SPL171
Assignment 4

Responsible TA's: **Morad Muslimany, Itay Ariav**
Published on: **19.01.2017**
Due date: **26.01.2017 at 23:59**

1 – General Description

$Cron$ is an important Unix utility that allows tasks to automatically run in the system in a scheduled manner (every minute x of the day, every minute x and hour y in the day, or every first day of the month, every 3rd day of the week, etc..).
The file that has a list of all the tasks to be done is called a $Crontab$, and the $Cron\ Daemon$ is the name of the process that is always running in the system that is responsible for reading and executing the tasks present in the $Crontab$ file when it is time to do them.
You can read more about it here.

In this assignment, we will implement a (very) simplified version of $Cron$ that will help us in managing our new hotel – $SPHotel$, using ***Python*** and ***SQLite***.

In order for our program to be sane and checkable in a reasonable time, we will assume that each task will be done 'every x seconds' for a specified integer x of that task.
Furthermore, in order for our program to be finite and terminate in a reasonable time as well, we will assume that each task will be done 'z' times only, for a specified integer z of that task.

2 – Hotel Services

Our hotel will have the following services. These services are the 'tasks' to be done.

1. **Breakfast service**
   - This service serves breakfast to the resident of a given room.
   - Each breakfast service task has the following attributes:
     1.1 DoEvery – integer x – specifies that this task should be done once every 'x' seconds.
     1.2 Room – integer y – specifies that the person this breakfast will be served to is the resident of room number y
     1.3 NumTimes – integer z – specifies that breakfast to this resident should be served 'z' times.
2. **Wakeup-call service**
   - This service gives a wake-up call to the resident of a given room
   - Each wakeup-call service task has the following attributes:
     2.1 DoEvery – integer x – specifies that this task should be done once every 'x' seconds.
     2.2 Room – integer y – specifies that the person this wakeup-call will be served to is the resident of room number y

2.3 NumTimes – integer z – specifies that the wakeup-call to this resident should be served 'z' times.

3. **Room-cleaning service**
   - This service cleans all **empty** rooms (rooms with no resident)
   - There can be at most 1 (or no) cleaning task. (but can be done several times)
   - The cleaning task, if it exists, will have the following attributes:

   3.1 DoEvery – integer x – specifies that this task should be done once every 'x' seconds.

   3.2 NumTimes – integer y – specifies that this task should be done 'y' times.

   For consistency, we will assume that the 'Parameter' Room for the 'clean' task is always 0, and we will always ignore it.

3 – Method and technical description

We will build an $sqlite3$ database that will hold our tasks tables, as well as our rooms tables.
The database's filename will be $cronhoteldb.db$.
We will have 3 python modules: $miniCronRunner$, $hotelManagement$, $hotelWorker$.

Each task should have a unique $TaskId$.

3.1 – The database structure

The database $cronhoteldb.db$ will have four tables.

This database will actually simulate (to an extent) a $Crontab$ file, but suited to our application and simplifications.

You can find examples in the Examples section of the assignment text.

3.1.1 **TaskTimes** table: this table will hold information about $DoEvery$ and $NumTimes$ for each $TaskId$.
The columns are:

$TaskId\ integer\ PRIMARY\ KEY\ NOT\ NULL$
$DoEvery\ integer\ NOT\ NULL$
$NumTimes\ integer\ NOT\ NULL$

3.1.2 **Tasks** table: this table will hold information about $TaskName$ and $Parameter$ for each $TaskId$.
The columns are:

$TaskId\ integer\ NOT\ NULL\ REFERENCES\ TaskTimes(TaskId)$
$TaskName\ text\ NOT\ NULL$
$Parameter\ integer$

3.1.3 **Rooms** table: this table will hold all the room numbers that exist in our hotel.
The columns are:

$RoomNumber\ integer\ PRIMARY\ KEY\ NOT\ NULL$`

3.1.4 **Residents** table: this table will hold information about the names of the residents of each occupied room.

The columns are:

$RoomNumber\ integer\ NOT\ NULL\ REFERENCES\ Rooms(RoomNumber)$
$FirstName\ text\ NOT\ NULL$
$LastName\ text\ NOT\ NULL$

3.2 – $miniCronRunner.py$

This module is the one that runs the tasks in the suitable time. This is a very simplified version of the $Cron\ Daemon$.
The **only operations** it is allowed to do on the database is:

1. Read ($SELECT$) the $TaskTimes$ table.
2. Read ($SELECT$) the $Tasks$ table.
3. Update ($UPDATE$) the $NumTimes$ field (and this field only) of the $TaskTimes$ table

It is **not allowed** to do any other operations on the database.

It will run in a loop until one of the following conditions hold:
1. The database file $cronhoteldb.db$ does not exist.
2. There are no tasks in the $TaskTimes$ table where their $NumTimes$ is > 0. Note that this means that whenever it does a task, it should decrease its' according $NumTimes$ entry by 1. (and thus the $UPDATE$ operation on the $NumTimes$ field)

In the first iteration, the $miniCronRunner.py$ should perform each of the tasks listed in the $TaskTimes$ table for the first time by calling the $dohoteltask(taskname, parameter)$ method of $hotelWorker$ with the task's $TaskName$ and $Parameter$.

In each later iteration, it must query the $TaskTimes$ table and execute each task that still needs to be done, and the time to do it again has come.

3.3 – $hotelWorker.py$

This module will actually behave as the applications the $CronTab$ tasks call.
It **must** have a method called $dohoteltask(taskname, parameter)$, which the $miniCronRunner$ will call.

$dohoteltask(taskname, parameter)$ should return the time t_i that this task has actually been done (ie, the time the print to screen has been done) to the $miniCronRunner$. The next time this task should be done (if it should) is $t_{i+1} = t_i + x$ where x is the specified $DoEvery$ parameter of this task. Furthermore, the $dohoteltask$ method should:

1. If it is a wakeup call (ie, $taskname = 'wakeup'$), then the parameter is a room number, and it should print:

$$[\boldsymbol{firstname}]\ [\boldsymbol{lastname}]\ \boldsymbol{in\ room}\ [\boldsymbol{roomnumber}]\ \boldsymbol{received\ a\ wakeup\ call\ at}\ [\boldsymbol{time}]$$

2. If it is a breakfast service, (ie, $taskname = 'breakfast'$), then the parameter is again a room number, and it should print:

$$[\boldsymbol{firstname}]\ [\boldsymbol{lastname}]\ \boldsymbol{in\ room}\ [\boldsymbol{roomnumber}]\ \boldsymbol{has\ been\ served\ breakfast\ at}\ [\boldsymbol{time}]$$

3. If it is a clean service, (ie, $taskname = 'clean'$), then it should print:

$$\boldsymbol{Rooms}\ [\boldsymbol{roomNum_1}, \dots, \boldsymbol{roomNum_k}]\ \boldsymbol{were\ cleaned\ at}\ [\boldsymbol{time}]$$

where the rooms are sorted in ascending order. Note that the fact that $dohoteltask$ receives only the task name and a room number, it should have a connection to the database so it would know what is the name of the resident of that room in the case of a wakeup-call or a breakfast service, or figure out which rooms are empty for the cleaning task, so that it is able to do the correct printout.

3.4 – $hotelManagement.py$

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 of the config file. (for example: $python\ hotelManagement.py\ config1$)

If it is the first time this module runs (ie, the database file $cronhoteldb.db$ does not exit), 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 it runs (ie, the database file exists already), then it should just exit.

4 – Configuration files

Each line in the config file either represents a room, a room with a resident, or a task.
For example:

$room, 112, Harry, Potter$ – represents that there is a room whose number is 112 with a resident whose first name is $Harry$, and whose last name is $Potter$.

$room, 200$ – represents that there is an empty room whose number is 200

$breakfast, 5,201,3$ – represents that there is a task of serving breakfast every 5 seconds to the person that is the resident of room 201, and this task should be done 3 times.

$wakeup, 2,093,15$ – represents that there is a task of making a wakeup-call every 2 seconds to the person that is the resident of room 093, and this task should be done 15 times.

$clean, 12,1$ – represents that there is task of cleaning the empty rooms every 12 seconds, and this tasks should be done 1 time only.

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. You have a video capture of the behavior for both configs as well ☺

Note that we will assume that $NumTimes > 0$ (except if we have ran the task enough times), and $DoEvery > 1$ second.

5 – Very (!) Important Notes

1. We will test your three modules altogether, and also independently. Independently means that we will for example use our own database and use your modules. Another example is that we will also put our own $miniCronRunner$ along with your two other modules, or put our own $hotelManagement$ and use your other two modules, or put two modules of our own and test your third module etc.... 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 *cronhoteldb.db*. Failing to follow these guidelines will cause tests to fail and your grade will suffer accordingly.
2. Computations take time, although it is minimal and negligible for our assignment. For example, when you decide that a task needs to be done now, and since doing this task requires referring to the database, selecting, inferring, computing, etc, the task will not be done exactly every 'x' seconds where x is its' specified attribute, since there is still the (very small) time it takes the system to do that task.
   Therefore, *we will **neglect** error-time-deltas of **up to*** 1 *second.* Note that that means that you need to make your programming sane and rational, and not make too much un-needed computations.
3. Use the $time$ module's $.time()$ function to know what time it is.
4. To save you time, you can assume validity of input. For example, we will not add a room that already exists. We will not add a task of $breakfast/wakeup$ to a room that does not have a resident. We will not add a resident to a room that already has a resident, etc…
5. Note that 'doing a task' in our context means practically doing the printout the $hotelWorker$ does for that task.
6. You can assume that we will not cutoff the $miniCronRunner$ in the middle of its' run. Whenever we run it, we will let it finish.
7. **Warning**: Note that the printouts should actually be timed as specified in real-world seconds! 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!
8. We emphasize on this 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 $cronhoteldb.db$ and NOT $cronHoteldb.db$ and NOT $cronHotelDb.DB$ and NOT $mydb.db$ or anything else different from as precisely described. Also, make sure your tables names are exactly the same, and your table columns names are also exactly the same. For example, $TaskId$ and not $TASKID$ and not $Taskid$ or anything else other than described. If you are not sure about something then feel free to look at the examples and the video example in the assignment page, or ask in the forum!

6 – 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 as a whole (no double data, illegal residents, illegal tasks, etc…)

Suppose you are given the following config file with the filename $shortConfig$:

```
room,112,Harry,Potter
room,115,Darth,Vader
room,234
room,127
clean,10,2
breakfast,5,112,2
wakeup,10,115,2
breakfast,3,115,1
```

Then the database will look as follows:

TaskTimes table

TaskId	DoEvery	NumTimes
0	10	2
1	5	2
2	10	2
3	3	1

Tasks table

TaskId	TaskName	Parameter
0	clean	0
1	breakfast	112
2	wakeup	115
3	breakfast	115

Rooms table

RoomNumber
112
115
234
127

Residents table

RoomNumber	FirstName	LastName
112	Harry	Potter
115	Darth	Vader

We first run: $python\ hotelManagement.py\ shortConfig$

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

Then, we will run: $python\ miniCronRunner.py$

which will give us the following output:

```
> python hotelManagement.py shortConfig
> python miniCronRunner.py
Rooms 127, 234 were cleaned at 1483976871.69
Harry Potter in room 112 has been served breakfast at 1483976871.69
Darth Vader in room 115 received a wakeup call at 1483976871.69
Darth Vader in room 115 has been served breakfast at 1483976871.69
Harry Potter in room 112 has been served breakfast at 1483976876.69
Rooms 127, 234 were cleaned at 1483976881.69
Darth Vader in room 115 received a wakeup call at 1483976881.69
>
```

Note that the times are the output of printing $str(time.time())$. Your initial numbers will vary depending on the system time you ran $miniCronRunner$, but when the tasks are repeated they will have a time difference that we specified in the config. For example, the config says that $Harry\ Potter$ should be served breakfast every 5 seconds, and indeed, the first time he received breakfast was at 1483976871.69, and the second time was at 1483976876.69, and the difference is 5.

Now, if we try to run $miniCronRunner$ again, it will just exit immediately without printing anything, because all tasks have been done already.

7 – Development Environment

- You should use $Python\ 2.7$ and $sqlite3$ (they are available at the computers in the laboratories)
- You can use any text editor to program the assignment, but make sure that your code works when running it from the terminal as specified.

8 – 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 three files only:
  - $miniCronRunner.py$
  - $hotelManagement.py$
  - $hotelWorker.py$
- Extension requests are to be sent to majeek@cs.bgu.ac.il. Your request email must include the following information:
  - Your name and your partner's name.
  - Your id and your partner's id.
  - Explanation regarding the reason of the extension request.
  - Official certification for your illness or army drafting.

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

Have Fun! ☺