

Murach's Python Programming

Case Study by Chapter: Baseball Team Manager

For this case study, you'll use the programming skills that you learn in *Murach's Python Programming* to develop a program that helps a person manage a baseball team. This program stores the data for each player on the team, and it also lets the manager set a starting lineup for each game.

After you read chapter 2, you can develop a simple program that calculates the batting average for a player. Then, after you complete most of the other chapters, you can enhance and improve this program. By chapter 16, you'll create an object-oriented version of this program. And after you complete chapters 17 and 18, you can add the use of a database and a GUI.

General guidelines	2
Chapter 2: Calculate a player's batting average	3
Chapter 3: Add a menu	4
Chapter 4: Use functions to organize the program	5
Chapter 5: Test and debug the program	6
Chapter 6: Use a list to store the players	7
Chapter 7: Use a file to save the data	8
Chapter 8: Handle exceptions	9
Chapter 10: Improve number and string formatting	10
Chapter 11: Get the date of the game	11
Chapter 12: Use a dictionary to store player data	12
Chapter 14: Use an object-oriented approach	13
Chapter 15: Add an object that has an iterator	14
Chapter 16: Implement the three-tier architecture	15
Chapter 17: Use a database	16
Chapter 18: Use a GUI	17

General guidelines

Naming

- When creating the folder and file names for your programs, please use the conventions specified by your instructor. Otherwise, for a program that consists of a single file, use this naming convention: *first_last_baseball_chXX.py* where *first_last* specifies your first and last name and *chXX* specifies the chapter number, as in *ch05*. For programs that have multiple files, store the files in a folder named *first_last_baseball_chXX*.
- 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. That way, you'll have something to show for your efforts if you run out of time. Then, you can build out that version of the program until it satisfies all of the specifications.
- From chapter 5 on, you should use top-down coding and testing as you develop your programs. You might also want to sketch out a hierarchy chart for each program as a guide to your top-down coding.

Chapter 2: Calculate a player's batting average

Create a program that displays a welcome message and calculates a player's batting average.

Console

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

This program calculates the batting average for a player based
on the player's official number of at bats and hits.
=====

Player's name: Pat
Official number of at bats: 11
Number of hits: 4

Pat's batting average is 0.364
```

Specifications

- The formula for calculating batting average is:
$$\text{average} = \text{hits} / \text{at_bats}$$
- The program should accept integer entries.
- Assume the user will enter valid data.
- The program should round the batting averages to a maximum of three decimal places.

Chapter 3: Add a menu

Update the program so it allows the user to select options from a menu.

Console

```
=====
                        Baseball Team Manager
=====
MENU OPTIONS
1 - Calculate batting average
2 - Exit program
=====
Menu option: 1
Calculate batting average...
Official number of at bats: 10
Number of hits: 3
Batting average: 0.3

Menu option: 1
Calculate batting average...
Official number of at bats: 11
Number of hits: 4
Batting average: 0.364

Menu option: 3
Not a valid option. Please try again.

Menu option: 2
Bye!
```

Specifications

- Assume the user will enter valid data.
- Display an error message if the user chooses an invalid menu option.

Chapter 4: Use functions to organize the program

Update the program so it uses functions to organize the code so it's easier to reuse and maintain.

Console

```
=====
                        Baseball Team Manager
=====
MENU OPTIONS
1 - Calculate batting average
2 - Exit program
=====
Menu option: 1
Calculate batting average...
Official number of at bats: 10
Number of hits: 3
Batting average: 0.3

Menu option: 1
Calculate batting average...
Official number of at bats: 11
Number of hits: 4
Batting average: 0.364

Menu option: 3
Not a valid option. Please try again.
MENU OPTIONS
1 - Calculate batting average
2 - Exit program

Menu option: 2
Bye!
```

Specifications

- Use a function to store the code that displays the menu.
- Use a function to store the code that calculates the batting average.
- Use a main function to store the rest of the code.
- Assume the user will enter valid data.
- 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.

Chapter 5: Test and debug the program

Specifications

- Create a list of valid entries and the correct results for each set of entries. Then, make sure that the results are correct when you test with these entries.
- Create a list of invalid entries. These should include entries that test the limits of the allowable values. Then, handle the invalid integers (such as negative integers and unreasonably large integers). In addition, make sure the user can't enter data that doesn't make sense (such as a player having more hits than at bats).
- Don't attempt to handle invalid entries that cause exceptions, such as the user entering a string like "x" for an integer value. You can do that after you read chapter 8.

Chapter 6: Use a list to store the players

Update the program so that it allows you to store the players for the starting lineup. This should include the player's name, position, at bats, and hits. In addition, the program should calculate the player's batting average from at bats and hits.

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: 0
Hits: 0
Mike was added.

Menu option: 1
  Player      POS    AB    H    AVG
-----
1  Joe        P      10    2    0.2
2  Tom        SS     11    4    0.364
3  Ben        3B      0    0    0.0
4  Mike       C       0    0    0.0

Menu option: 6
Lineup number: 4
You selected Mike AB=0 H=0
At bats: 4
Hits: 1
Mike was updated.

Menu option: 4
Current lineup number: 4
Mike was selected.
New lineup number: 1
Mike was moved.

Menu option: 7
Bye!
```

Specifications

- Use a list of lists to store each player in the lineup.
- Use a tuple to store all valid positions (C, 1B, 2B, etc).
- Make sure that the user's position entries are valid.

Chapter 7: Use a file to save the data

Update the program so it reads the player data from a file when the program starts and writes the player data to a file anytime the data is changed.

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: 1
  Player      POS    AB    H    AVG
-----
1  Denard     OF      545   174   0.319
2  Joe        2B      475   138   0.291
3  Buster     C       535   176   0.329
4  Hunter     OF      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    OF      215    58   0.27
9  Madison    SP      103    21   0.204

Menu option: 7
Bye!
```

Specifications

- Use a CSV file to store the lineup.
- Store the functions for writing and reading the file of players in a separate module than the rest of the program.

Chapter 8: Handle exceptions

Thoroughly test the program and update it so it handles all exceptions that you encounter during testing.

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

Team data file could not be found.
You can create a new one if you want.
=====
Menu option: 2
Name: Mike
Position: SS
At bats: 0
Hits: 0
Mike was added.

Menu option: X
Not a valid option. Please try again.

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

Menu option: 7
Bye!
```

Specifications

- 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.

Chapter 10: Improve number and string formatting

Update the program to improve the formatting of the numbers and the strings.

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: 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 make sure that horizontal separator lines use 64 characters.
- Use spaces, not tabs, to align columns. This should give the program more control over how the columns are aligned.
- Make sure the program always displays the batting average with 3 decimal places.
- Display the positions by processing the tuple of valid positions.

Chapter 11: Get the date of the game

Update the program so it begins by displaying the current date, getting the date of the game from the user, and displaying the number of days until the game.

Console 1

```
=====
                        Baseball Team Manager

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

MENU OPTIONS
1 - Display lineup
...
```

Console 2

```
=====
                        Baseball Team Manager

CURRENT DATE:      2016-12-19
GAME DATE:

MENU OPTIONS
1 - Display lineup
...
```

Specifications

- Use the YYYY-MM-DD format to display the current date and to get the current date from the user.
- Only display the number of days until the game if the game is in the future. If the game date is in the past, or if the user doesn't enter a game date, don't display the number of days until the game.

Chapter 12: Use a dictionary to store player data

Update the program so it uses a dictionary to store the data for each player (name, position, at_bats, hits). This shouldn't change the functionality of the program, but it should improve the readability of the code.

Console

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

CURRENT DATE:      2016-12-19
GAME DATE:

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 a dictionary to store the data for each player.
- The csv module only works with a list of lists, not a list of dictionaries. To work around this, you can modify the functions that read and write the data to the file so they work correctly with a list of dictionaries.

Chapter 14: Use an object-oriented approach

Convert the 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!
```

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 player's full name and batting average.

Chapter 15: Add an object that has an iterator

Update your program by adding a Lineup class that includes an iterator. This shouldn't change the functionality of the code, but it should make the code more modular, reusable, and easier to maintain.

Console

```
=====
                        Baseball Team Manager
=====
CURRENT DATE:      2016-12-19
GAME DATE:

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 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.

Chapter 16: Implement the three-tier architecture

If the program doesn't already implement the three-tier architecture, implement it now. This shouldn't change the functionality of the code, but it should make the code more modular, reusable, and easier to maintain.

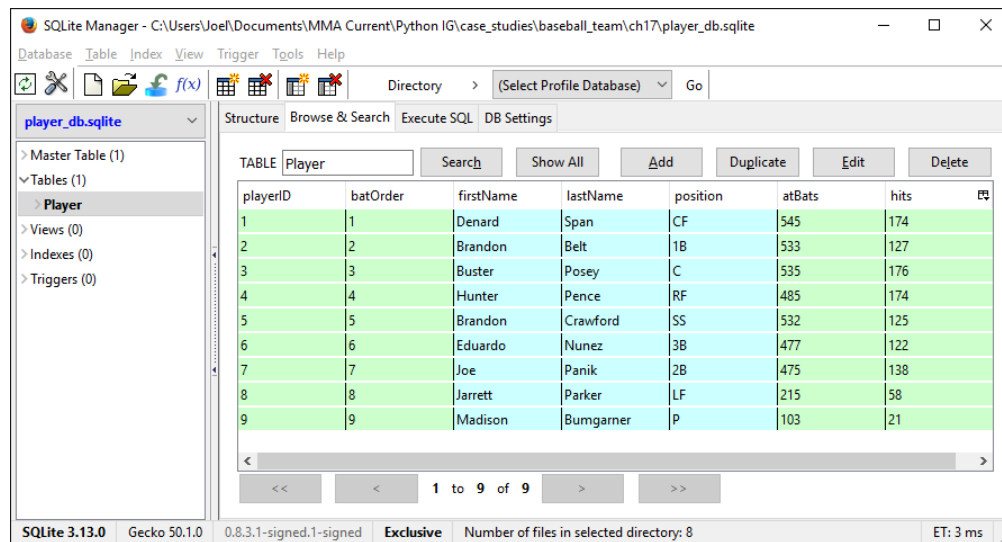
Specifications

- 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.

Chapter 17: Use a database

Use a database to store the data for the 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.

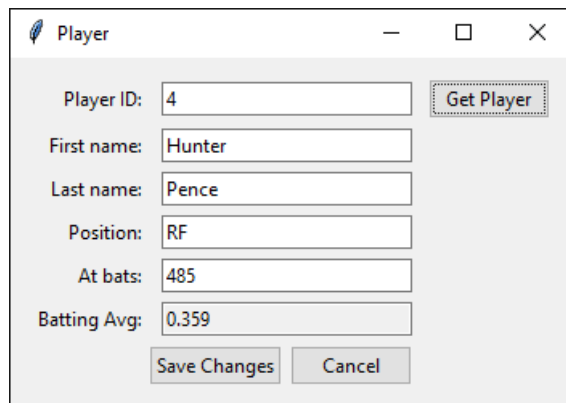
- Modify the db module so it provides all functions necessary to work with the player data. This should include functions for reading all players, adding a player, deleting a player, updating the batting order for all players, and updating the data for a player.
- To update multiple columns for a single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

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


Chapter 18: Use a GUI

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

GUI



Player

Player ID: 4 Get Player

First name: Hunter

Last name: Pence

Position: RF

At bats: 485

Batting Avg: 0.359

Save Changes Cancel

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.