SPL 191
Assignment 4

Published on: **6.1.2019**

Due date: **17.1.2018 23:59**

Responsible TA: Hagit Bachmat

1 General Description

BGU needs your help to schedule courses. You are required to implement a simulator of assigning classrooms to courses.
In this assignment, you will implement such a tool using Python and SQLite.

2 Method and Technical Description

You will build a `sqlite3` database that will hold the courses, students, and classrooms tables.
The database filename will be **`classes.db`**.
You will have 2 Python modules: **`create_db.py`** and **`schedule.py`**.

2.1 The Database Structure

The database **`classes.`**db has three tables. You can find examples in the "Examples" section of the assignment.

- `courses:` This table holds information of the courses. The columns are:

```
id INTEGER PRIMARY KEY
course_name TEXT NOT NULL
student TEXT NOT NULL
number_of_students INTEGER NOT NULL
```

```
class_id INTEGER REFERENCES classrooms(id)
course_length INTEGER NOT NULL
```

- `students: This` table holds the number of students per grade. The columns are:

```
grade TEXT PRIMARY KEY
count INTEGER NOT NULL
```

- `classrooms: This` table holds the location and the status of each class room. The columns are:

```
id INTEGER PRIMARY KEY
location TEXT NOT NULL
current_course_id INTEGER NOT NULL
current_course_time_left INTEGER NOT NULL
```

2.2 schedule.py

This module is in charge of orchestrating the schedule of the courses.

It will run in a loop until one of the following conditions hold:

1. The database file schedule.db does not exist.
2. All courses are done (The courses table is empty)

At each iteration of the loop it will print the tables, the name of the table followed by the tuples (each in a row) as can be seen in the example

At the beginning all classrooms are free and available.

At each iteration, many courses as possible need to be assigned to their classrooms (by the input file order). A classroom is available if "current_course_time_left" is zero. When a course is assigned to a classroom the amount of students allowed to take that course should be deducted from the total amount of available students since each student is allowed to participate only in a single course.

At each iteration do the following per classroom

1. If a classroom is free assign next course to it if exists and
    a. print: ([iteration No.]) [classroom location]: [course name] is scheduled to start
    b. Update the values of "current_course_id" and "current_course_time_left in the classrooms table.

2. If a classroom is occupied:
    a. Print ([iteration No.]) [classroom location]: occupied by [course name]
    b. Decrease by 1 the current_course_time_left in the classrooms table.

3. If a course just finished (current_course_time_left=0)
    a. print ([iteration No.]) [classroom location]: [course name] is done
    b. Remove the course from the DB
    c. Go to 1 to assign a new course to the classroom immediately if exists (same iteration number).

2.3 `create_db.py`

This module is the module that builds the database and inputs the initial data from the configuration file. When run, it will be given an argument of the path for the config file. For example, python3 create_db.py config

If it is the first time the module runs, i.e. the database file does not yet exist, then it should create the database and the tables as specified, parse the config file, and store the data given in the config file in the database appropriately.
When finished creating the data base print the tables - The name of the table followed by the tuples (each in a row) as can be seen in the example

If it is not the first time the module runs, i.e. the database file does exist, then the module should just exit.

3 Configuration Files

Each line in the configuration file represents either a course(C), students(S), or a classroom(R).
For example:
"C, 1, SPL 191, cs_ungrd, 80, 3, 2" represents a course id is 1, named "SPL 191" requires 80 computer science undergraduate students, located at classroom with id 3 and needs 2 iterations to

complete , and.
“S, cs_grad, 150” represents there are 150 computer science graduate students
“R, 1, 90/233” represents the classroom 90/ 233 whose id is 1.

Note that S,C & R at the beginning of each input row define record type.

See the “Example” section here for an example.

4 Very (!) Important Notes

1. We will test your modules together, and independently. Independently means that we will, for example, use our own database but use your modules, or put one module of our own and use your other module. Therefore, make sure your database has the structure we specified **exactly**, and that the behavior of each module is precisely as specified. Also, make sure that the database filename is schedule.db. Failing to follow these guidelines will cause tests to fail, and your grade will suffer accordingly.

2. To save you time, you may assume the validity of input. For example, a course will not requires a non-existent classroom, and it always has enough students.

3. Note that course length is counted by the number of iterations that are needed to complete it.

4. Use the function print(*tuple*) (When *tuple* is your-tupple-that-you-want-to-print) in order to print the tables DO NOT format the output yourself
   For example:
   ```
   def print_table(list_of_tuples):
       for item in list_of_tuples:
           print(item)
   ```

5. **Warning:** Note that the printouts should be done according to the needed iterations! You cannot just do all the prints immediately and exit. You also must use databases, you cannot just save the info in lists or infer beforehand. So please, do not try to cheat us, and do the modules exactly as specified, as we will test this intensively!

6. We emphasize again, **please make sure everything is as described, otherwise you will lose critical points from your grade!** We care about your success as much as you

do. Make sure the database filename is **schedule.db** and NOT Schedule.db and NOT schedule.DB or any other permutations. Make sure your table and column names are also exactly as described. If you are not sure about something, then feel free to ask in the forum!

5 Example

Here is an example. Note that you ***cannot*** assume the order the information appears in the config file. You can only assume validity of each line's syntax, and the validity of the config file (no double data, enough students to fill up all the courses, classrooms exist, no illegal courses, etc.)

Suppose you are given the following config file with the filename `short_config`:

```
S, hist_grad, 12
S, cs_undgrad, 550
R, 1, 90/233
R, 2, 90/234
C, 1, French Revolution, hist_grad, 10, 1, 4
C, 2, SPL 191(1), cs_undgrad, 80, 1, 2
C, 3, SPL 191(2), cs_undgrad, 80, 2, 2
```

Then the database will look as follows:

courses table:

id	course_name	student	number_of_students	classroom_id	course_length
1	French Revolution	hist_grad	10	1	4
2	SPL 191(1)	cs_undgrad	80	1	2
3	SPL 191(2)	cs_undgrad	80	2	2

students table:

grade	count
hist_grad	12
cs_undgrad	550

classrooms table:

id	location	current_course_id	current_course_time_left
1	90/233	0	0
2	90/234	0	0

We first run: `python3 create_db.py short_config`

Since this is the first run, and the `schedule.db` file does not exist yet, it will be created, and the tables will be created like described, and the initial data from the config file will be parsed and put into the tables appropriately as described above.

That will give us the following output:

```
courses
(1, 'French Revolution', 'hist_grad', 10, 1, 4)
(2, 'SPL 191(1)', 'cs_undgrad', 80, 1, 2)
(3, 'SPL 191(2)', 'cs_undgrad', 80, 2, 2)
classrooms
(1, '90/233', 0, 0)
(2, '90/234', 0, 0)
students
('hist_grad', 12)
('cs_undgrad', 550)
```

Then, we will run: python3 simulator.py that will give us the following output:

```
(0) 90/233: French Revolution is schedule to start
(0) 90/234: SPL 191(2) is schedule to start
courses
(1, 'French Revolution', 'hist_grad', 10, 1, 4)
(2, 'SPL 191(1)', 'cs_undgrad', 80, 1, 2)
(3, 'SPL 191(2)', 'cs_undgrad', 80, 2, 2)
classrooms
(1, '90/233', 1, 4)
(2, '90/234', 3, 2)
students
('hist_grad', 2)
('cs_undgrad', 470)
```

```
(1) 90/233: occupied by French Revolution
(1) 90/234: occupied by SPL 191(2)
courses
(1, 'French Revolution', 'hist_grad', 10, 1, 4)
(2, 'SPL 191(1)', 'cs_undgrad', 80, 1, 2)
(3, 'SPL 191(2)', 'cs_undgrad', 80, 2, 2)
classrooms
(1, '90/233', 1, 3)
(2, '90/234', 3, 1)
students
('hist_grad', 2)
('cs_undgrad', 470)
(2) 90/233: occupied by French Revolution
(2) 90/234: SPL 191(2) is done
courses
(1, 'French Revolution', 'hist_grad', 10, 1, 4)
(2, 'SPL 191(1)', 'cs_undgrad', 80, 1, 2)
classrooms
(1, '90/233', 1, 2)
(2, '90/234', 0, 0)
students
('hist_grad', 2)
('cs_undgrad', 470)
(3) 90/233: occupied by French Revolution
courses
(1, 'French Revolution', 'hist_grad', 10, 1, 4)
(2, 'SPL 191(1)', 'cs_undgrad', 80, 1, 2)
classrooms
(1, '90/233', 1, 1)
(2, '90/234', 0, 0)
students
('hist_grad', 2)
('cs_undgrad', 470)
(4) 90/233: French Revolution is done
```

```
(4) 90/233: SPL 191(1) is schedule to start
courses
(2, 'SPL 191(1)', 'cs_undgrad', 80, 1, 2)
classrooms
(1, '90/233', 2, 2)
(2, '90/234', 0, 0)
students
('hist_grad', 2)
('cs_undgrad', 390)
(5) 90/233: occupied by SPL 191(1)
courses
(2, 'SPL 191(1)', 'cs_undgrad', 80, 1, 2)
classrooms
(1, '90/233', 2, 1)
(2, '90/234', 0, 0)
students
('hist_grad', 2)
('cs_undgrad', 390)
(6) 90/233: SPL 191(1) is done
courses
classrooms
(1, '90/233', 0, 0)
(2, '90/234', 0, 0)
students
('hist_grad', 2)
('cs_undgrad', 390)
```

Now, if we try to run `schedule.py` again, it will just print the tables and exit immediately because all courses are done already.

6 Development Environment

- You should use Python 3.5 and `sqlite3` (they are available at the computers in the laboratories).

- You can use any text editor to program the assignment, but make sure your code works when running it from the terminal as specified.

7 Submission

- The submission is done in pairs only. You cannot submit in a group larger than two, or a group smaller than two.

- You must submit one file with all your code. The file should be named `id1_id2.tar.gz`. This file should include the following files only:

    - `create_db.py`

    - `schedule.py`

- Extension requests are to be sent to majeek@cs.bgu.ac.il. Your request email must include the following information:

    - Your and your partner's name

    - Your and your partner's ID

    - Explanation regarding the reason of the extension request

    - Official certification for your illness or army draft

- Make sure your code runs correctly and as described in the labs.