Klausur Einführung in die Programmierung I

Bjoern Stuetz (bstuetz@lehre.dhbw-stuttgart.de)

2020-12-22

Klausur - MUSTER

Übersicht

- Klausurdauer: **120 Minuten**
- Mögliche Gesamtpunktzahl: **120 Punkte**
- Es gilt das *"Merkblatt"Internet-gestützte Open-Book-Klausuren"* in der aktuellen Ausführung
- Alle Hilfsmittel (siehe Hinweise und Merkblatt)

Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
A	1	05	[ ]
A	2	05	[ ]
A	3	05	[ ]
A	4	05	[ ]
B	1	10	[ ]
B	2	10	[ ]
B	3	10	[ ]
B	4	10	[ ]
C	1	30	[ ]
C	2	30	[ ]
–	–	**Summe**	**Erreicht**
–	–	**120**	**[ ]**

Tabelle 1: Aufgabenteile und Punkte

Falls Ihnen nicht immer sofort die richtige Lösung einfällt, beschreiben Sie bitte Ihren Lösungsansatz kurz und knapp mit einer Schema-Zeichnung, in Pseudocode oder direkt in Worten.

Viel Erfolg!

Informationen zur 'Programmierung auf Papier'

Bitte nehmen Sie sich kurz Zeit für diesen Abschnitt. Dieser wurde auch bereits in der Vorlesung vorgestellt. Es sind alle Hilfsmittel erlaubt. Programmieren Sie in dieser Klausur konzeptionell (d.h. soweit wie möglich vollständig aber sinnvoll und zielorientiert vereinfacht). Vermeiden Sie unnötige Teile, die nicht ausdrücklich gefragt wurden. Die Syntax sollte größtenteils korrekt sein; es ist allerdings wichtiger Ihre Idee auszudrücken, als vollständig übersetzbaren (also compilierbaren) Code zu schreiben.

Vereinfachte Darstellung

Folgende Dinge können vereinfacht dargestellt werden (außer es wird ausdrücklich in der Fragestellung auf die reguläre Darstellung verwiesen):

- Ausgabe von Text muss nicht formatiert werden.
- Input von Text wird nie über die Konsole verwendet, Input-Variablen werden vorab zugewiesen.
- `__str__()` sind immer vorhanden, auch wenn diese nicht von Ihnen ausprogrammiert wurden.
- Module sind vorhanden und automatisch importiert.
- Standard-Konstruktoren, also die `__init__()` Methode, sind immer vorhanden, auch wenn diese nicht von Ihnen ausprogrammiert wurden.

Diese Punkte müssen entsprechend **nicht** ausprogrammiert bzw. voll ausgeschrieben werden.

Programmierstil

Bitte beachten Sie folgende Hinweise zum Programmierstil: Beachten Sie die Regeln und Konventionen der Programmiersprache. Programmieren Sie auf **deutsch** oder **englisch** (*engl.*); falls Ihnen nicht das richtige Wort in englisch einfällt, nutzen Sie einfach ein **deutsch**es Wort. Bitte denken Sie an die notwendigen Konventionen für Python (z.B. Einrückung, Variablen, Namen). Bitte rücken Sie den Programmcode so weit wie möglich ein.

Bitte schreiben Sie leserlich und verständlich.

Informationen zur Verwendung von Python

- siehe gesonderte Hinweise zur Klausur in Moodle

Aufgabenteil A - Grundlagen

Aufgabe A1 - Python und Python Interpreter

Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
A	1	05	[ ]

Problem:

- Für was stehtdas Programm `python`, dass Sie unter anderem in der Konsole aufrufen, also z.B. `$ python` ...?
- Was ist der Python Interpreter? Was ist der Unterschied zwischen interpretierten und compilierten Sprachen?
- Was ist eine typische Dateiendung für Python-Quellcodedateien?

Lösung:

Beschreiben und erklären Sie jeweils bitte **kurz**:

- (A1.1) `python`
- (A1.2) Python Interpreter, Compiler
- (A1.3) Dateiendung

Aufgabe A2 - Print

Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
A	2	05	[ ]

Problem:

Erstellen Sie ein Python-Programm in der Datei `main.py`,

- das den `string` `'Hallo Du!'` auf der Kommandozeile mit `print()` ausgibt.
- Wie führen Sie das Programm aus?
- Was ist die Ausgabe?

Lösung:

Programmieren Sie bitte:

- (A2.1) `main.py`

Beschreiben Sie anhand des Programms bitte **kurz**:

- (A2.2) `main.py` in der Kommandozeile ausführen
- (A2.3) Ausgabe des Programms auf der Kommandozeile

Aufgabe A3 - Dateitypen

Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
A	3	05	[ ]

Problem:

Erstellen Sie ein kommentiertes Python-Programm in der Datei `convert.py`, das

- die Gleitzahl `1.7` (`float`)
- zu einer Ganzzahl (`int`) umwandelt
- und auf der Kommandozeile ausgibt.

Lösung:

Programmieren Sie bitte **mit** wenigen erläuternden Kommentaren:

- (A3.1) `convert.py`

Beschreiben Sie anhand des Programms bitte **kurz**:

- (A3.2) `convert.py` in der Kommandozeile ausführen
- (A3.3) Ausgabe des Programms auf der Kommandozeile

Aufgabe A4 - Typisierung

Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
A	4	05	[ ]

Problem:

Was bedeuten es, dass die Programmiersprache Python

- stark typisiert
- und dynamisch, impliziert typisiert

ist?

Lösung:

Erklären Sie jeweils bitte **kurz**:

- (A4.1) stark typisiert
- (A4.2) dynamisch, impliziert typisiert

Aufgabenteil B - Weiterführende Themen

Aufgabe B1 - Rechnen und Funktionen

Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
B	1	10	[ ]

Problem:

Erstellen Sie ein Python-Programm in der Datei `divide.py`, das

- in der Funktion `divide()`
  - als Parameter zwei `int`-Werte erhält,
  - diese teilt (Division Parameter *1* durch Parameter *2*),
  - das Ergebnis als `float`-Zahl zurückgibt,
  - und das Ergebnis auf der Kommandozeile mit `print()` ausgibt.
- Wie rufen Sie in einem Python-Programm die Funktion auf? Nehmen Sie als Beispiel die Zahlen `4` und `2`. Wie lautet das Ergebnis?

Lösung:

Programmieren Sie bitte:

- (B1.1) `divide.py`
- (B1.2) Aufruf von `divide()` mit `4` und `2`

Beschreiben Sie anhand des Programms bitte **kurz**:

- (B1.3) `divide.py` in der Kommandozeile ausführen
- (B1.4) Ausgabe des Programms auf der Kommandozeile

Aufgabe B2 - Funktionen und Stringausgabe

Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
B	2	10	[ ]

Problem:

Erstellen Sie ein Python-Programm in der Datei `hallo.py`, das

- das den `string` 'Hallo' über eine Funktion mit einem Parameter
  - mit `print()` auf der Kommandozeile ausgibt
  - der Namen der Funktion lautet `hallo()`
  - die Funktion erhält als Parameter einen `string`, der nach 'Hallo' ausgegeben wird
- Wie rufen Sie in einem Python-Programm die Funktion auf? Nehmen Sie als Beispiel-Parameter einen `string` mit Ihrer **Matrikelnummer**
- Was ist eigentlich eine Funktion in Python?

Lösung:

Programmieren Sie bitte Kommentaren:

- (B2.1) `hallo.py`
- (B2.2) Aufruf von `hallo()`, mit Ihrer **Matrikelnummer** als Parameter

Beschreiben Sie anhand des Programms bitte **kurz**:

- (B2.3) `hallo.py` in der Kommandozeile ausführen
- (B2.4) Ausgabe des Programms auf der Kommandozeile

Erklären Sie bitte **kurz**:

- (B2.5) Funktionen in Python

Aufgabe B3 - Schleifen

Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
B	3	10	[ ]

Problem:

Erstellen Sie ein kommentiertes Python-Programm in der Datei `loop.py`, das

- in einer Schleife von `1` bis `100` zählt, und dabei mit `print()` jede **ungerade** Zahl auf der Kommandozeile ausgibt;
  - denken Sie bitte dabei an die Standardfunktion `range()` und
  - denken Sie bitte dabei an den Modulo-Operator `%`, z.B. `if i % 2`.
- Welche andere Art von Schleifen ist noch möglich? Nennen Sie eine **andere** Möglichkeit

Lösung:

Programmieren Sie bitte **mit** wenigen erläuternden Kommentaren:

- (B3.1) `loop.py`

Beschreiben Sie anhand des Programms bitte **kurz**:

- (B3.2) `loop.py` in der Kommandozeile ausführen
- (B3.3) **Beispiel**-Ausgabe des Programms auf der Kommandozeile für die **ersten vier Durchläufe**

Erklären Sie bitte **kurz**:

- (B3.4) die Unterschiede zwischen beiden Möglichkeiten

Aufgabe B4 - Verzweigungen

Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
B	4	10	[ ]

Problem:

Erstellen Sie ein Python-Programm in der Datei `condition.py`, das

- eine erste Variable `sweet = False` definiert (süß),
- eine zweite Variable `salty = True` definiert (salzig),
- ein Programm, das mit `if`, `elif` und/oder `else` folgendes erreicht:
  - Ausgabe des Programms auf der Kommandozeile mit `print()`
    * wenn `sweet` `True` und `salty` `False`:
      · `string` `'sweet'` (süß)
    * wenn `sweet` `False` und `salty` `False`:
      · `string` `'bland'` (fade, ohne Geschmack)
    * wenn `sweet` `False` und `salty` `True`:
      · `string` `'salty'` (salzig)
    * wenn `sweet` `True` und `salty` `True`:
      · `string` `'confused'` (verwirrt)
- Erklären Sie bitte **kurz**, was ein `bool`scher Datentyp ist, und welche Werte dieser annehmen kann.

Lösung:

Programmieren Sie bitte:

- (B4.1) `condition.py`

Beschreiben Sie anhand des Programms bitte **kurz**:

- (B4.2) `condition.py` in der Kommandozeile ausführen
- (B4.3) Ausgabe des Programms auf der Kommandozeile

Erklären Sie bitte **kurz**:

- (B4.4) `bool`scher Datentyp

Aufgabenteil C - Fallbeispiele mit Klassen und Objekten

Aufgabe C1 - Studierendenverwaltung

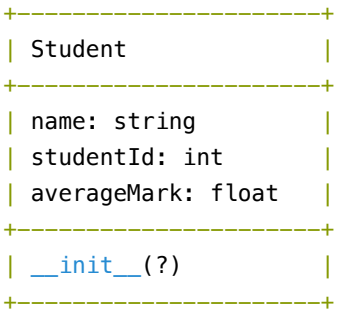
Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
C	1	30	[ ]

Problem:

Erstellen Sie ein kommentiertes Python-Programm in der Datei `student.py` und `main.py`, das

- die Klasse (`class`) `Student` (Student) in `student.py` definiert.
  - Ein Student hat die Attribute
    - Name (Vor- und Zuname) `name` als `string`,
    - Matrikelnummer `studentId` als `int`, und
    - Durchschnittsnote `averageMark` der Student*innen als `float`.
  - Methoden über `__init__()` hinaus sind nicht notwendig, Getter und Setter müssen nicht programmiert werden, diese sind 'automatisch' vorhanden.
  - Attribute müssen nicht als `private` deklariert werden.
  - Definieren Sie `__init__()` selbst, ist also nicht automatisch vorhanden.

Klassendiagramm:



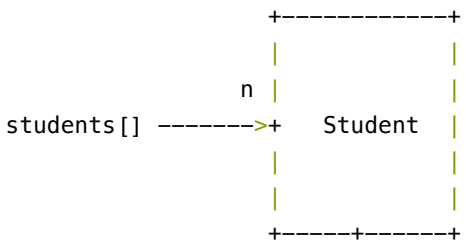
In dem Programm soll

- eine Liste `students` (Studenten) in `main.py` erstellt und mit Daten befüllt werden.

Die Liste hat

- zwei Objekte (`object`) als Elemente, über einen Konstruktor `__init__(?)` initialisiert mit:
  - Name 'Maxine Muster', Matrikelnummer `2`, Durchschnittsnote `1.0`,
  - Name 'Bert Beispiel', Matrikelnummer `1`, Durchschnittsnote `2.0`.

Diagramm:



In dem Programm soll

- die Durchschnittsnote aller Student*innen (`totalAverageMark`) aus der Liste `students` als `float` in `main.py`

berechnet werden.

- Die Berechnung kann direkt oder als Funktion umgesetzt werden.
- Als Formel soll `(Summe aller Durchschnittsnoten / Anzahl Studenten)` angewendet werden.
- Der Durchschnitts aller Student*innen soll auf der Kommandozeile mit `print()` ausgegeben werden.
- **Klassen, Methoden und Attribute über die hier gefragten Inhalte sind nicht relevant**
- **Dateien und Module müssen nicht angegeben werden, Importe werden ebenfalls angenommen und müssen nicht angegeben werden**

Lösung:

Programmieren Sie bitte **mit** wenigen erläuternden Kommentaren:

- (C1.1) `student.py`
- (C1.2) `main.py`

Beschreiben Sie anhand des Programms bitte **kurz**:

- (C1.3) `main.py` in der Kommandozeile ausführen
- (C1.4) Ausgabe des Programms auf der Kommandozeile

Aufgabe C2 - E-Auto Batterie

Aufgabenteil	Aufgabe	Erreichbare Punkte	Erreichte Punkte
C	2	30	[ ]

Problem:

Erstellen Sie ein kommentiertes Python-Programm in der Datei `ecar.py` und `main.py`, das

- die Klasse (`class`) `ECar` (E-Auto) in `ecar.py` definiert.
- Die Klasse E-Auto hat die Attribute
  - Name `name` als `string`,
  - Batterieladung `batteryCharge` in % als `int`, mit einem
    - Wertebereich von `0` % bis `100` %, die nicht unter- oder überschritten werden können, und
    - der Standardwert liegt bei `100` %, der in der `__init__()`-Methode festgelegt werden soll.
  - Ein E-Auto hat die Methoden
    - `charge()` (Laden), mit dem Parameter `amount` (Menge) in % als `int`,
      - mit dem Minimum `0` %, nur positives Laden ist möglich (kein entladen),
      - mit dem Maximum `100` %, kein Überladen ist möglich.
      - Die Ausgabe des Ladestandes ist mit einer Meldung auf der Kommandozeile mit `print()` möglich.
    - `drive()` (Fahren), mit dem Parameter `distance` (Menge) in Kilometer als `int`,
      - pro Kilometer Fahrt werden `1` % Batterieladung verbraucht, vorwärts wie rückwärts.
      - Bei `0` % Batterieladung ist kein weiteres Fahren möglich, deshalb Ausgabe einer Meldung auf der Kommandozeile mit `print()`, es muss nur pro ganzer Fahrt, nicht pro einzelnen Kilometern geprüft werden (`charge >= distance`).
      - Ab `10` % Batterieladung wird eine Warnung auf der Kommandozeile mit `print()` ausgegeben, es muss nur pro ganzer Fahrt, nicht pro Kilometer geprüft werden.
    - Die Ausgabe der gefahrenen Kilometer und des Ladestandes nach einer Fahrt mit mit einer Meldung auf der Kommandozeile mit `print()`

Klassendiagramm:

```
+-----------------------------------+
| ECar                              |
+-----------------------------------+
| name: string                      |
| batteryCharge: int                |
+-----------------------------------+
| __init__(batteryCharge = 100: int) |
| charge(amount: int)               |
| drive(distance: int)              |
+-----------------------------------+
```

In dem Programm soll:

- ein `ECar` `ecar1` (E-Auto), in `main.py` erstellt und initialisiert
  - beispielhaft als Objekt `object` initialisiert über `__init__()` mit
  - Name 'Resla Toadster', Batterieladung `50` % (statt des Standardwertes)
- folgende Programmschritte in `main.py` umgesetzt werden:
  - `ecar1` fährt `42` km,
  - `ecar1` lädt `10` %,
  - `ecar1` fährt `10` km.
- Der Import (`import`)

  muss

  in diesem Beispiel angegeben werden.
  - Getter und Setter müssen nicht programmiert werden, diese sind 'automatisch' vorhanden.
  - Attribute müssen nicht als `private` deklariert werden.
- **Klassen, Methoden und Attribute über die hier gefragten Inhalte sind nicht relevant**
- **Dateien und Module müssen samt Importe angegeben werden**

Lösung:

Programmieren Sie bitte **mit** wenigen erläuternden Kommentaren:

- (C2.1) `ecar.py` :
- (C2.2) `main.py`

Beschreiben Sie anhand des Programms bitte **kurz**:

- (C2.3) `main.py` in der Kommandozeile ausführen
- (C2.4) Ausgabe des Programms auf der Kommandozeile für die Werte:
  - `ecar1` fährt `42` km,
  - `ecar1` lädt `10` %,
  - `ecar1` fährt `10` km.
