Einführung in die Programmierung

Prof. Dr. Franziska Matthäus
Prof. Dr. Matthias Kaschube
Dr. Karsten Tolle
Kenan Khauto

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Übungsblatt ÜE-07 Team Teil

Ausgabe: 04.12.2024
Abgabe: 21.12.2024
13:00 Uhr

Restaurant

Hinweis:

- *Es dürfen keine Lösungen aus dem Skript, dem Internet oder anderen Quellen abgeschrieben werden. Diese Quellen dürfen nur mit Quellenangaben verwendet werden und es muss ein hinreichend großer Eigenanteil in den Lösungen deutlich zu erkennen sein.*
- *Digitale Abgaben, die nicht im Format* ***.pdf*** *oder* ***.txt*** *für Texte oder* ***.py*** *für Code erfolgen, werden nicht bewertet. Bei Abgaben mehrerer Dateien müssen diese als* ***.zip*** *zusammengefasst werden.*
- *Achten Sie darauf die Variable* ***__author__*** *in allen Quellcode Dateien (.py) korrekt zu setzen (am Anfang des Quellcodes):* __author__ = "<Matr-Nr>, <Nachname>"
  Beispiel: __author__ = "1234567, Tolle" *bei Zweierteams*
  __author__ = "<Matr-Nr_1>, <Nachname_1>, <Matr-Nr_2>, <Nachname_2"
  … Leerstellen vor und nach dem „=" und Leerstelle nach dem Komma beachten, sowie keine spitzen Klammern verwenden, die dienen bei der Schreibweise oben zum Definieren von Variablen.
- *Außerdem muss Ihr(e) Name(n) in jeder abgegebenen* ***.pdf*** *und* ***.txt*** *Datei zu finden sein.*
- *Abgaben der Dokumentation, die per Hand geschrieben und eingescannt werden, sind nur in zuvor abgesprochenen Ausnahmefällen erlaubt.*
- *Datei- und Ordnernamen sollen keine Umlaute, diakritische oder Sonderzeichen, mit Ausnahme des Unterstrichs in der Namesmitte, enthalten! (Insb. MAC Nutzer sollten aufpassen, dass nicht zusätzliche Dateien im ZIP-Ordner hinzugefügt wurden! … diese ggf. löschen!)*
- *Beim Programmieren und Kommentieren halten Sie sich an die Regeln im Programmierhandbuch, siehe Moodle-Kurs (Programmierhandbuch WiSe 24/25 (Style Guide)). Im Zweifelsfall gilt PEP 8.*
- *Bitte auch darauf achte, dass Ihre .py Dateien lauffähig sind. Es ist den Tutoren nicht zuzumuten Abgaben zu Debuggen. Probleme, auch was vielleicht nicht geht, kann auskommentiert und mit Kommentaren versehen werden.*
- *Lösungen sind eigenständig zu erstellen, ohne Zuhilfenahme von generativer KI (z. B. ChatGPT). In der Klausur wird dieses Hilfsmittel ebenfalls nicht zur Verfügung stehen.*
- *Eine Tutorin/Tutor oder die Dozentin/der Dozent kann ein Code-Review verlangen, bei dem Sie Ihren Code erklären und rechtfertigen müssen.*

Hinweis: Dieses Aufgabenblatt ist mit einer Bearbeitungszeit von zwei Wochen dafür ausgelegt in einem Zweierteam gelöst zu werden. Neben der Implementierung wird bei der Bearbeitung Wert auf die folgenden Punkte gelegt: Dokumentation (auch die Benutzerführung sollte erklärt werden (**Benutzerhandbuch**)), Docstrings (http://www.python.org/dev/peps/pep-0257), Strukturierung und Einhalten des Style-Guides.

Das Blatt muss in Zweiterteams bearbeitet werden. Teamarbeit ist auch Teil des Lerninhaltes. Die Tutor*innen sind angehalten in den Tutorien die Teambildung zu unterstützen.

Die Teams sollten aus einem Tutorium (bzw. aus Tutorien des*r gleichen Tutor*in) stammen. Wird hiervon abgewichen, muss dies mit einem der beiden Tutor*innen abgesprochen sein. Diese*r muss also zustimmen. Die Abgabe muss dann durch den Teilnehmenden erfolgen, der diesem Tutor*in zugeordnet ist. Die finale Bewertung durch den Tutor*in erfolgt dabei erst nach einem erfolgreichem Code-Review (ohne dies gibt es keine Punkte!). Dort wird geprüft, ob von allen Teammitgliedern der abgegebene Inhalt auch verstanden wurde. … es geht also nicht darum die Arbeit einfach aufzuteilen! Jede*r muss alles verstehen! Nur in extremen Ausnahmefällen und nach Rücksprache mit und Genehmigung durch dem*r Tutor*in, sind auch Dreierteams gestattet.

Pro Team nur eine Abgabe in Moodle!

∑ 14 Punkte

Aufgabe – Restaurant

Ein Restaurant soll eine Software bekommen, welche die Bestellungen verwaltet und die Rechnungen erstellt. Es soll für jeden Tisch einzeln betrachtet werden können.

Die Produkte (Gerichte, Beilagen, Getränke, …), welche angeboten werden sollen, sollen über eine Datei eingelesen und im System verwendet werden können. Eine Beispielhafte *food.csv*-Datei findet ihr im Moodle-Kurs unter demselben Namen.

Die CSV-Datei kann nach Belieben erweitert werden.

Implementierung

Wie oben erwähnt, sollten Bestellungen pro Tisch aufgenommen werden. Für jeden Tisch sollen die bestellten Produkte vermerkt werden. Natürlich kann es Nachbestellungen geben. Dabei soll es auch möglich sein, Sonderwünsche für die Gerichte zu ergänzen. Bei Sonderwünschen, welche etwas hinzufügen, z.B. „extra viel Käse“ soll sich der Preis pro Sonderwunsch um 1 Euro erhöhen. Sonderwünsche, welche etwas weglassen, z.B. „ohne Käse“, sollen den Preis nicht verändern. Am Ende soll eine Rechnung erstellt werden können. Um Unstimmigkeiten lösen zu können, muss es noch bis zur Bezahlung der Rechnung möglich sein, einzelne Posten zu stornieren oder zu ergänzen.

Gerne können diese Grundregeln erweitert werden, auch sind Verfeinerungen denkbar (Bestellungen pro Person statt pro Tisch). Dies ist aber optional.

Die Implementierung soll Objektorientiert erfolgen!

User Interface

Erstellen Sie eine passende Konsolen-UI, was dazu dient Ihr Programm in der Konsole zu nutzen.

Folgendes muss erfüllt werden:

- Aussagekräftige Erklärungen auf der Konsole (Eingaben und Ausgaben).
- Das Programm muss stabil sein! Das bedeutet, es soll bei einer Fehleingabe nicht abgebrochen werden.
- Das Programm soll mit einer bestimmten Eingabe gestoppt werden können.
- Rechnungen sollten in einen *.txt*-Datei dauerhaft gespeichert werden.

Dokumentation

Bitte die Präzisierung der Aufgabenstellung mit einer eine Beschreibung, was Sie umsetzen wollen, in der **Dokumentation** festhalten. Erstellen und nutzen Sie auch ein **UML-Klassendiagramm** in der Dokumentation. Im UML-Klassendiagramm sollen auch die Datentypen für die Variablen und für die Funktionen (Eingabe und Rückgabe) berücksichtigt werden. Schreiben Sie ein Python 3.x **Programm**, welches die Funktionalitäten der einzelnen Klassen umsetzt. Im optimalen Fall kann das UML-Klassendiagramm auch zur Umsetzung genutzt werden, Sie sollten möglichst also ein Tool (wie Umbrello) verwenden☺. Die Dokumentation soll auch ein Benutzerhandbuch enthalten, welches genau erklärt, wie das Programm funktioniert (aus Sicht eines Nutzers).

Erstellen Sie sinnvolle Testfälle für dieses Aufgabenblatt. In diesem Blatt sind Sie jedoch frei, wie Sie diese umsetzen. Es müssen also keine Doctests sein. Auch muss nicht jede Methode einer Klasse mit Testfällen getestet werden (auch wenn das nicht verkehrt wäre). Gehen Sie in der Dokumentation darauf ein, was getestet wurde und auch was und warum Sie beim Testen vernachlässigt haben.

Hinweis:

- Einiges was Sie für dieses Aufgabenblatt benötigen, wird erst im Laufe der Bearbeitungszeit in den Vorlesungen besprochen. Insb. Objektorientierte Programmierung kommt erst in der Vorlesung am 13.12.2024! Nutzen Sie trotzdem die Zeit vorher, um sich das Team zu bilden, die Aufgabenstellung zu überdenken und z.B. das Lesen und Schreiben von Dateien zu üben. Außerdem gibt es ja noch das Blatt 7 Teil 2, welches jeder alleine umsetzen soll und was bereits am 14.12.2024 abzugeben ist.