Übungsblatt 3

Hinweise

- **Abgabe:** Digital bis zum 04.12.2023, 12:00 Uhr im ILIAS, als *zip*-Datei mit einem Ordner mit *py*-Dateien. Benennt eure Abgabedatei entsprechend mit **blatt3_nachname.zip**. Bewertet wird die späteste Version eurer Abgabe, deren Upload vor der Deadline liegt.
- **Einzelarbeit:** Die Aufgaben auf diesem Übungsblatt sind ausschließlich in Einzelarbeit zu lösen und abzugeben. Es werden keine Gruppenarbeiten akzeptiert. Bei offensichtlich abgeschriebenen Lösungen werden *beide* Lösungen mit 0 Punkten bewertet.
- Ihr benötigt mindestens 10 Punkte, um das Übungsblatt zu bestehen. Über alle drei Übungsblätter hinweg müsst ihr mindestens 45 Punkte erreichen, d.h. im Schnitt 15 Punkte pro Blatt.

Auf diesem Blatt sollt ihr eine interaktive Python-Anwendung entwickeln und an die Datenbank der letzten beiden Übungsblätter anbinden. Hierbei dürft ihr natürlich Teile des dort geschriebenen Codes weiterverwenden. Da einzelne Aufgabenteile aufeinander aufbauen, ist es sinnvoll, das Blatt erst **vollständig** zu lesen und anschließend mit der Bearbeitung zu beginnen. Ausgabeformate usw. sind an einigen Stellen nicht komplett spezifiziert. Hier sollt ihr selbst eine Lösung entwerfen, die die geforderten Informationen transportiert. Das ist so gewollt und Teil der Aufgabe. Ihr könnt natürlich trotzdem im Forum nach Anregungen fragen.

Euer Programm darf aus beliebig vielen Python-Dateien bestehen, muss allerdings einen klar definierten Einstiegspunkt `main.py` haben, der mit `python main.py` ausgeführt werden kann. Ihr dürft dem Programm Bibliotheken und Abhängigkeiten hinzufügen, müsst diese dann aber in einer `requirements.txt` dokumentieren (und ggf. Support leisten, wenn euer Programm auf meinem Rechner nicht läuft ☺).

Stellt für alle Aufgaben sicher, dass euer Programm im Fehlerfall nicht unkontrolliert abstürzt, sondern Fehler so gut wie möglich behandelt und sinnvolle Fehlermeldungen ausgegeben werden. Terminiert das Programm nur, wenn keine Fehlerbehandlung möglich ist, anderenfalls fordert den:die Nutzer:in zur erneuten Eingabe eines korrekten Befehls auf.

Aufgabe 1 — 10 Punkte

Die Nutzer:innen-Schnittstelle eures Programms soll eine einfache *Read-Eval-Print-Loop (REPL)* sein. Es sollen also textuelle Befehle eingelesen und anschließend ausgewertet werden, um dann das Ergebnis auszugeben. Ihr sollt dabei (mindestens) die folgenden Befehle unterstützen:

- `help` um eine Erklärung zur Nutzung des Programms anzuzeigen,
- `query $SQL`, wobei `$SQL` hierbei eine beliebige SQLite-Anfrage sein kann. Diese Anfrage soll auf der Datenbank des Programms ausgeführt und das Ergebnis (oder zumindest die ersten Zeilen des Ergebnisses) in lesbarer Form ausgeben.

- `describe database` um eine kurze Erklärung der Datenbank und ihrer Tabellen auszugeben (inkl. beispielsweise der Anzahl Zeilen),
- `add directory $DIR` um ein Verzeichnis mit Artikeln in die Datenbank des Programms zu laden,
- `remove directory $DIR` um ein alle Artikel, die sich im übergebenen Verzeichnis befinden aus der Datenbank zu entfernen,
- `lookup $KEYWORD` um die Titel aller Artikel auszugeben, die `$KEYWORD` in ihrem Volltext enthalten, und als letztes
- `example [1..8]`, die jeweils eure Query für die entsprechende Aufgabe auf dem letzten Blatt anzeigen, auf der aktuellen Datenbank ausführen und das Ergebnis präsentieren.

Aufgabe 2 — 10 Punkte

Entwerft und implementiert ein *Data Access Object (DAO)* für `article`-Objekte bzw. deren Entsprechungen in der Datenbank. Dieses soll das *CRUD*-Interface implementieren, allerdings benötigen wir keine *update*-Methode.

Euer Programm soll eine eigene Datenbank verwalten, bspw. `articles.db`. Diese ist zu Beginn leer und hat das gleiche Schema wie auf den bisherigen Übungsblättern. Sie muss beim ersten Programmstart natürlich angelegt werden, Daten können dann über das Programm selbst eingefügt werden.

- `create(...)` bekommt ein komplett initialisiertes `Article`-Objekt übergeben und schreibt dieses in die Datenbank. Achtet hierbei darauf, dass einige Felder der `Article`-Klasse wiederum Objekte sind und entpackt werden müssen, und dass ihr mehr als ein `INSERT`-Statement benötigt.
- `createMany(...)` ist im Prinzip analog, bekommt aber nicht nur ein `Article`-Objekt, sondern ein `Iterable[Article]`, sodass ihr für eure `INSERT`-Statements `executemany(...)` statt `execute(...)` verwenden könnt.
- `read(...)` bekommt eine Artikel-ID übergeben, liest die zugehörigen Daten aus der Datenbank aus und gibt ein vollständig initialisiertes `Article`-Objekt zurück.
- `readMany(...)` analog.
- `delete(...)` bekommt ebenfalls eine Artikel-ID übergeben und entfernt den zugehörigen Artikel aus der Datenbank. Stellt hierbei sicher, dass auch alle anderen Tabellen aufgeräumt und keine Fremdschlüsselbeziehungen verletzt werden.
- `deleteMany(...)` analog.

Aufgabe 3 — 5 + 3* Punkte

Fügt eurer REPL einen weiteren Befehl `plot $SQL` hinzu. Dieser soll die übergebene SQL-Anfrage ausführen und prüfen, ob das Ergebnis

1. aus genau 2 Spalten besteht,
2. die erste davon einen ISO-Timestamp enthält und

3. die zweite ein numerisches Attribut.

Ist das nicht der Fall, darf die Methode eine Fehlermeldung ausgeben und nichts weiter tun. Sind alle Bedingungen erfüllt, soll ein Plot des Ergebnisses im `output`-Ordner erzeugt werden. Art und Gestaltung des Plots sind dabei euch überlassen. Ihr könnt Bibliotheken wie *matplotlib* oder *seaborn* verwenden und euch dort bei den Gestaltungselementen frei bedienen. Zur Erfüllung der Aufgabenstellung sind einfache Line-Plots ausreichend, besonders schöne oder sogar dynamisch an die Anfrage angepasste Plots können aber zu Bonuspunkten führen.