Eccentric billionaire Joffrey Hosencratz just purchased the web development company you work for. You've met him once in an elevator and he was impressed with your skill in developing web applications with the Django framework. He also relayed that his most recent trip to Sedona, AZ has left him in a bit of trouble. See, he fancies the show Rick and Morty and a particular scene coupled with a traumatic childhood squirrel experience and a bad crystal bath experience in Sedona has left him wanting.

He would like to start keeping track of all the known squirrels and plans to start with Central Park. He's asked you to build an application that can import the 2018 Central Park Squirrel Census data and allow his team to add, update, and view squirrel data.

The project should be Django based should have the following features:

- Management commands:
  Import: A command that can be used to import the data from the 2018 census file (in CSV format). The file path should be specified at the command line after the name of the management command.
    - `$ python manage.py import_squirrel_data /path/to/file.csv`

  The squirrel census file can be downloaded here:
  https://data.cityofnewyork.us/api/views/vfnx-vebw/rows.csv

  Export: A command that can be used to export the data in CSV format. The file path should be specified at the command line after the name of the management command.
    - `$ python manage.py export_squirrel_data /path/to/file.csv`
- Views:
  - A view that shows a map that displays the location of the squirrel sightings on an OpenStreets map.
    - Located at: `/map`
    - Methods Supported: GET
    - Will use the https://leafletjs.com/ library for plotting
      - Please note: Your browser will start to freeze if you plot more than 100 points at once. We will assume our client is okay with this plotting issue. Any 100 unique coordinates will do. Any 100 squirrel sightings will do. A simple slice of the database records is sufficient.
    - You can find the HTML template here: https://gist.github.com/logston/0b6f2cbb928a386decd63fd616d084dd
  - A view that lists all squirrel sightings with links to view each sighting
    - Located at: `/sightings`
    - Methods Supported: GET
    - Fields to show:
      - Unique Squirrel ID
      - Date

        - Link to unique squirrel sighting
    - This view should also have a single link to the “add” sighting view
- A view to update a particular sighting
    - Located at: `/sightings/<unique-squirrel-id>`
        - Hint: https://docs.djangoproject.com/en/3.1/topics/http/urls/#using-regular-expressions
    - Methods Supported: GET & POST
    - Fields to show:
        - **Latitude**
        - **Longitude**
        - **Unique Squirrel ID**
        - **Shift**
        - **Date**
        - **Age**
- A view to create a new sighting
    - Located at: `/sightings/add`
    - Methods Supported: GET & POST
    - Fields supported:
        - **Latitude**
        - **Longitude**
        - **Unique Squirrel ID**
        - **Shift**
        - **Date**
        - **Age**
        - *Primary Fur Color*
        - *Location*
        - *Specific Location*
        - *Running*
        - *Chasing*
        - *Climbing*
        - *Eating*
        - *Foraging*
        - *Other Activities*
        - *Kuks*
        - *Quaas*
        - *Moans*
        - *Tail flags*
        - *Tail twitches*
        - *Approaches*
        - *Indifferent*
        - *Runs from*
- A view with general stats about the sightings

- Particular stats are for you to decide but must include five of the attributes listed in the initial CSV download.
- Located at: `/sightings/stats`
- Method: GET

Data validation should happen at all points of data ingestion. Any field that does not have data for each record can be considered optional.

WHY

The purpose of this project is three fold:

1. Allow you to cultivate your programming skills in a real world project.
2. Practice working on a software project in a group setting (2 people to a group).
3. Practice use of the version control system Git.

WHO

Your teams will be made up of 2 people of your choosing. You will have a week to find group members. If you prefer a surprise, you can wait until the end of the week and you will be assigned to a group by me or a CA.

Groups will be managed via Canvas. Assign yourself to a group by navigating to the "People" section of this course in Canvas and joining a **Project Group**. Do not join or create a student group for this project. *Projects submitted within a "Student Group" as labeled on canvas will not be graded.*

WHAT

What exactly will you turn in? You will need to submit two links:

- A link (https) to the repository for your application code
    - For example: *https://github.com/YourUsername/YourProject.git*
    - A **public git clone** operation with the URL provided will take place to pull down your code. Thus, you must make sure the URL starts with "https" and ends with ".git". DO NOT SUBMIT THE git@github.com/*YourUsername/YourProject.git* ADDRESS!!

`PLEASE NOTE: YOU DO NOT NEED TO DEPLOY THIS PROJECT FOR THE FALL A 2020 iteration of this course.`
Both will be submitted on courseworks. In order to test your code, you will need to leave your server running. Please pick the server with the most credit left to deploy your code on top of.

Only one person per group needs to submit the application links on courseworks.

Grading

Everyone in the group will share the same grade.

- 10% - Clarity of code (ie. PEP8 + PEP257 style)
- 70% - Functioning as described
    - Ie. Does your project work
- 20% - Thorough use of Git
    - Your team has used Git to save incremental progress in your work. Each member has roughly the same number of commits, additions, and subtractions.

WHEN

The project is due on the last day of classes for the semester.

HOW

Your project will need to include the following:

- Source code
    - The code that makes your project do what it does
- README file
    - A file describing the tool, its use, and its behavior
- requirements.txt file
- .gitignore file
    - A file to ignore files that should not be committed (eg. pycache files)

You must use Python 3.6+ for your project and your project must be under Git version control. You must specify all dependencies (supplemental packages required for you project to operate correctly) in the requirements.txt file in the correct format. Otherwise, your application will not install and run correctly on the graders machines. Please not that keeping dependencies to a minimum is **strongly** encouraged both in and outside of this class.

This project requires a database. It must use a sqlite3 database located at the root of the repo. Setting up any other databases or importing data is too time consuming for our graders.

You will be using OpenStreetMaps for map background data. Please be respectful (and lawful) and show attribution in your maps. Please do not attempt to write your own JS mapping code for this project. Please use Leaflet instead.

You will most likely want to add styling to your site. Please use Bootstrap4 for this.

For deployments we will use Google App Engine.

What's a README?

A README file is a that sits at the root of the project, repo, or directory of an application and describes what the application does and how to use it. Here are some good examples of READMEs.

- https://github.com/python/cpython/blob/master/README.rst
- https://github.com/pandas-dev/pandas/blob/master/README.md
- https://github.com/tensorflow/tensorflow/blob/master/README.md

Your README needs to include:

- A description of what has been implemented
    - This only needs to be a paragraph or less of a description. More is welcome but not needed.
- Your group name and section
    - Eg. Project Group 51, Section 2
- A list containing the UNI for each member on the team
    - Of the form: "UNIs: [uni1, uni2, uni3]". Eg. "UNIs: [ab1234, cd7847, ef9873]"
- A link to the server running your application
    - For example: https://<your project id>.appspot.com/