Murach's Python Programming

Case Study by Section: Baseball Team Manager

For this case study, you will use the skills that you learn in *Murach's Python Programming* to develop a program for managing a baseball team. This program stores the data for each player on the team, including the player's name, position, and batting average. This program also lets the manager specify a starting lineup for each game.

After you finish section 1 of *Murach's Python Programming*, you can develop a starting version of the program. After you finish section 2, you can improve this program. After you finish section 3, you can create an object-oriented version of this program. And after you finish section 4, you can store the data for this program in a database and use a GUI to work with that data.

General guidelines

Naming

- When naming the folders for your programs, please use the conventions specified by your instructor. Otherwise, store the files for a program in a folder named *first_last_baseball*_sec*X* where *first_last* specifies your first and last name and *X* specifies the section number, as in sec1.
- When creating names for variables and functions, please use the guidelines and recommendations specified by your instructor. Otherwise, use the guidelines and recommendations specified in *Murach's Python Programming*.

User interfaces

- You should think of the user interfaces that are shown for the case studies as starting points. If you can improve on them, especially to make them more user-friendly, by all means do so.

Specifications

- You should think of the specifications that are given for the case studies as starting points. If you have the time to enhance the programs by improving on the starting specifications, by all means do so.

Top-down development

- Always start by developing a working version of the program for a case study. That way, you'll have something to show for your efforts if you run out of time. Then, you can build out that starting version of the program until it satisfies all of the specifications.
- In particular, you should use top-down coding and testing as you develop your programs as described in chapter 5. You might also want to sketch out a hierarchy chart for each program as a guide to your top-down coding.

Section 1: Create the program

Create a program that lets the manager of a baseball team keep the data for each player and also specify and display the lineup for a baseball game.

Console

```
================================================================
                    Baseball Team Manager
MENU OPTIONS
1 – Display lineup
2 – Add player
3 – Remove player
4 – Move player
5 – Edit player position
6 – Edit player stats
7 - Exit program

POSITIONS
C, 1B, 2B, 3B, SS, LF, CF, RF, P
================================================================
Menu option: 2
Name: Mike
Position: c
At bats: 11
Hits: 4
Mike was added.

Menu option: 1
   Player       POS    AB     H      AVG
----------------------------------------------------------------
1  Denard       CF     545    174    0.319
2  Joe          2B     475    138    0.291
3  Buster       C      535    176    0.329
4  Hunter       RF     485    174    0.359
5  Brandon      SS     532    125    0.235
6  Eduardo      3B     477    122    0.256
7  Brandon      1B     533    127    0.238
8  Jarrett      LF     215    58     0.27
9  Madison      P      103    21     0.204
10 Mike         C      11     4      0.364

Menu option: 3
Number: 10
Mike was deleted.

Menu option: 4
Current lineup number: 8
Jarrett was selected.
New lineup number: 2
Jarrett was moved.

Menu option: 5
Lineup number: 1
You selected Denard POS=CF
Position: lf
Denard was updated.

Menu option: 7
Bye!
```

Section 1: Create the program (continued)

Specifications

- The formula for calculating batting average is:

  ```
  average = hits / at_bats
  ```

- The program should round batting average to a maximum of three decimal places.
- Use functions to organize the code to make it more reusable, easier to read, and easier to maintain.
- If the user enters an invalid menu option, display an error message and display the menu again, so the user can clearly see the valid menu options.
- Make sure the user can't enter data that doesn't make sense (such as a negative number of hits or the player having more hits than at bats).
- Use a list of lists to store each player in the lineup.
- Use a tuple to store all valid positions (C, 1B, 2B, etc).
- When entering/editing positions, the program should always require the user to enter a valid position.
- Use a CSV file named players.txt to store the lineup.
- Store the functions for writing and reading the file of players in a separate module named db.py.
- Handle the exception that occurs if the program can't find the data file.
- Handle the exceptions that occur if the user enters a string where an integer is expected.
- Handle the exception that occurs if the user enters zero for the number of at bats.

Section 2: Improve the program

Use the skills you learned in section 2 to improve this program. This should improve the appearance of the console and the readability of the code.

Console

```
================================================================
                    Baseball Team Manager

CURRENT DATE:    2016-12-19
GAME DATE:       2016-12-21
DAYS UNTIL GAME: 2

MENU OPTIONS
1 – Display lineup
2 – Add player
3 – Remove player
4 – Move player
5 – Edit player position
6 – Edit player stats
7 - Exit program

POSITIONS
C, 1B, 2B, 3B, SS, LF, CF, RF, P
================================================================
Menu option: 1
   Player                                POS    AB     H     AVG
----------------------------------------------------------------
1  Denard Span                            CF   545   174   0.319
2  Brandon Belt                           1B   533   127   0.238
3  Buster Posey                            C   535   176   0.329
4  Hunter Pence                           RF   485   174   0.359
5  Brandon Crawford                       SS   532   125   0.235
6  Eduardo Nunez                          3B   477   122   0.256
7  Joe Panik                              2B   475   138   0.291
8  Jarrett Parker                         LF   215    58   0.270
9  Madison Bumgarner                       P   103    21   0.204

Menu option: 7
Bye!
```

Specifications

- Use the multiplication operator to display separator lines that use 64 characters.
- Use spaces, not tabs, to align the columns of data for the players.
- Make sure the program always displays the batting average with 3 decimal places.
- Display the positions by processing the tuple of valid positions.
- When the program starts, use the YYYY-MM-DD format to display the current date and to get the date of the next game from the user.
- Only display the number of days until the game if the game is in the future. Don't display the number if the game date is in the past or the user doesn't enter a date.
- Use a dictionary to store the data for each player. To get this to work, you need to modify the functions that read and write the data to the file so they work correctly with a list of dictionaries. That's because the previous version of this program used the csv module to work with a list of lists, but the csv module doesn't work with a list of dictionaries.

Section 3: Create an object-oriented program

Convert the Baseball Team Manager program from procedural to object-oriented. This shouldn't change the functionality of the code much, but it should make the code more modular, reusable, and easier to maintain.

Console

```
================================================================
                    Baseball Team Manager

CURRENT DATE:    2016-12-19
GAME DATE:       2016-12-21
DAYS UNTIL GAME: 2

MENU OPTIONS
1 – Display lineup
2 – Add player
3 – Remove player
4 – Move player
5 – Edit player position
6 – Edit player stats
7 - Exit program

POSITIONS
C, 1B, 2B, 3B, SS, LF, CF, RF, P
================================================================
Menu option: 1
   Player                                POS    AB     H     AVG
----------------------------------------------------------------
1  Denard Span                            CF   545   174   0.319
2  Brandon Belt                           1B   533   127   0.238
3  Buster Posey                            C   535   176   0.329
4  Hunter Pence                           RF   485   174   0.359
5  Brandon Crawford                       SS   532   125   0.235
6  Eduardo Nunez                          3B   477   122   0.256
7  Joe Panik                              2B   475   138   0.291
8  Jarrett Parker                         LF   215    58   0.270
9  Madison Bumgarner                       P   103    21   0.204

Menu option: 2
First name: Mike
Last name: Murach
Position: c
At bats: 0
Hits: 0
Mike Murach was added.

Menu option: 7
Bye!
```

Section 3: Create an object-oriented program (continued)

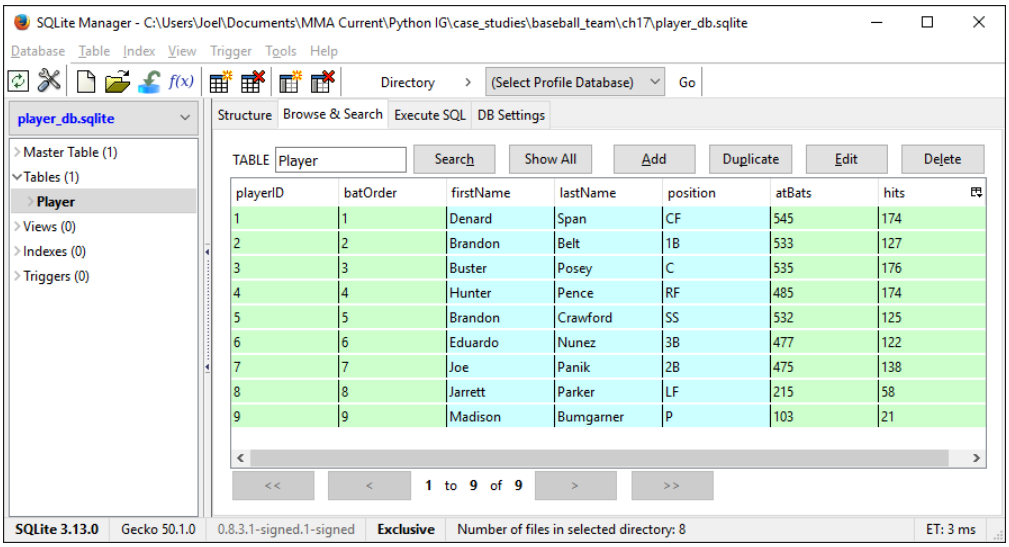
Specifications

- Use a Player class that provides attributes that store the first name, last name, position, at bats, and hits for a player. This class should also provide methods that return the full name and batting average for each player.
- Use a Lineup class to store the starting lineup for the team. This class should include methods that allow you to add, remove, move, and edit a player. In addition, it should include an iterator so you can easily loop through each player in the lineup.
- Use a file named ui to store the code for the user interface.
- Use a file named objects to store the code for the Player and Lineup classes.
- Use a file named db to store the functions that work with the file that stores the data.

Section 4: Use a database and a GUI

Use a database to store the data for the Baseball Team Manager program.

SQLite Manager with the Player table displayed



Specifications

- Use SQLite Manager to create a new database for the program.
- Use a single table named Player to store the data for the lineup of players. Here's a SQL statement that defines the columns and their data types for this table:

```
CREATE TABLE Player(
    playerID    INTEGER PRIMARY KEY     NOT NULL,
    batOrder    INTEGER                 NOT NULL,
    firstName   TEXT                    NOT NULL,
    lastName    TEXT                    NOT NULL,
    position    TEXT                    NOT NULL,
    atBats      INTEGER                 NULL,
    hits        INTEGER                 NULL
);
```

  To create the Player table, use SQLite's Execute SQL tab to run this CREATE TABLE statement.

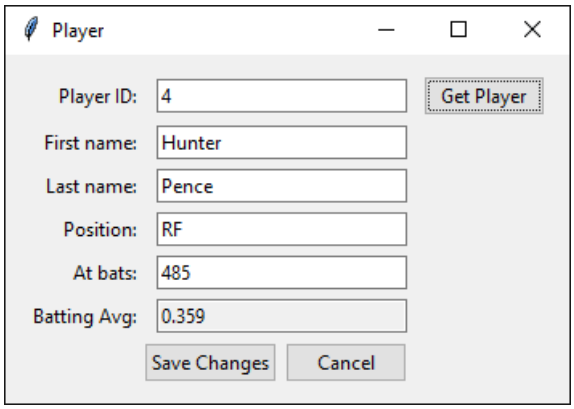
- Modify the db module so it provides all functions necessary to work with the player data. This should include functions for reading all players, adding a player, deleting a player, updating the batting order for all players, and updating the data for a player.
- To update multiple columns for a single row, you can use a SQL statement like this:

```
UPDATE Player
SET position = ?,
    atBats = ?,
    hits = ?
WHERE playerID = ?
```

Section 4: Use a database and a GUI (continued)

Use a GUI to allow the user to view and edit some of the data for a player.

GUI



Specifications

- The GUI should allow the user to view player data by entering a player ID and clicking the Get Player button. If the player ID exists, the GUI should display the player data as shown above. Then, the user can edit that data. If the player ID doesn't exist, the program should clear all text entry fields.
- The program should allow the user to edit player data by changing the data and clicking the Save Changes button. This should save the data to the database and clear all text entry fields.
- The program should allow the user to cancel any unsaved changes that have been made to the data by clicking on the Cancel button. This should restore the data in the text entry fields to the saved data.
- Assume the user will enter valid data for the first name, last name, position, at bats, and hits fields.
- This GUI should not allow the user to edit the player's position in the batting order.