|  |  |  |  |
| --- | --- | --- | --- |
| Rep # | Test Case | Expected Result | Actual Result |
| 1 | The formula for calculating batting average is: average = hits / at\_bats | All instances of batting average are calculated using the specified formula | The batting average for players is only calculated using the specified formula |
| 2 | The program should round batting average to a maximum of three decimal places. | Batting averages are rounded to two or three places when displayed to the user. | The batting average for players is rounded to two or three decimal places. |
| 3 | Use functions to organize the code to make it more reusable, easier to read, and easier to maintain. | All codes used in the program will be stored in functions and will be modular for future reuse | All the code accessed by the program is stored in easy-to-read functions |
| 4 | 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. | Entering anything except the integers 1 through 7 for the menu will result in displaying an error message and displaying the menu options to the user again. | Any invalid entry on the main menu results in displaying an error message, the menu is immediately displayed again, and another input is requested. |
| 5 | When entering/editing positions, the program should always require the user to enter a valid position. | Adding a position or editing a position causes an error unless the position is in the position’s tuple. | Only positions in the position’s tuple are considered valid input when adding a new player or editing a player’s position. |
| 6 | Store the functions for writing and reading the file of players in a separate module named db.py. | All functions used for writing or reading to the players bin file in a separate module in db.py | All functions used for writing or reading to the players bin file in a separate module in db.py |
| 7 | Handle the exception that occurs if the program can’t find the data file. | If the program cannot find the players bin file the exception is handled, the user is informed that no file was found, and an empty players bin file is created | When the program is unable to find the players bin file the exception is handled, the user is informed that no file was found, and an empty players bin file is created |
| 8 | Handle the exceptions that occur if the user enters a string where an integer is expected. | If the user enters a string where an integer is expected, the exception in handled, the user is informed that the input was invalid and to try again. | When the user enters a string where an integer is expected, the exception in handled, the user is informed that the input was invalid and to try again. |
| 9 | Handle the exception that occurs if the user enters zero for the number of at bats. In that case, the player’s batting average should be 0.0. | If the user enters zero for the number of at bats, the player’s batting average is set to 0.0 | When the user enters zero for the number of at bats, the player’s batting average is set to 0.0 |
| 10 | Use the repetition operator to display separator lines that use 64 characters. | Each player entry will span 64 characters by providing the same amount of space for each entry type | All entries in the table displayed to the user are formatted to span 64 characters. |
| 11 | Use spaces, not tabs, to align the columns of data for the players. | Table columns displayed to the user will be padded with spaces using f-string formatting | All table columns are aligned using f-string formatting. No tabs are used in player data formatting. |
| 12 | Make sure the program always displays the batting average with 3 decimal places. | The batting average will display with 3 decimal places for all batting averages. | The batting average is displayed with 3 decimal places for all batting averages. |
| 13 | Display the positions by processing the tuple of valid positions. (C, 1B, 2B, etc.). | Positions will be presented as a string of entries from the POSITIONS tuple to always incorporate all positions. | Positions displayed in the menu are generated based on the contents of the POSITIONS tuple. |
| 14 | 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. | The current date is displayed in YYYY-MM-DD format and the date of the next game is requested from the user | The current date is displayed in YYYY-MM-DD format and the date of the next game is requested from the user |
| 15 | Only display the number of days until the next game if the game is in the future. | If the user enters a date for the next game, the current date is subtracted from that date. If the result is greater than or equal than zero, then display the Date of next game and days until next game | When the user enters a date for the next game and it’s in the future, display the date of next game and days until next game. |
|  | Don’t display the number if the game date is in the past or the user doesn’t enter a date. | If the users enter x for the date or the current date is subtracted from the next game date is less than zero, than do not display the Date of next game and days until next game | When the user enters an x or the user enters a date for the next game and it in the past, do not display the date of next game and days until next game. |
| 16 | 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. | The program will load a dictionary from a bin file into a dictionary in memory. All subsequent data manipulation will be completed using this list of dictionaries. | The program loads a list of dictionaries into memory from the player bin file and all data manipulation uses the data in this list of dictionaries. The list of dictionaries is saved to the same bin file when the user selects exit from the menu. |

**Outline**

Main

Load players

Show dates

Display menu

Add player

Remove player

Move player

Edit player position

Edit player stats

Exit program

Save players

**Pseudocode**

Display program title and today’s date in YYYY-MM-DD format

Get date of next game in YYYY-MM-DD format **from user** OR x from user

**WHILE TRUE**

**IF** user enters x

**BREAK**

**ELSE IF** user enter date

Get date format YYYY-MM-DD from date of next game

Handle exception for value error if date is invalid format

**IF** date of next game minus current date is less than zero

Display program title

**BREAK**

**ELSE**

Display current date, date of next game, days until next game

**BREAK**

Display menu

Display positions in positions dictionary

**WHILE** menu option is not == 7

Get option from the user

**IF** menu option == 1 (show lineup function)

Display the first line of table displaying the attribute names

**WHILE** lineup is less than the length of the players list

**FOR** player in players

I=1

Get name, lineup position, position, hits, at bats

Average = hit / at bats

Convert average to decimal and quantize to display 3 decimal places

Handle exceptions for zero division error by setting average to 0.0

**IF** lineup position for player is equal to I

Display player lineup pos, name, position, at bats, hits, and average

I=I+1

**ELSE IF** menu option == 2 (add player function)

Show lineup

Get player name, position, at bats, and hits

Handle exceptions for value errors and empty inputs

Display user message stating the player’s name and that they were added.

**RETURN** players

**ELSE IF** menu option == 3 (remove player function)

Show lineup

Get integer of lineup position

Remove player from dictionary with the corresponding lineup position

**CONTINUE**

**ELSE** **IF** menu option == 4 (move player function)

Get integer of lineup position from user

Append player to index in player reference list + 1 and add to the player’s dictionary

Set lineup position to new player position

Set lineup position for each player based on new player position

**CONTINUE**

**ELSE IF** menu option == 5 (edit player position function)

Get lineup position, new player position from user

Handle exceptions for Value Error and Type Error

**IF** player position is in tuple of player positions

Set player position to new player position

**CONTINUE**

**ELSE** **IF** menu option == 6 (edit player stats function)

Get player lineup position, new at bats and new hits from user

Handle exceptions for Value Error and Type Error

**IF** hits > at bats

Display message to user that hits must be less than at bats

Get new input for at bats

**ELSE**

**FOR** player in players:

**IF** menu option == lineup position:

Set player hits and at bats to new hits and at bat stats

**ELSE IF** menu option == 7 (exit function)

Write data from players list to the players' file

Close program

**ELSE**

Display message that user option was an invalid integer and to try again

Handle Exceptions for value error and ask for valid entry

Display menu

Request a menu option from the user

**Write the Code**

I elected to write the Baseball Team Manager program







