Using Codeboard and Mantra to Run and Grade SQL Assignments

Liam Tarr
Supervisor: Dr. Ramon Lawrence

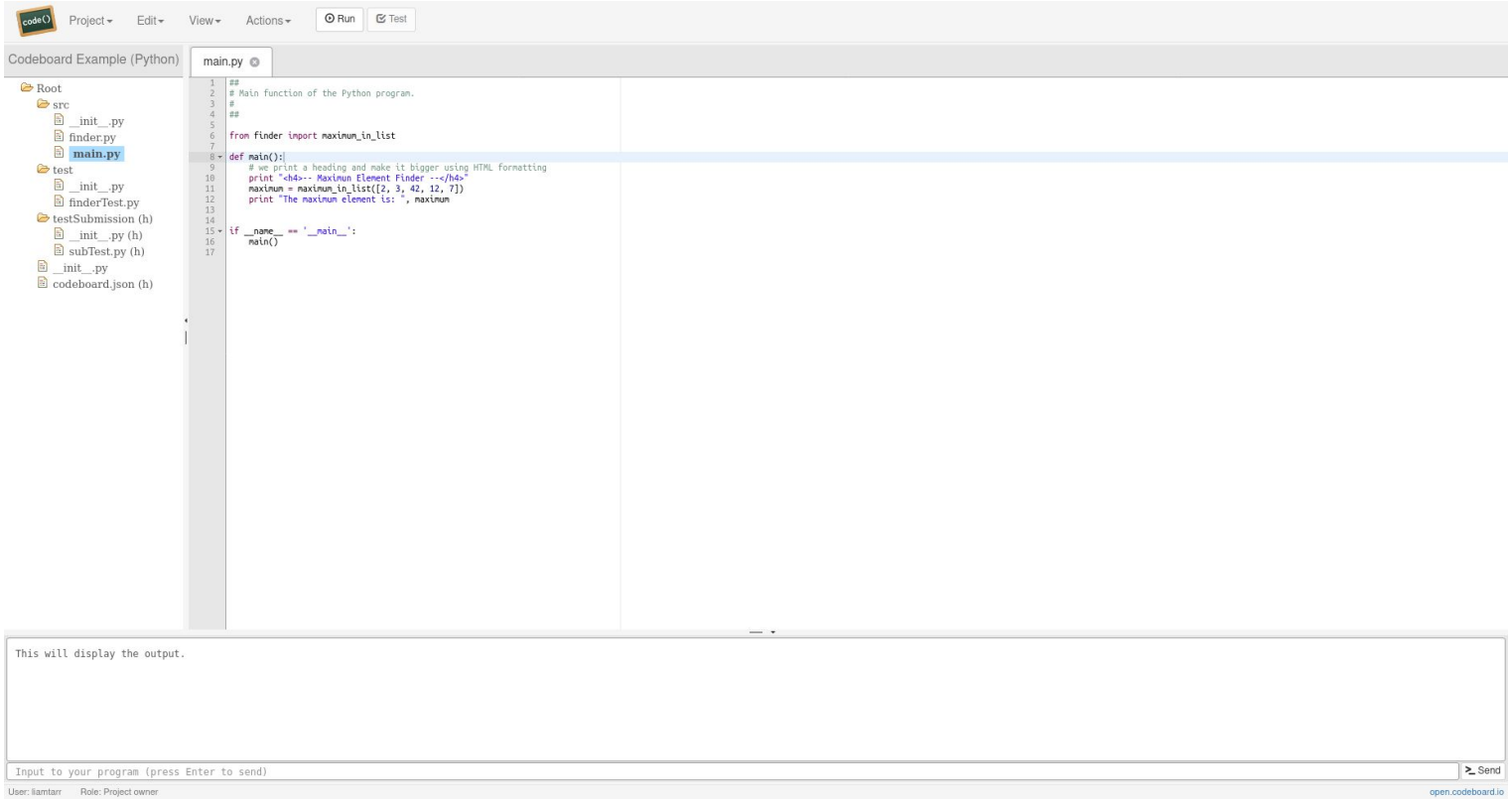
Introduction

- SQL is an important skill
- DBMS may be hard for students to set up
- Large classes (150+ students)
- Grading assignments is time consuming
- Is there a better way?

Codeboard

- Codeboard is a "web-based IDE to teach programming in the classroom."
- Supports Python, Java, C, and more out of the box.
- Teachers create exercises and share with students.
- Students can run and test code.
- Students submit their assignment for grading.
- Automatically graded using unit tests
- Teachers can inspect students code

What the Codeboard IDE Looks Like



Mantra

- The “middleman” between Codeboard and Docker.
- Web service for compiling and running programs.
- Responsible for validation, running and grading assignments.

How Does it All Work?

- User sends request to Codeboard
- Codeboard sends http request to Mantra
- Mantra can create Docker container
- Container runs program and sends results back to Mantra
- Mantra then sends results back to Codeboard
- User sees results

Using Python to Run and Grade SQL Assignments

- Use the fact that Python with unit tests works out of the box.
- Python has built in sqlite3 support
- Various methods to populate and select from database using SQL
- Create multiple questions using text file
- Use “unittest” module to test and grade assignments

Example Question

code() Project Edit View Actions Run Test

SQL using Python

- Root
  - src (h)
    - SQLService.py (h)
    - __init__.py (h)
    - file_input.py (h)
    - main.py (h)
  - test (h)
    - __init__.py (h)
    - test.py (h)
  - testSubmission (h)
    - __init__.py (h)
    - subTest.py (h)
  - Question1.txt
  - Question2.txt
  - Question3.txt
  - __init__.py (h)
  - codeboard.json (h)

Question1.txt

```
# Insert question text at the top of the file here, all lines starting with '#' will not be read into compiler
# To add additional questions, create a new file (Project -> Add file) and name it Question#.txt, where # is the previous number + 1
# Students can put their SQL query where in the text file.
# Example Question: "Create a A Recipe table where each recipe is identified by a field called id that is an integer.
# Other attributes include name (string up to 40 characters), author id (integer), and directions (string up to 255 characters).
# Make all foreign keys set to null on delete and no action (generate error) on update."

CREATE TABLE Recipe (
    id        int          NOT NULL,
    name        VARCHAR(40),
    authorId    int,
    directions  VARCHAR(255),
    PRIMARY KEY (id),
    FOREIGN KEY (authorId) REFERENCES Author(id)
            ON DELETE SET NULL ON UPDATE NO ACTION
)
```

Example of Test Output

```
Number of passing tests: 3
Number of failing tests: 0

--- Details ---

test_Create_Table (Root.test.test.subTest) ... ok
test_Populate_Table (Root.test.test.subTest) ... ok
test_Select_Statements (Root.test.test.subTest) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.002s

OK
```

Note: There were actually 3 questions in total on project

How are Tests Written?

```python
class subTest(unittest.TestCase):

    # Question 1: Test Create Table 'Recipe'
    # We are testing the user contents (Question1.txt) versus the q1Table variable
    def test_Create_Table(self):
        q1Table = """CREATE TABLE Recipe (
        id      int         NOT NULL,
        name        VARCHAR(40),
        authorId    int,
        directions  VARCHAR(255),
        PRIMARY KEY (id),
        FOREIGN KEY (authorId) REFERENCES Author(id)
        ON DELETE SET NULL ON UPDATE NO ACTION
        )"""
        self.assertEqual(SQLTable(q1Table, "Recipe"), SQLTable(getUserQuestionContent(1), "Recipe"))

    # Question 2: Test Delete Statement on 'Recipe'
    def test_Populate_Table(self):
        query1 = """CREATE TABLE Recipe (
        id      int         NOT NULL,
        name        VARCHAR(40),
        authorId    int,
        directions  VARCHAR(255),
        PRIMARY KEY (id),
        FOREIGN KEY (authorId) REFERENCES Author(id)
        ON DELETE SET NULL ON UPDATE NO ACTION
        )"""
        query2 = """INSERT INTO Recipe VALUES (100,'Cookies',1,'Mix butter, flour, milk, eggs, and sugar.  Then hope for the best.');"""
        query3 = """INSERT INTO Recipe VALUES (200,'Bread',2,'Knead flour with milk and eggs.  Bake at 450F or until brown.');"""
        qlist = (query1, query2, query3)
        queryForOutput = "SELECT * FROM Recipe"

        self.assertEqual(SQLPopulate(qlist, "DELETE FROM Recipe WHERE authorId = 2;", queryForOutput, "Recipe"), SQLPopulate(qlist, getUserQuestionContent(2), queryForOutput, "Recipe"))

    #Question 3: Test Create a SELECT statement
    def test_Select_Statements(self):
        query1 = """CREATE TABLE Recipe (
        id      int         NOT NULL,
        name        VARCHAR(40),
        authorId    int,
        directions  VARCHAR(255),
        PRIMARY KEY (id),
        FOREIGN KEY (authorId) REFERENCES Author(id)
        ON DELETE SET NULL ON UPDATE NO ACTION
        )"""
        query2 = """INSERT INTO Recipe VALUES (100,'Cookies',1,'Mix butter, flour, milk, eggs, and sugar.  Then hope for the best.');"""
        query3 = """INSERT INTO Recipe VALUES (200,'Bread',2,'Knead flour with milk and eggs.  Bake at 450F or until brown.');"""
        qlist = (query1, query2, query3)

        self.assertEqual(SQLSelect(qlist, "SELECT * FROM Recipe", "Recipe"), SQLSelect(qlist, getUserQuestionContent(3), "Recipe"))
```

SQLTable

```
def SQLTable(sql, tbname):
    if(sql.split(' ')[2] != tbname): #Compares the table name argument with the table name from the CREATE TABLE statement
        return "Did not find correct table name, please use CREATE TABLE (tablename);\nWe are looking for table name '" + tbname + "' but we found '" + sql.split(' ')[2] + "'"
    conn = sqlite3.connect(":memory:")
    cursor = conn.cursor()
    result = ""
    try:
        cursor.execute(sql)
        cursor.execute("pragma table_info('" + tbname + "')")
        result = cursor.fetchall()
    except:
        pass
    return result
```

Run SQL Directly

- Another possible approach is running SQL directly
- This means no Python!
- Need to add new language to Codeboard and Mantra
- How will we run it?

How to Add a New Language to Codeboard and Mantra

- Not a simple thing to do
- Create new dockerfile
- Think about project structure
- Important files, such as “codeboard.json”
- See paper for more detail on all the steps

Dockerfile for SQLite

Open *sqlite.docker ~/autoed/src/code_boi/codeboard_docker/docker_build_files Save

```
# Docker file to create a container for running MySQL.
# Will install MySQL for Ubuntu 14.04.
#
#
# To build this file execute (including "." at the end): docker build --file="sqlite.docker" --tag="cobo/sqlite" --rm=true .
#
# Note: we're only using a single RUN command to minimize the number of layers
# that Docker will create (every RUN commands creates a new layer). Commands are
# conncatenated with && and a new line is indicated with \
#
#
# author: Liam Tarr
# date: November 21, 2020
# version: 0.1

# all compilers run on cobo/ubuntu (make sure you've build that one already)
FROM cobo/ubuntu

RUN \
        # update the package manager
        apt-get update && \

        # install sqlite
        apt-get install -y --force-yes sqlite3 libsqlite3-deSSv

# set maintainer
MAINTAINER cobo <support@codeboard.io>
```

Plain Text Tab Width: 8 Ln 6, Col 2 INS

Example SQL Project Structure

```
Root
├── codeboard.json
└── src
    └── main.sql
```

Example SQL Input and Output

codeQ Project Edit View Actions Compile Run

SQLite Attempt 2

- Root
  - src (h)
    - main.sql
  - codeboard.json (h)

main.sql

```sql
DROP TABLE people;

CREATE TABLE people (
  id INTEGER PRIMARY KEY,
  first_name TEXT NOT NULL,
  last_name TEXT NOT NULL
);

INSERT INTO people (first_name, last_name)
VALUES
("Clarence", "Walmsley"),
("Tia", "Holloway");

SELECT * FROM people;
```

```
1|Clarence|Walmsley
2|Tia|Holloway
```

Ways to Improve

- Need to be able to test and grade assignments, not just run them
- Might need to use something like T-SQL
- Also, we need to be wary of students altering their own databases
- We could do this with permissions

Conclusion

- Manually grading SQL assignments in time-consuming
- It’s possible to automate the grading
- Codeboard and Mantra makes this possible
- We can run SQL through Python
- We can run SQL directly
- Running SQL directly still needs work to flesh out issues