Edward Myers
CS 521 Final Project – Asset Manager
8/19/21

For my final project I have created an asset management database. The code is split across two files. *Database.py* holds the Database class object, attributes and methods, while *emyers01_final_project.py* deploys the interfacing code which runs the primary menu loop and formats the output in a readable fashion.

Database.py defines several methods useful for manipulating a database. Load() and save() act as file input and output, allowing the user to read data into and save data out from the object. Four pieces of information are stored regarding individual objects: ID, name, category and quantity. These are loaded into a dictionary data type and appended to a list which acts as the primary collector. Next, two search functions search_by_name() and search_by_id() scan the database for their respective dictionary values. Add_item(), delete_item(), increment(), and decrement() are all set functions designed to add or remove whole dictionary items from the list or change the amount of a specific item's quantity value. Lastly, the three get functions get_db(), get_item_by_name() and get_item_by_id() return the dictionary list or one of its elements respectively for use as a variable in the calling program.

I believe this program achieves the goal of asset management in a robust way. At each level of method calling, several data quality checks kick out many possibilities for error, as evident by the long series of asserts in the class unit testing. The multiple text files accompanying the code are used for testing, the program's demo mode and the user interactive menu system.

The calling program runs an interactive menu loop for using the database:

```
1:   Load database from file.
2:   Save database to file. (only appears if the database has items)
3:   Display current database.
4:   Search database by name.
5:   Search database by ID.
6:   Add item to database.
7:   Remove item from database.
8:   Add quantity to existing item.
9:   Remove quantity from existing item.
10:  Demo Mode.
11:  Quit.
```

As just mentioned this program has a demo mode. As there are 11 options for database action off of the main program loop, I created a one-by-one demonstration of each function of the database with hard-coded inputs where possible to show full functionality with one option. Additionally, I've included a demonstration of how the program can merge databases together while adjusting the quantities of duplicate items. The last functionality added is analytical, demonstrating the use of sets to find the number of unique categories.

To run successfully this program should have all information from the zipped file present to run. The main interface program is *emyers01_final_project.py*, therefore it should be the file to execute. For your review, I have included a second report file titled *Final Program Description.docx* which shows the console printout of the option 10 demo mode execution for readability.

Objective checklist:

- One-page summary of what the program does and why it is useful.
    - Within this document.
- All source files: programs, classes and data.
    - Included in *emyers01_final_project.zip.*
- Instructions how to run your code and install any third-party modules
    - Within this document.
- 4 container types (list, tuple, set AND dictionary)
    - **List**: core collector of *Database.py*, initialized in __init__() and used throughout.
    - **Tuple**: used in *Database.py* within the increment() and decrement() methods to read and parse through the return tuple of search_by_id().
    - **Se**t: used in *emyers01_final_project.py* within the demo_mode() function. Removed duplication from the output listing unique item categories.
    - **Dictionary**: primary item attribute collector within *Database.py*. Dictionaries store each item's ID, name, category and quantity.
- 1 iteration type (for, while)
    - Used throughout *Database.py*, but most notably in the search_by_name() and search_by_id() methods.
- 1 conditional (if)
    - Everywhere in both program files.
- 1 try block with an else condition
    - Within *emyers01_final_project.py*, in the primary menu/input loop.
- 1 user-defined function
    - Numerous functions created within *emyers01_final_project.py*.
- 1 input and/or output file (include input data with your project)
    - For *Database.py* unit testing : Default_DB.txt, Default_OutDB.txt
    - For *emyers01_final_project.py* main program loop : Asset MGR.txt
    - *For emyers01_final_project.py* demo mode : Demo Save.txt, MergeDB1.txt, MergeDB2.txt
- 1 user-defined class. The class must be imported by your main program from a separate file and have the following required structures.

    Database(), imported from Database.py.

    - at least 1 private and 2 public self attributes

    __data_list, date_str, current_id.

    - at least 1 private and 1 public method that take arguments, return values and are used by your program

    Many public methods, __id_to_int().

    - an init() method that takes at least 1 argument — Present.
    - a repr() method — Present.
    - a magic method — __len__().
- Provide unit tests that prove that your class methods work as expected. The tests should evaluate results using assert statements.
    - Ample unit test assert loops with verified feedback present within the if __name__ == '__main__': code in *Database.py*.