# CS 340 README Template

## About the Project/Project Title

This project has been implemented in order for the end user to interact with the database by creating entries, querying entries, updating entries and deleting entries.

## Motivation

Allowing users to create, view, update and delete data is important because it creates a place where the user can input their own data which will be stored safely into the database. With this program, the user will also be able to view their already inputted data to verify their records. Users will also be able to view the company logo, interactive pie chart and future button interactivity.

## Getting Started

In order to get started with this program, the first step is to login as an admin user within mongo using terminal. After logged in as an admin user, you will then create a new user ensuring you provide them with the correct database you wish to interact with in the program.

## Installation

The tools needed to get your program running will be a file program such as Jupyter notebook, and a mongo account which can be accessed through the terminal. One main benefit of using Jupyter is being able to manipulate multiple files in the same window by means of different tabs. The file formats that need to be created are a python and an .ipynb file. These files work together so that the mongo db of your choice can be interacted with.

## Usage

This program is mainly run off of the python file created and ran in Jupyter notebook. In this file, it is important to first install the Mongo client and bson objects. Next, we will create a class object. Commands will then be used within the class object and here is where you will specify how you want to interact with the mongo database. The first command is essentially logging into the mongo client as the new user created from the mongo admin account. You will provide the local host number which can be accessed in the mongo terminal, and the username/password for that new user. The next commands will be methods which will allow the user to interact with the database as needed, such as creating entries, reading entries, updating entries and deleting entries. In this program we are able to create an entry only if data has been entered. If no data is entered, a message will be displayed. Similar to creating entries, another method was created to query data and display the results. If there is no data queried, a message is displayed to the user.

The next method is the update query which first searches a query, then if that query is found, it is updated based on the command line. If no results are found, a message is displayed. The delete method will delete all rows. Once these methods are created, we created tests to ensure the create, read, update and delete functionality is working as expected. It is important to note that we first made sure to import the name of the python file we are working off of. If a file was successfully created, it will display “Results of create: x”, where x is the new entry. Similarly, when a term is queried and exists in the database, the system will display “Results of read: x”, where x is the entry within the database. In addition, this program has also include the logo for the Grazioso Salvare company. A drop down has also been included to filter certain datapoints. Once an option is selected, the pie chart will display all records pertaining to that filter. A data table is displayed to show all results from the database. Each column has a filtering option as well as sorting and an empty text box to search for results within the datatable. A geographical map is displayed to show the first result in the data table. Future opportunities for this program include interactivity between filtering options and the data table as well as the geographical map.

### Code Example

Create Method

def create(self, data):

if data is not None:

self.database.animals.insert(data)

return True

else:

raise Exception(“Nothing to save, because data parameter is empty”)

return false

Read Method

def read(self, data):

if data is not None:

self.database.animals.find()

else:

raise Exception(“Nothing to search, because data parameter is empty”)

return false

Update Method

Def update(self,searchinfo, updatedinfo):

If searchinfo is not None:

return self.database.animals.update(searchinfo,updatedinfo)

else:

raise Exception(“Nothing to update, because data parameter is empty”)

return False

Delete Method

Def delete(self,deletedinfo):

If deletedinfo is not None:

return self.database.animals.remove(deletedinfo)

else:

raise Exception(“Nothing to delete, because data parameter is empty”)

return False

### Tests

*Test Create Method*

Print(“Results of create:”)

Print(x.create({‘breed’:”test”}))

*Test Read Method*

Print(“Results of read:”)

Print(x.read({‘breed’:”test”}))

*Test Update Method*

Print(“Results of update:”)

Print(x.update({}))

*Test Delete Method*

Print(“Results of delete:”)

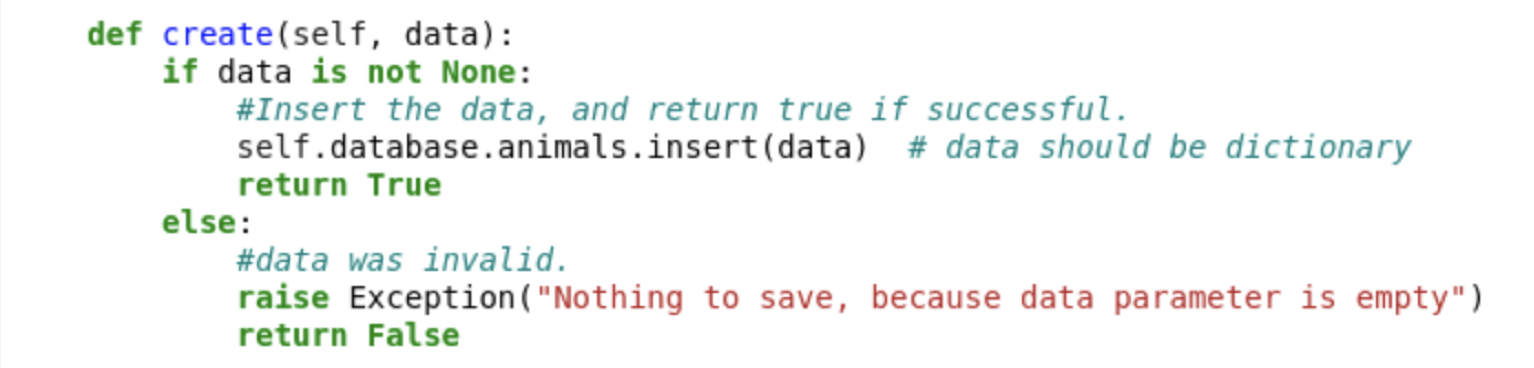
Print(x.delete({}))

### Screenshots

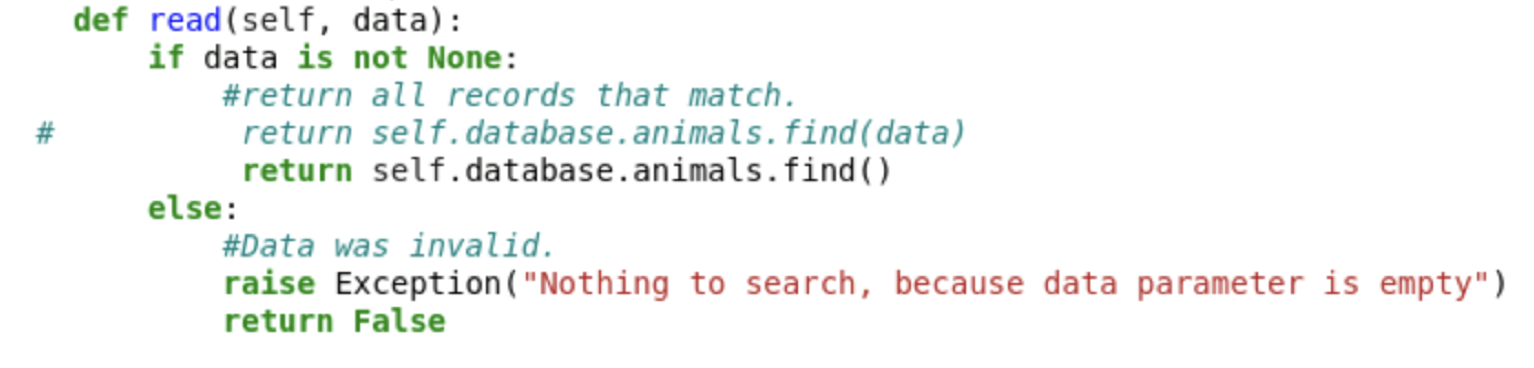
*Import/Login*

**

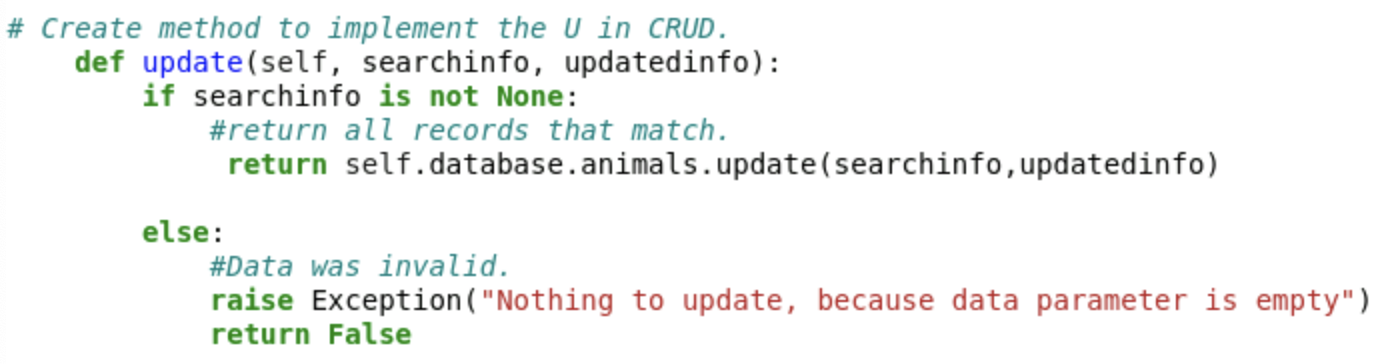
*Create Method*

**

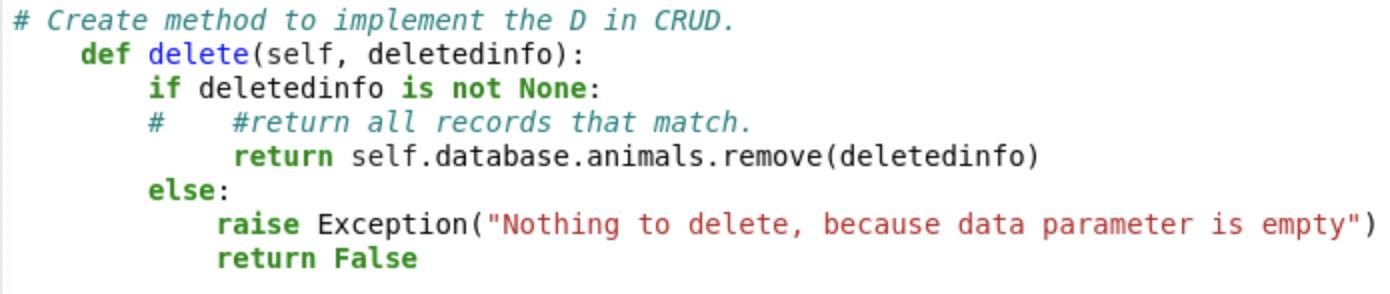
*Read Method*



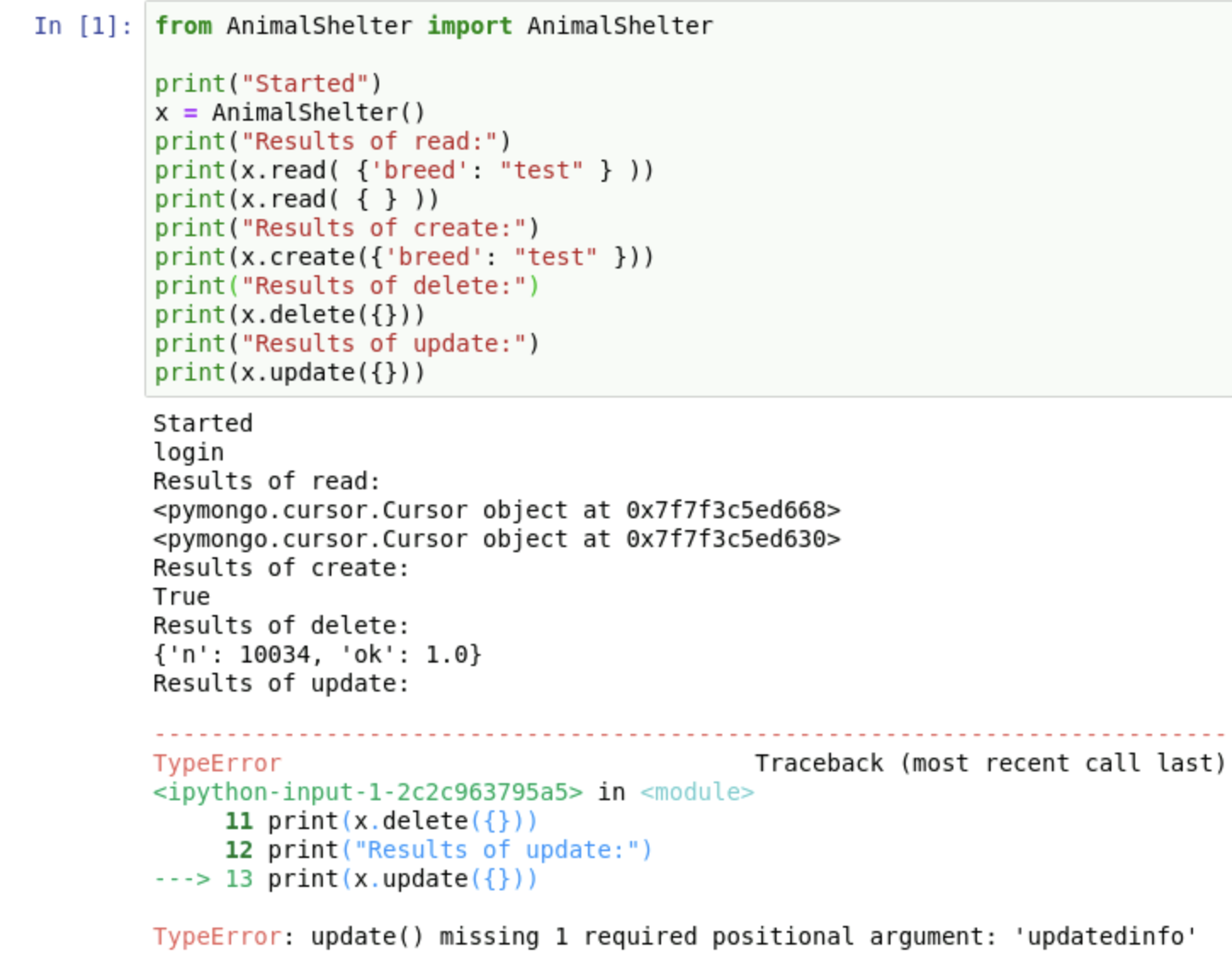
*Update Method*

**

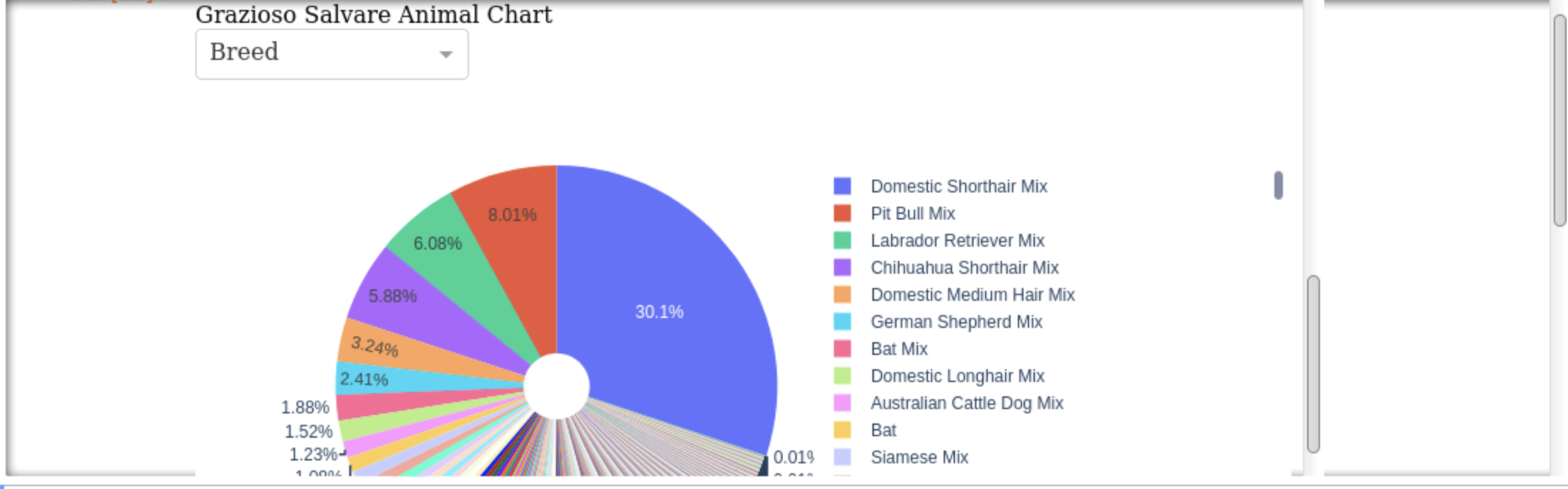
*Delete Method*

**

*Tests*

**

## Pie Chart



## Data tabe with filtering/sorting options and buttons



## *Geographical map*

## 

## Roadmap/Features (Optional)

* *Update Method test is missing 1 required positional argument: ‘updatedinfo’*
* *Update button interactivity so that specific query results are visible once button is clicked*
* *In addition to above item, geographical map will be updated to include results from interactive filter actions such as the button.*

## Contact

Your name: David Roehm