**Structure:**

**Does the code completely and correctly implement the design -** The program is written to accomplish two primary functions, CRUD and Filtering of results. The CRUD operations are accomplished by using a Python script that can be access through a Jupyter Notebook interface. Filtering is handled though the Jupyter Notbook python file to create a dashboard with selection items through use of specialty capabilities base on breed of animal.

**Does the code conform to any pertinent coding standards –** The code conforms to coding standard based on organization and use of class object with operation being called via the various methods for CRUD and Filtering functions.

**Is the code well-structured, consistent in style and consistently formatted –** Yes, the class is declared, and the methods formatted in the same manor to be executed with error handling exceptions to identify potential issue with user interactions. Code comments are also provided to expound on developer ideas and plans.

**Are there any uncalled-for or unneeded procedures or any unreachable code –** There does not appear to be any unneeded or unreachable code.

**Are there any leftover stubs or test routines in the code –** There are no stubs associated with the code and currently no tests included.

**Can any code b replaced by calls to external reusable components or library functions –** Yes, the code is currently written as a Python script that can be access through a Jupyter Notebook interface.

**Are there any blocks of repeated code that could be condensed into a single procedure –** No, the code is written with simplicity in mind and focuses on the primary requirement for CRUD and filtering operations.

**Is storage use efficient –** Yes, the use of the MongoDB storage allows for all data for class representation to be stored in a flat-file document conserving on storage capacity.

**Are symbolics used rather than “magic number” constants or string constants –** No they are not, for execution of the filtering operations, hard coded strings are used to set the filtering parameters.

**Are any modules excessively complex and should be restructured or split into multiple routines –** The answer to this is no, simplicity was designed into the original code base, but the filtering method can be updated to remove the hard coded string values.

**Documentation:**

**Is the code clearly and adequately documented with an easy-to-maintain commenting style –** Yes, and as a process of improving, the comments will be updated to become more specific and supportive of the code being described.

**Are all comments consistent with the code –** Yes, they are consistently written throughout the provided code, however, some are likely not needed and will be evaluated for removal.

**Variables:**

**Are all variables properly defined with meaningful, consistent, and clear names –** Yes, there does not appear to be any confusion as to the variable used, especially since there are only a couple of them in the code base.

**Do all assigned variables have proper type consistency or casting –** Yes they are.

**Are there any redundant or unused variables –** No there are not.

**Arithmetic Operations:** There are no Arithmetic operations in the existing code.

**Loops and Branches:** There are no Loops or Branches in the existing code.

**Defensive Programming:**

**Are indexes, pointers, and subscripts tested against array, record, or file bounds –** There are not any of these structures used in the existing code.

**Are imported data and input arguments tested for validity and completeness –** There are no tests associated with the code base and the program does not allow for imported data.

**Are all output variable assigned –** There are no output variable associated with the code base.

**Are the correct data operated on each statement –** Yes, while executing the CRUD operations, the proper files and user input data is evaluated and verified for use and access.

**Is every memory allocation deallocated –** This is not a concern for the existing code due to the format of the data.

**Are timeouts or error traps used for external device access –** There are no external devices associate with the functionality.

**Are files checked for existence before attempting to access them –** Yes, error the code performs lookup of the requested data and performs validation prior to executing the chosen operation. Error handling messages are provided when needed.

**Are all files and devices left in the correct state upon termination –** Yes, selected operations are completed before closing out or logging out of the program.

**Practical Enhancements:**

Developer comments were included to help identify direction of the written code, but in some cases these could use a little more details and in others, they are just more lines that provide no clarification from the code that was written.

A note for improvement would be to add unit testing to provide code coverage to validate the execution of each operation is correct. To do this, the plan is to convert the current Python into JavaScript and include the unit test for each case use as well as provide more robust code comments.

There is a fundamental problem associated with the filtering functionality, in that the prescribed values are hardcoded and become very cumbersome to update if a change is needed. To help resolve this issue, in the conversion to JavaScript a flag will be created to call to make the operation symbolic vice a “magic numbers” type query

The data structure implemented is a flat-file structure which is good for storage and retrieval of the single file data stream, but limits query responses. As part of an effort to provide a more normalized set of data as part of the update to an SQLite database, we will update the data structure to include integers, strings, and floats.

There are currently no algorithms included in the program, but as development goes forward, we will investigate their uses to see if they could be implemented to enhance the filtering functionality.