Harvard John A. Paulson School of Engineering and Applied Sciences

F. Wermelinger
Office: Pierce 211

Pair-Programming 11

Custom file reader, SQLite3 in Python

Issued:	November 11, 2022
Due:	November 25, 2022 11:59pm

In this pair-programming session you will implement a reader class for some file data you have to work with. You will then use the reader object to populate a SQLite3 database using Python code.

You should work on the exercises in groups of 3 to 4 students via a `tmate` session. Your team members can submit the same file. Please indicate your names in a header in the files. See the tutorials on the class website for an example pair-programming workflow.[1] Do not forget to commit and push your work when you are done. Ensure that you are on your *default branch* for this and not, possibly, on your homework branch.

Exercise 1: SQLite3 in Python

Deliverables:

1. `exercise_1.py`

You are a clerk in some office related to presidential elections and you know about Python. You are working with candidate data that is provided to you in the form of ASCII data files such as the one given in `data/candidates.txt`.

Implement a data reader class called "`CandidateReader`" with the following methods:

`__init__` Create an instance of a data reader object. The method takes a `path` argument which is a string that points to the data file to be processed. You are free to add optional arguments if you see fit. Since the files you are working with are small, you should read the file once in this method and keep the data in a convenient data structure. When you read string elements, make sure you strip away surrounding white space of the delimited data. *Do not assume you will always get clean data*. You will read these elements as strings and you do not need to convert them to other types, although you could if you prefer that.

`__len__` Should return the number of entries (rows) in the data file.

[1] https://harvard-iacs.github.io/2022-CS107/pages/tutorials.html#tutorial-pp

`__iter__` Returns an iterator to iterate over the data rows (presidential candidates). Each item returned by the iterator should be an iterable itself (e. g. a tuple) with the columns of the current row as its elements.

`get_header` Returns an iterable for the items contained in the file header. You can assume the first line in the files you work with will always be the file header describing the column identifiers.

Hint: *You can open files in Python using the* `open()` *built-in.*[2] *The returned file object supports the iterator protocol. For example, you can get the first line in the file by calling* `next()` *on the file object.*

Your goal is to use this data reader to populate a SQLite3 database for presidential candidates and possibly other data for supporters later on. For that purpose, write a helper function called `add_table` to add new tables to a `sqlite3` cursor object. The function signature could look like this

```
def add_table(fname, table_name, cursor):
    """
    Add a new table to sqlite3 database.

    Add the data in the file `fname` to a new table with `table_name` to the
    `cursor` sqlite3 object. Column names are determined from the header
    associated to the data file.  If a table with `table_name` exists in the
    `cursor` object, the table will be dropped before a new table is added.

    Parameters
    ----------
    fname : str
        Path to data file.
    table_name : str
        Name of the new table.  Drop the table if it exists.
    cursor : sqlite3.Cursor
        SQLite3 cursor object.

    Notes
    -----
    The function replaces all occurrences of white space in file header items
    with underscore characters.

    """
    pass
```

If `table_name` already exists in the database, drop it first before you add a new table. To improve robustness between your code and the SQLite interface you are communicating

[2] https://docs.python.org/3/library/functions.html#open

with, make sure that you replace possible white space in data header items with underscores. You should use your `CandidateReader` class in this helper function. See the SQLite slides of lecture 21 for further hints about what you will need from the `sqlite3` Python module.

Finally, you need application code that will create a SQLite database called `presidential.db` which you then populate with data you read from files using your helper function and the data reader class. Such application code should look similar to this:

```
if __name__ == "__main__":
    # 1. Create a connection to a SQLite3 database called `presidential.db`

    # 2. Obtain a cursor object form your database
    # See: https://www.python.org/dev/peps/pep-0249/#cursor-objects

    # 3. Add a new table called `candidates` associated to a data file you pass
    # as an argument.
    add_table("./data/candidates.txt", "candidates", cursor)

    # 4. Clean up the connection to the database (make sure you commit changes)
```

You can optionally inspect your `presidential.db` database with a browser tool such as https://sqlitebrowser.org/.

Please implement the code for this exercise in `lab/pp11/exercise_1.py`.

Please see `solution/exercise_1.py` for the solution code.