Exercise 1

Using code provided in Example 2, create `bikesharing` table in `default` database with the following columns:

- tstamp **timestamp**
- cnt **integer**
- temperature **decimal(5, 2)**
- temperature_feels **decimal(5, 2)**
- humidity **decimal(4, 1)**
- wind_speed **decimal(5,2)**
- weather_code **integer**
- is_holiday **boolean**
- is_weekend **boolean**
- season **integer**

Exercise 2

- load values from `london-bikes.csv` and insert it into the database, one by one
- * commit after every 100 inserts, not after every one

Fetching values

- To fetch values execute a select statement in cursor
- Then use cursor's `fetchone` to fetch a single row or `fetchall` to fetch all resulting rows
- Cursor also stores row count in a `rowcount` property and
- Column names in `description` property

Exercise 3

- fetch total sum of new shares by season
- fetch total sum of new shares during thunderstorms
- fetch the date and hour with the most new shares

Updating values - Exercise 4

- Add 10 to `cnt` column for all 2015-01-09 entries

Deleting values - Exercise 5

- Delete all entries from 2017-01-03 in `bikesharing` table

Exercise: “TODO application”

Exercise 6

The goal: a “To Do” application

- Create a `todo_app` database
- Create a `tasks` table with the following schema
  - id **int not null auto_increment**
  - task **text not null**
  - done **boolean**
  - **primary key** - id
- In a loop:
  - ask user what to do using `input()`
  - show task list
  - mark task as done
  - add new task
  - exit application
- Implement functions which perform the above actions using the database as task storage
- For `show tasks`:
  - print all open tasks and their ids in order of ids
- For `mark as done`
  - ask user which `id` to mark as done
  - update the done field in the table for given `id`
- For `add new task`
  - ask for task name/description
  - insert a new record to the tasks db