Connecting Python to a Database

We have seen in the lecture that it is possible to access an SQL database from within a Python program. In the case of an SQLite database, this is done using Python's '`sqlite3`' module. This exercise merely confirms that you can successfully create an SQLite database that is accessible from with a Python script.

1. Importing the database

This test assumes the existence of the Airline database from last week's workshop. You can re-create the Airline database by importing the 'dump' script file `airline.sql` via a graphical interface such as the *DB Browser for SQLite*. Use this or an equivalent tool to create database `airline.db` in the same folder that you'll be developing your Python scripts. Using the *DB Browser for SQLite* interface this is most easily done by choosing the 'Import database from SQL file' menu option. Alternatively, you can first create a 'new database' and then execute the `airline.sql` file as an SQLite script.

Having done this, use the graphical interface to browse the database's contents to ensure that it has been set up correctly. (If you have problems creating the database using the script provided we have also supplied a copy of the database itself, `airline.db`, but it would be better for you to use the script so that you can see the SQLite statements involved.)

2. Testing the Python-SQLite connection

Python scripts that access SQLite databases will, in general, do the following steps:

1. Import the necessary SQLite functions;
2. Create a "connection" to the database;
3. Get a "cursor" pointing into the database of interest;
4. Execute SQLite statements and queries and retrieve the results; and
5. Close the connection to the database to unlock it.

A simple Python program, `connect_to_database.py`, has been provided to show how to connect to an SQLite database and retrieve data from it. Examine this code and run it to confirm that you can successfully connect to the Airline database.