Probeklausur
Informatische Werkzeuge in den Geistes- und Sozialwissenschaften II

19. Juni 2018

The „Lösungen“ der Aufgaben in diesem Dokument sollen den Studierenden als Anfangspunkt für die Beantwortung der Aufgaben dienen. Trotz aller Bemühungen kann es zu unvollständigen oder sogar Fehlern in den „Lösungen“ kommen. Da die Korrektur und Benotung der gestellten Aufgaben niemals lediglich auf einen „Vergleich mit der Musterlösung“ hinausläuft, ist dies auch nicht so schlimm. In jedem Fall sollten die Studierenden die Lösungen nachvollziehen und im Prinzip mittels des Lehrstoffs selbst verifizieren können.

Sollten Sie „Lösungen“ finden, die Sie nicht verstehen oder sogar für fehlerhaft halten diskutieren Sie diese am besten mit den Tutoren oder auf dem Kursforum und benachrichtigen Sie die Lehrenden.

1 Versionskontrolle

Aufgabe 1.1 Was ist ein “Issue” im Kontext von Versionskontrolle und was zeichnet ein gutes Issue aus? 3 min

Lösung:Ein *Issue* ist ein (normalerweise) kurzer Text assoziiert zu einem Softwareprojekt. Issues sind entweder Berichte über Fehler in der Software (“bug reports”) oder Vorschläge für zukünftige Erweiterungen (“feature requests”).

Ein gutes Issue sollte alle relevanten Details beinhalten (z.B. wie ein Bug wiederholbar verursacht werden kann oder was *genau* an Verhalten vorgeschlagen wird) ohne zu vage oder ungenau zu werden.

Aufgabe 1.2 Warum wird von Versionskontrollsystemen verlangt, dass Entwickler'innen eine sogen. “commit message” formulieren, wenn sie Änderungen hochladen? 2 min

Lösung:Commit messages sind Teil der Dokumentation und der Kommunikation zwischen Teammitgliedern. Sermöglichen einen schnellen Überblick (sowohl mit Augen als auch mit Suchfunktionen) über welche Commits mit welchen Features oder eventuellen Bugs zu tun haben könnten.

Aufgabe 1.3 Was ist ein “Diff” zwischen zwei Textdateien? Welche Rolle spielen Diffs in der Versionskontrolle? 3 min

Lösung:Ein *Diff* zwischen zwei Dateien ist eine Liste von Unterschieden (engl.: *differences*) (auf Zeilenbasis). Ein Diff enthält also alle Informationen darüber, wie die erste Datei geändert werden müsste, um identisch zur Zweiten zu sein.

Hier ist ein Beispiel:

```
jbetzend@turing :: 11:15:05 :: ~/Desktop > # Here are some files
jbetzend@turing :: 11:15:12 :: ~/Desktop > cat a.txt
This line is in both files.
This one isn't.
This one is a bit different.
jbetzend@turing :: 11:15:16 :: ~/Desktop > cat b.txt
This line is in both files.
This one is slightly different.
jbetzend@turing :: 11:15:18 :: ~/Desktop > # And now the Diff
jbetzend@turing :: 11:15:25 :: ~/Desktop > diff a.txt b.txt
2,3c2
< This one isn't.
< This one is a bit different.
---
> This one is slightly different.
```

Versionskontrollsysteme benutzen Diffs um anzugeben, welche Änderungen eine Datei erfahren hat. So benötigt ein Repository deutlich weniger Speicherplatz als wenn jede Version der Datei (Snapshot) gespeichert werden müsste.

Aufgabe 1.4 Beatrice Beispielperson führt in ihrer Shell folgende Befehle aus, um ein `git`-Repository zu klonen: 5 min

```
git clone https://gitlab.cs.fau.de/iwgs-ss19/collaboration.git
cd collaboration
```

Sie verändert danach lokal die Datei "users.txt" und führt folgende Befehle durch um ihre Änderungen zu veröffentlichen.

```
git commit -m "changes to users.txt"
git push
```

Leider hat dies nicht den gewünschten Effekt. Was hat Beatrice falsch gemacht? Welcher Befehl hätte ausgeführt werden müssen?

Lösung:Beatrice hat alles richtig gemacht, sie hat lediglich vergessen, ihre Änderungen für den commit zu markieren (staging).

Vor dem `git commit`-Befehl hätte sie noch `git add users.txt` ausführen müssen.

---

2 Datenbanken

Aufgabe 2.1 Was bezeichnet der Begriff "Cursor" im Kontext von Datenbanksystemen? 2 min

Lösung:Ein Cursor ist ein benamtes Objekt, dass die Menge der Ergebnisse einer Anfrage an eine Datenbank enthält. Der Cursor verhält sich dabei wie eine (virtuelle) Tabelle in der Datenbank.

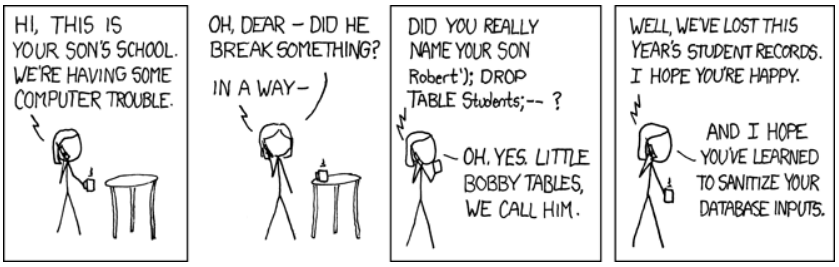
Intuition: Cursors erlauben es Programmierer'innen, die gleiche Datenbankanfrage mehrmals zu benutzen.

Aufgabe 2.2 Beschreiben Sie das Prinzip hinter einer "SQL injection attack" und erklären Sie, wie bereits beim Designprozess dagegen vorgegangen werden kann. 3 min

Lösung:Viele Programme, die mit Datenbanken interagieren, brauchen Input direkt von den Benutzer'innen des Programms. Bei einer SQL Attacke nutzt die angreifende Partei diesen Fakt, um direkt SQL-code auf der Datenbank auszuführen, was zu Datenverlust oder Datenlecks führen kann.

Dies kann verhindert werden, indem das Programm niemals direkt den Input benutzt, sondern immer erst überprüft, ob es legitimer Input ist (dieser Prozess wird "sanitising user input" genannt.).

Siehe auch:



Aufgabe 2.3 Erklären Sie kurz die SQL-Befehlskonstrukte `INSERT INTO`, `DROP TABLE IF EXISTS` und `SELECT FROM WHERE` und geben Sie jeweils ein einfaches Beispiel. 5 min

Lösung:

- `INSERT INTO` erwartet den Namen einer Tabelle und ein `VALUES(...)` statement, das die Daten beschreibt, die in die entsprechende Tabelle eingefügt werden sollen.
  Beispiel: `INSERT INTO BooksVALUES ('Twain', 'Mark', '1835', '1910', 'Huckleberry Finn', '1986','Penguin USA', 'NY')`

- `DROP TABLE IF EXISTS` erwartet einen Namen einer Tabelle und dropt diese Tabelle, falls sie existiert. Nützlich, um Fehler beim anlegen einer Tabelle zu verhindern.
  Beispiel: `DROP TABLE IF EXISTS Images`

- `SELECT «columns» FROM «table» WHERE «condition»` erwartet eine Liste von Spaltennamen, den Namen einer Tabelle und eine boolsche (i.e. `== True` oder `== False`) Bedingung und liefert genau die Werte aus der Tabelle zurück, die in den benannten Spalten bei Einträgen (Zeilen) stehen, die die boolsche Bedingung wahr werden lassen.
  Beispiel: `SELECT Title FROM Books WHERE Year = 2000`

Aufgabe 2.4 Benjamin Beispielperson erstellt mit folgendem Befehl eine Tabelle in der Datenbank "bookdatabase.db". 4 min

```
CREATE TABLE Books (Author TEXT, Title TEXT, Year INT, Cover BLOB);
```

Die Tabelle wird danach von einem Programm automatisch befüllt (*nicht* Teil dieser Aufgabe!). Geben Sie eine SQL-Anfrage an, die alle Bücher in der Tabelle Books in der obigen Datenbank zurück gibt, welche im Jahr 2000 erschienen sind.

Lösung:Dies ist ein simples `select-from-where` statment, wie in Aufgabe 2.3 beschrieben. Wir wollen hier aber ganze Einträge und nicht nur einzelne Spalten, also selecten wir `*` (= alle Spalten).

Das gesuchte Statement ist: `SELECT * FROM Books WHERE Year = 2000`

Aufgabe 2.5 Schreiben Sie ein python-Programm, das eine Textdatei mit dem Namen "loeschen.txt" einliest (welche genau einen Titel pro Zeile enthält), sich dann mit der obigen Datenbank verbindet und alle Einträge aus der Tabelle Books löscht, die einen Titel haben, der in der Textdatei vorkommt. 10 min

Lösung:

```
import sqlite3

# Create connection and get cursor object.
con = sqlite3.connect("bookdatabase.db")
cur = con.cursor()

with open("loeschen.txt","r") as file:
    for line in file:
        # Possibly dangerous! We assume this input to be safe here.
        sql_delete = "DELETE FROM Books WHERE Title = " + line
        cur.execute(sql_delete)

# Commit changes and close connection.
con.commit()
con.close
```

3 Bilder und Bildmanipulation

Aufgabe 3.1 In digitalen Bildformaten werden Pixel oft in einem Vier-Kanal-Format (RGBA) abgespeichert. Erklären Sie den Aufbau und die Bedeutung dieser Kanäle. 3 min

Lösung:Die Kanäle R,G und B tragen Informationen (z.B. als Zahl zwischen 0 und 255) über die Farbe des Pixels, aufgeteilt in **R**ot, **G**rün und **B**lau.

Der A-Kanal gibt den **A**lpha-Wert des Pixels an, also wie "durchsichtig" das Pixel sein soll, falls mehrere Bilder aufeinander liegen sollten.

Aufgabe 3.2 Warum benötigt ein Bild in Graustufen bei identischer Auflösung weniger Speicherplatz als das gleiche Bild in Farbe? 2 min

Lösung:Grautöne im RGB-Spektrum sind daran zu erkennen, dass die Werte aller Farbkanäle identisch sind ($R = G = B$). Wir wir also wissen, dass es sich um ein Graustufenbild handelt, müssen wir nur einen statt drei Werten speichern.

Aufgabe 3.3 (Raster- und Vektorgrafiken)
Erklären Sie kurz den Unterschied zwischen Vektor- und Rastergrafiken. Warum macht das Konzept von "Auflösung" nur bei einem der beiden Sinn? 4 min

Lösung:Eine *Raster*grafik speichert Informationen in einem großen Gitter (oder Raster) von einzelnen Pixeln. Die Dimensionen dieses Gitters ergeben die Auflösung.

Eine Vektorgrafik hingegen speichert keine Pixel, sondern Informationen über geometrische Formen (z.B. Zentrum, Radius und Strichdicke eines Kreises, Start und Ziel und Breite einer Linie, ...), die zusammen das Bild ergeben. Hier gibt es keine eindeutige Auflösung, weil wir diese Formen beliebig skalieren (zoomen) können.

Aufgabe 3.4 Schreiben Sie eine python-Funktion mit folgender Signatur: 10 min

def forgetRed(pixels):

Der Parameter dieser Funktion ist ein zwei-dimensionales Array (i.e. eine Liste von Listen) von Pixeln. Ein Pixel ist für die Zwecke dieser Aufgabe ein 3-Tupel (R,G,B) mit numerischen Werten für die Rot-, Grün- und Blauanteile des Pixels.

Ihre Funktion soll das gleiche Bild im gleichen Format zurück geben, jedoch sollen dabei alle Rotwerte auf 0 reduziert werden.

Lösung:

```
example = [[(1,11,21),(1,21,41)],[(32,16,1),(52,26,1)]]

def forgetRed(pixels):
    # Das gesamt Bild ist eine Liste von Listen.
    # Dies ist unsere aeussere neue Liste.
    newImg = []
    for i in pixels:
        # Innere neue Liste fuer jede in Liste in der aeusseren.
        newList = []
        for j in i:
            # Konstant 0 fuer alle Rotwerte, sonst gleich Input.
            newPixel = (0, j[1], j[2])
            # Haengt das neue Pixel an die neue innere Liste an.
            newList.append(newPixel)
        # Haengt die neue innere Liste an die neue aeussere an.
        newImg.append(newList)
    return newImg

print(forgetRed(example))
```
