# SPL 181
# Assignment 4

Published on: **14.1.2018**
Due date: **4.2.2018 23:59**
Responsible TA: Ben Eyal

## 1 General Description

The folks over at SPL course have decided to start a pilot of using students as cheap slave labor. To explore this new field, you are required to implement a simulation of workers carrying out tasks to evaluate the performance of such venture.

In this assignment, you will implement such a simulator using Python and SQLite.

## 2 Method and Technical Description

We will build a `sqlite3` database that will hold our tasks, workers, and resource tables.
The database filename will be **`world.db`**.
We will have 2 Python modules: **`create_world.py`** and **`simulator.py`**.

Each task should have a unique ID.

### 2.1 The Database Structure

The database `world.db` will have three tables. You can find examples in the "Examples" section of the assignment text.

- `tasks` table will hold information needed to execute each task.
  The columns are:
  ```
  id INTEGER PRIMARY KEY
  task_name TEXT NOT NULL
  worker_id INTEGER REFERENCES workers(id)
  time_to_make INTEGER NOT NULL
  resource_name TEXT NOT NULL
  resource_amount INTEGER NOT NULL
  ```
- `workers` table will hold the names and status of each worker.
  The columns are:
  ```
  id INTEGER PRIMARY KEY
  name TEXT NOT NULL
  status TEXT NOT NULL
  ```
- `resources` table holds which resources are available and how much.
  The columns are:
  ```
  name TEXT PRIMARY KEY
  amount INTEGER NOT NULL
  ```

## 2.2 `simulator.py`

This module is in charge of orchestrating the execution of tasks by the different workers.

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

1. The database file `world.db` does not exist.
2. The `tasks` table is empty (meaning there are no more pending tasks)

In the first iteration, all workers should have a status of “idle”.

In each iteration, a worker either accepts a new task and becomes “busy”, or is already working on an existing task. A worker’s status becomes “idle” if there are no more tasks which require it. Before starting to work on a new task, the resources required by the task should be deducted from the overall amount of the resource.

When a worker becomes “busy”, print:

```
[worker name] says: work work
```

If a worker is already working on a task, print:

```
[worker name] is busy [task name]...
```

When a worker finishes working on a task, print:

```
[worker name] says: All Done!
```

## 2.3 `create_world.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_world.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.

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 worker, a task, or a resource. For example:

`metal,250` represents there are 250 of a resource called “metal”.

`worker,1,Alice` represents a worker named “Alice” whose id is 1.

`researching nukes,2,metal,100,5` represents a task named “researching nukes” that should be assigned to worker with id 2, require 100 metal, and 5 iterations to complete.

You have an example (long) config file supplied in the assignment page, with the result. See the "Example" section here for a short and full example.

Note that we will assume that `time_to_make > 0` (except if we have run the task enough times).

## 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 `world.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 task will not require a resource that doesn't exist or there isn't enough of, and will not require a non-existent worker.
3. Note that "doing a task" in our context means the required amount of resources was taken, and that the worker worked for the specified number of iterations.
4. **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 them 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!
5. 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 **`world.db`** and NOT `World.db` and NOT `world.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 a full (and short) 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, no illegal tasks, enough resources to complete the task, etc.)

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

```
wood,50
metal,250
worker,1,Alice
worker,2,Bob
assembling wagons,1,wood,10,2
researching nukes,2,metal,100,5
```

Then the database will look as follows:

Tasks table

| id | task_name | worker_id | time_to_make | resource_name | resource_amount |
|---|---|---|---|---|---|
| 1 | assembling wagons | 1 | 2 | wood | 10 |
| 2 | researching nukes | 2 | 5 | metal | 100 |

Workers table

| id | name | status |
|---|---|---|
| 1 | Alice | idle |
| 2 | Bob | idle |

Resources table

| name | amount |
|---|---|
| wood | 50 |
| metal | 250 |

We first run: `python3 create_world.py short_config`

Since this is the first run, and the `world.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.

Then, we will run: `python3 simulator.py`

That will give us the following output:

```
Alice says: work work
Bob says: work work
Alice is busy assembling wagons...
Bob is busy researching nukes...
Alice is busy assembling wagons...
Bob is busy researching nukes...
Alice says: All Done!
Bob is busy researching nukes...
Bob is busy researching nukes...
Bob is busy researching nukes...
Bob says: All Done!
```

Now, if we try to run `simulator.py` again, it will just exit immediately without printing anything, because all tasks have been 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_world.py`
    - `simulator.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

  Requests without a compelling reason will not be accepted.
- Make sure your code runs correctly and as described in the labs.