C:\Users\Franz-Karl Skala\Dropbox\Lehrveranstaltung
en\DigiBiz\G5.py
False
>>>
    
```

Phase 3: Der anwendungsorientierte Zugang

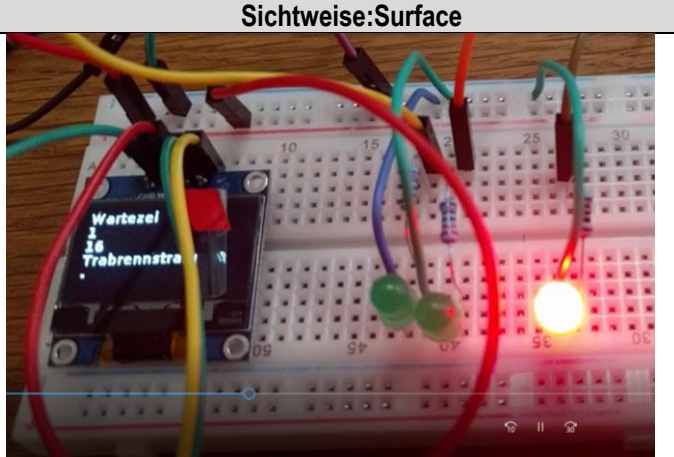
Ansatz	Adressierte Learning Outcomes	Ziel
<p>Wie können Messdaten der realen Welt (Umgebungswerte, Eingaben) erfasst, verarbeitet, analysiert und gespeichert werden? Welche Auswirkung haben Programme und Algorithmen auf mein tägliches Leben?</p>	<ul style="list-style-type: none"> Algorithmen zur Problemlösung zu strukturieren. Einfache IoT-Projekte zu konzipieren und zu realisieren. 	<p>Einfache IoT-Projekte zu konzipieren, umzusetzen und zu evaluieren. Zu erkennen, welche Vor- und Nachteile IoT-Devices für den Alltag mit sich bringen. Grenzen und Reichweiten unterschiedlicher Lösungen zu diskutieren.</p>

Projektbeispiel: Abfahrtsmonitor für Busstation Trabrennstraße 87A für den Schreibtisch

Sichtweise: Subface

```

Datei Bearbeiten Format Ansicht Hilfe
while True:
    if checkTimer == 0:
        url = urllib.request.urlopen("https://www.wienerlinien.at/ogd_realtime/monitor?rbl=3353")
        data = json.loads(url.read().decode())
        countdown=data["data"]["monitors"][0][0]["lines"][0]["departures"][0][0]["departureTime"]+"countdown"
        countdown+=data["data"]["monitors"][0][0][0]["departures"][1][0][0]["departureTime"]+"countdown"
        Haltestelle=data["data"]["monitors"][0][0][0]["locationStop"]["properties"]["title"]
        checkTimer = CHECK_TIME
        checkTimer = checkTimer - 1
        monitor.write(("%Wartezeit", str(countdown), str(countdown1), Haltestelle, "-"))
    if countdown > 5:
        GPIO.output(27, GPIO.LOW)
        GPIO.output(22, GPIO.LOW)
        GPIO.output(17, GPIO.LOW)
    elif countdown >= 2:
        GPIO.output(17, GPIO.LOW)
        if green_blink == 0:
            green_blink = 1
            # print("ein")
            GPIO.output(27, GPIO.HIGH)
    
```



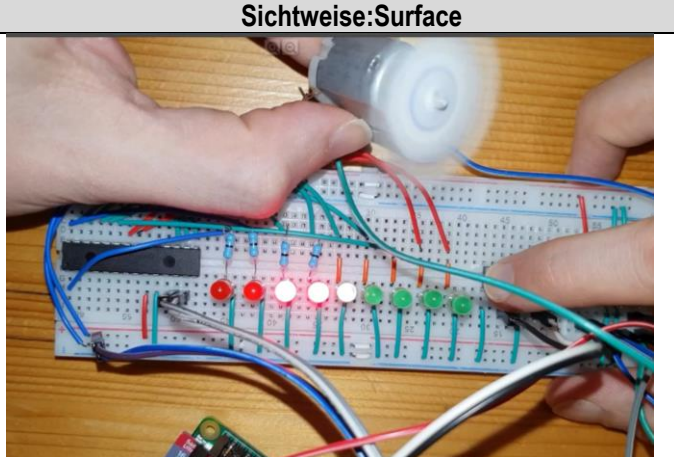
Projektbeispiel: Steuerung eines Elektromotors mit 9 Gängen

Sichtweise: Subface

```

Datei Bearbeiten Format Ansicht Hilfe
while True:
    if GPIO.input(gearup) == GPIO.LOW:
        if gear < 4:
            gear = gear + 1
    if GPIO.input(gearardown) == GPIO.LOW:
        if gear > max*-1:
            gear = gear - 1

    print(gear)
    lights(gear)
    if gear >= 0:
        forward(gear * 2.5)
    else:
        backward(gear*-1 * 2.5)
    sleep(0.15)
    
```



WAHLFACH PROGRAMMIEREN

HAUSÜBUNG 1

AUFGABE A

Link zum Beispiel: <https://scratch.mit.edu/projects/338364900>

(Tipp: Das Programm muss veröffentlicht worden sein!)

Die folgenden Probleme/Herausforderungen sind bei der Erstellung aufgetreten:

Da es leider keine Figur „Würfel“ gibt, war die erste Hürde eine Figur eigens zu erstellen und verschiedene Kostüme (Zahlen 1-6) zu entwerfen. Das Erstellen der zufälligen Zahl, welche erscheint, stellte kein Problem dar, allerdings die Implementierung der Achsendrehung. Der erste Versuch – Drehung um 360 Grad –ging schief. Die Drehung erfolgte, nur leider war die Bewegung selbst nicht zu sehen, da die Drehung ja sofort erfolgte und nur die Endstellung, welche die gleiche wie die zu Beginn war, zu sehen war. Dies muss demnach in vielen einzelnen Bewegungen um ein paar Grad erfolgen, bis die 360 Grad erreicht werden. Dabei reflektierte ich darüber nochmals unter Rücksichtnahme auf das Geisterbeispiel, da dieses das anspruchsvollste in der 1. Einheit war. So habe ich ein Bewegungsintervall als Variable erstellt und als Operator bei der Bewegung 360 / Bewegungsintervall festgelegt.

Bezüglich der mehreren Würfel war ich mir nicht sicher wie das am Besten erfolgen würde 2 bzw. 3 Würfel gleichzeitig zu werfen, deshalb entschied ich mich für Trial-and-Error und versuchte es einfach mit dem Nachricht senden, sobald eine bestimmte Taste (1,2,3) gedrückt wird. Dies hat auch bestens funktioniert.

Zusammenfassung Probleme/Herausforderung:

Würfel selbst erstellen

Intervall implementieren um 360 Grad-Drehung sichtbar zu machen

AUFGABE B

Link zum Beispiel: <https://scratch.mit.edu/projects/338400004/>

(Tipp: Das Programm muss veröffentlicht worden sein!)

Die folgenden Probleme/Herausforderungen sind bei der Erstellung aufgetreten:

Das erste Problem ergibt sich bei der Interaktion zwischen den Tieren. Bei einer Berührung von Dinosaurier und Katze erfolgt keine Pause bzw. auch nicht die vorgesehene Änderung des Kostüms des Dinos. Anstatt wie bisher gleiten zu verwenden, wechselte ich zu Bewegung in 10er-Schritten in Richtung der Maus bzw. der Katze. Ein weiteres Problem ergab sich durch den Start. Falls sich die Tiere in der Runde davor berührten, starteten sie in dieser Position wieder, was zu einem kurzen Spiel führt, da diese sich ja bereits berührt haben und damit ein Tier gefangen wird.

In weiterer Folge gab es ein Problem mit der Funktion „Skript stoppen“. So wurde beispielsweise dieses der Katze gestoppt, wenn sie die Maus fängt, aber der Dinosaurier hat sich noch eine Zeit lang der Katze entgegenbewegt, bis der Klang und die Nachricht abgespielt wurden, da erst danach das Stopp einsetzt. Hier stellt sich die Frage wie ein Stopp eingeführt werden kann, welches sofort alle Charaktere bewegungsunfähig macht, ohne die Kommentare und Klänge sofort zu unterbinden.

Als letztes großes Problem ergibt sich die Alternative dem Mauszeiger zu folgen. Beides zugleich funktioniert nicht, da die Maus immer dem Mauszeiger folgt und somit ein drücken der Tasten keinen Sinn macht, da die Maus dem Mauszeiger nachläuft.

Zusammenfassung Probleme/Herausforderungen:

Berührungen zwischen Charakteren und damit einhergehende Aktionen

Ungünstige Startpositionen der Charaktere bei erneutem Spielen

Verzögerung von „Skript stoppen“ durch vorhergehende Aktionen (Klänge, Aussagen der Charaktere)

Mauszeiger und Tasten für Bewegen der Maus funktionieren simultan nicht