Luigi De Russis, Teodoro Montanaro

LAB 4 - USING DATABASES (SQLITE) WITH PYTHON

EXERCISE 1 – CREATE A SQLITE DATABASE

Perform the following actions.

1. Using SQLiteStudio, create a sqlite database (task_list.db).
2. Create the "task" table with the following columns:
    - id_task: it will contain an (auto generated) integer value that represents the unique identifier of each task;
    - todo: it will contain the text of each task;
    - urgent: it will contain an integer value that will assume 0 if the task is not urgent, 1 otherwise.
3. Insert (by hand) in the just created database all the tasks contained in the "task_list.txt" file. Choose the urgent value randomly. The file can be downloaded at the following link: http://goo.gl/GdVNBo .

Suggestions:

1. SQLiteStudio is portable, so it does not need any installation and can be downloaded at the following link: http://sqlitestudio.pl/?act=download

EXERCISE 2 – USE THE DATABASE INSTEAD OF THE TEXT FILE

Extend the program developed in the first exercise of the third laboratory (todo_list_tts_ex1.py) to increasingly substitute the text file with the database.

The following exercises will incrementally re-implement existing features so that the script will use the database instead of the file.

Exercise 2 - part 1 – Add a new task

Add a new task to the "task_list.db" database: disable all existing options except the 1st one and modify the right method to read from the database (instead of reading from the text file).

(OPTIONAL: ask the user to specify if the task is urgent or not. Otherwise, set it manually to 0)

Consequently, at startup the program will show only the following list of possible actions:

1. insert a new task (a string of text)
2. close the program

and every time the user selects option 1, the program will add the task to the database (instead of adding it to the file).

Suggestions:

1. When you prepare the sql query, you should use placeholders to specify parameters. Look at the following documentation to understand what is the right placeholder for sqlite databases: https://docs.python.org/2/library/sqlite3.html

Example:

Run the program: > todo_list_db_ex2a

First screen shown by the program:
Insert the number corresponding to the action you want to perform:

1. *insert a new task;*
2. *close the program.*

Your choice:

Exercise 2 - part 2 – Show all existing tasks

Show all existing tasks, sorted in alphabetic order reading from the database.

Consequently, at startup the program will show the following list of possible actions:

1. insert a new task (a string of text)
2. show all existing tasks, sorted in alphabetic order
3. close the program

and, every time the user selects option 2, the program will show the tasks getting them from the database.

Example:

Run the program: > todo_list_db_ex2b

First screen shown by the program:
Insert the number corresponding to the action you want to perform:

1. *insert a new task;*
2. *show all the existing tasks in alphabetic order;*
3. *close the program.*

Your choice:

Luigi De Russis, Teodoro Montanaro

Exercise 2 - part 3 – Remove existing tasks

Remove all the existing tasks that contain a provided string from the database.

Consequently, at startup the program will show the following list of possible actions:

1. insert a new task (a string of text)
2. show all existing tasks, sorted in alphabetic order
3. remove all the existing tasks that contain a provided string
4. close the program

and every time the user selects option 3, the program will remove the tasks from the database.

Example:

Run the program: > todo_list_db_ex2c

First screen shown by the program:
Insert the number corresponding to the action you want to perform:

1. *insert a new task;*
2. *show all the existing tasks in alphabetic order;*
3. *remove all the existing tasks that contain a provided string;*
4. *close the program.*

Your choice: