Asteroid Hunters

Dongyu Zhu, Chris Peña, Haseeb Farid

**Goal:**

To collect data on Near-Earth Objects (NEOs), calculate the volume, and compare them to well-known, everyday objects (e.g. NEO x is similar in size to a fire engine.)

**Goals Achieved:**

NEO data was collected, average velocities and volumes were calculated.

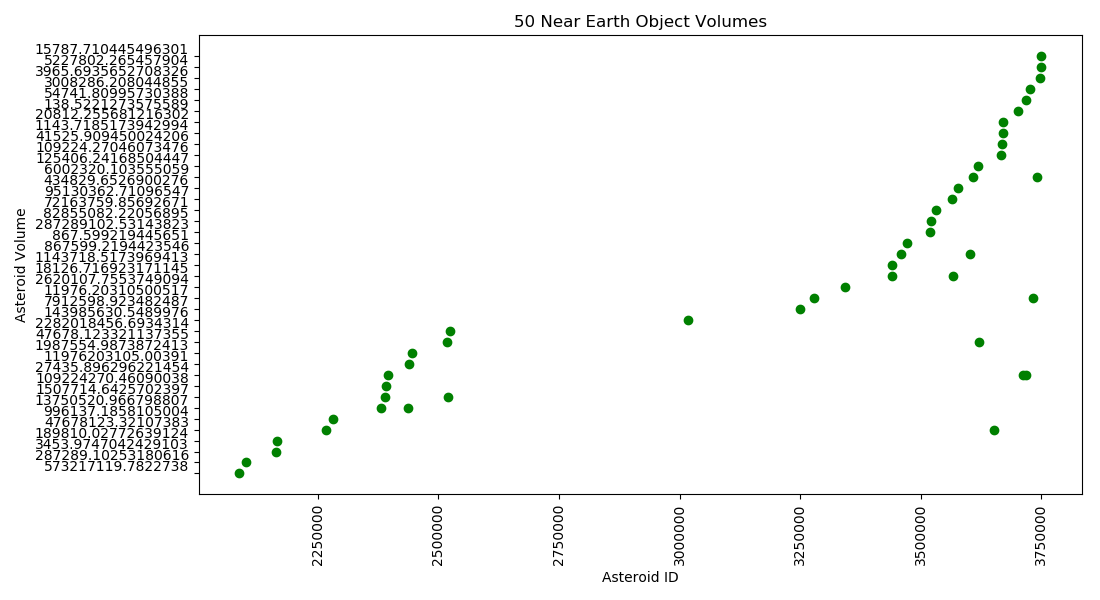
**Problems:**

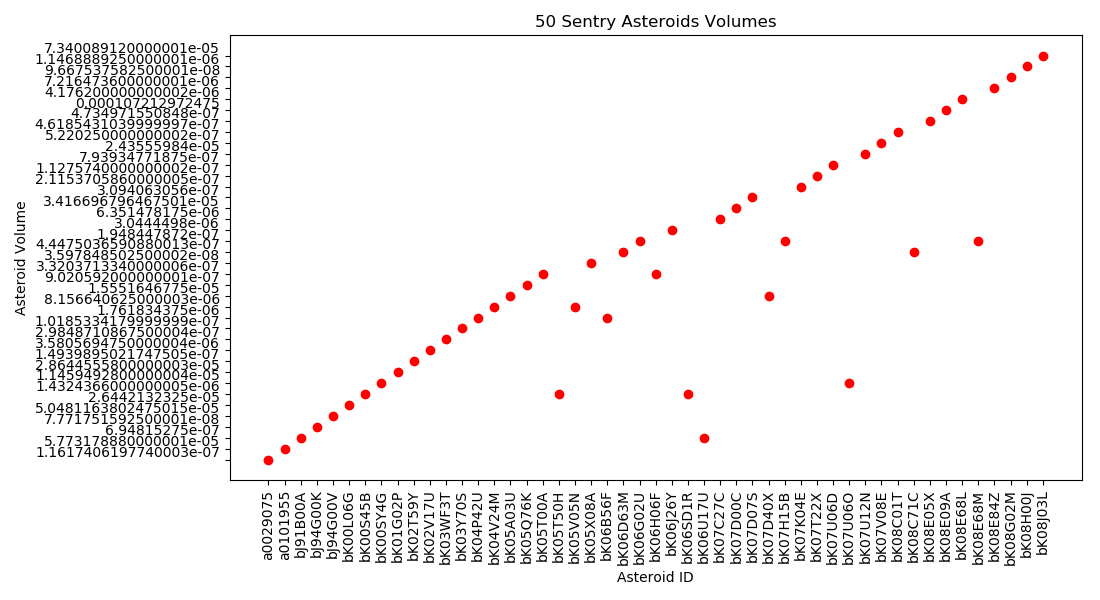
The API for comparing a given size to a well-known object required an authentication key. A request was made, but we decided to abandon the idea after a week of no response.

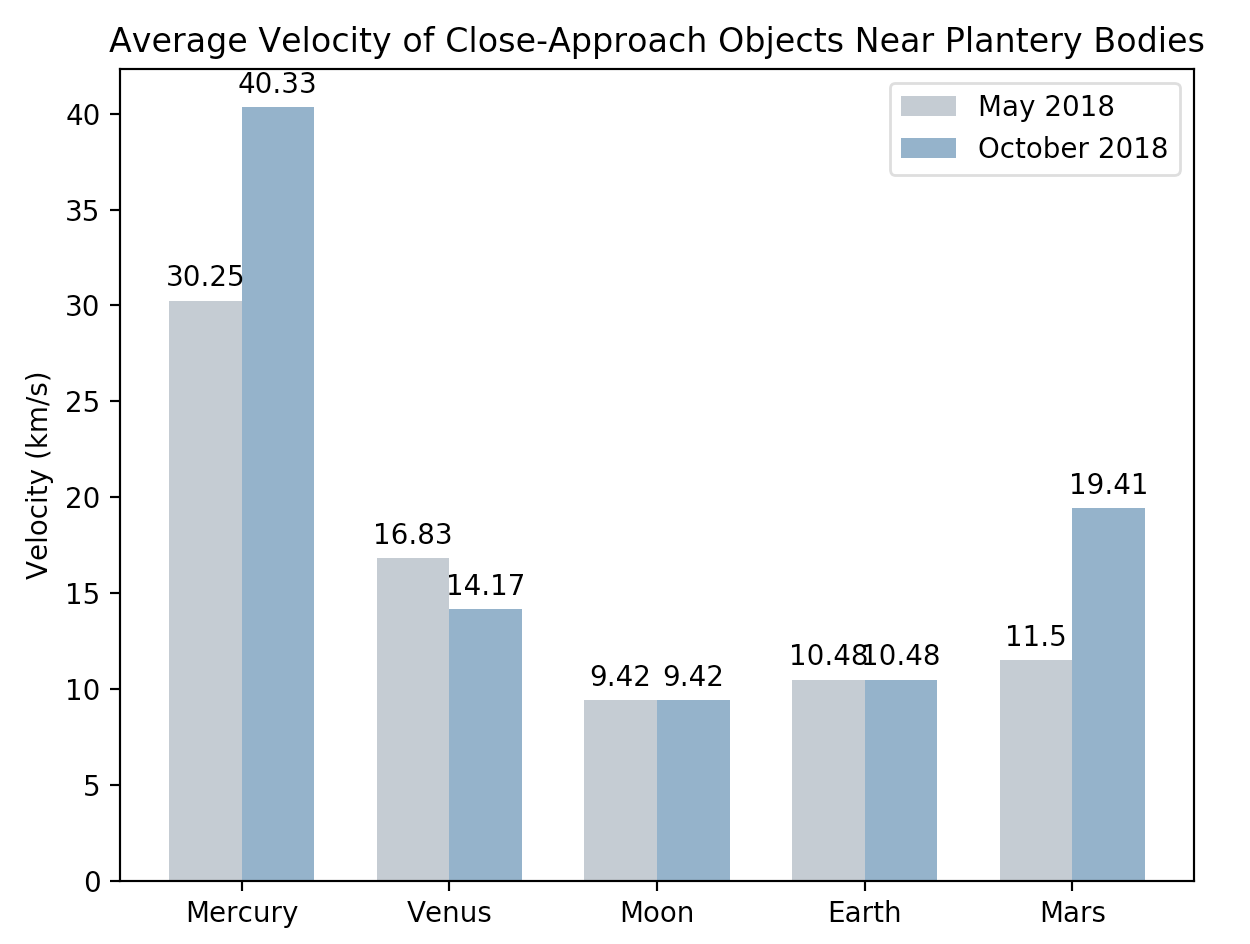
**Calculations files:**

volume\_calc\_neo.txt, volume\_cal\_sentry.txt, CAD\_Calc.txt.

**Visualizations:**

****

****

****

**Code instructions:**

**Documentation:**

NASASpider.py

**Overview:**

NASASpider.py is a module used to get asteroid data from 3 different NASA APIs, NeoWs, Sentry, and CAD. There are 6 functions in the module, 3 for spidering data, and 3 for storing them in a database.

**Functions:**

**get\_neo\_data(day\_date, key)**: get NEO data from NeoWs API. This function takes exactly 2 arguments and returns a dictionary of asteroids on a given day.

Day\_date parameter should take a datetime.date object. This parameter decides that the function will look for the NEO data for the specific date. Key parameter is the API key that is required to access and use web services available on the Data.gov developer network. User can apply it freely from <https://api.nasa.gov/>

The items in the returning dictionary have the format of “aid:{estimated\_min,estimated\_max}”. aid is the unique id of the asteroid, estimated\_min is the estimated minimum diameter of the asteroid, and estimated\_max is the estimated maximum diameter of the asteroid.

**get\_sentry\_data(impact\_p)**: get data of asteroids which have potential future Earth impact events from Sentry. The function takes only 1 argument and returns the data of asteroids satisfying the limitation.

Impact\_p parameter takes an int object as the input and limits the data collecting from the probability of impact. The function will only collect data of asteroids with impact probability larger than 10-<impact\_p>. The available range for impact\_p value is [0, 10].

The function returns a dictionary of the data of asteroids. The items in the returning dictionary have the format of “aid:{diameter, impact\_probability}”. aid is the unique id of the asteroid, diameter is the estimated diameter of the asteroid, and impact\_probability is the exact probability of Earth impact event of the asteroid.

**get\_ca\_data(start\_date)**: get data of asteroids that are flying towards solar planets or moon. The function takes only 1 argument and returns the data of asteroids in the given period.

Start\_date parameter takes a datetime.date object as the input. It decides the date period of the data of asteroids collected. The function will collect data from the given start date to 7 days later.

The function returns a dictionary of the data of asteroids. The items in the returning dictionary have the format of “designation:{velocity, body}”. Designation is the unique id of the asteroid, body is the name of the space object that the asteroid is flying towards, and velocity is the velocity relative to the body.

**store\_neo\_in\_db(dic, db, sheet\_name)**: store the NEO data in a given database. The function takes 3 arguments and returns a None object.

Dic parameter takes a dictionary of NEO data. The format of the dictionary should be the same as the returning dictionary of get\_neo\_data() function. This is the dictionary you want to store in the database. db parameter takes a string of the directory of the database in which you want to store your data. Sheet\_name parameter takes a string of the name of the table in which you want to store your data.

**store\_sentry\_in\_db(dic, db, sheet\_name)**: store the sentry data in a given database. The function takes 3 arguments and returns a None object.

Dic parameter takes a dictionary of sentry data. The format of the dictionary should be the same as the returning dictionary of get\_sentry\_data() function. This is the dictionary you want to store in the database. db parameter takes a string of the directory of the database in which you want to store your data. Sheet\_name parameter takes a string of the name of the table in which you want to store your data.

**store\_cad\_in\_db(dic, db, sheet\_name)**: store the close-approach data in a given database. The function takes 3 arguments and returns a None object.

Dic parameter takes a dictionary of close-approach data. The format of the dictionary should be the same as the returning dictionary of get\_ca\_data() function. This is the dictionary you want to store in the database. db parameter takes a string of the directory of the database in which you want to store your data. Sheet\_name parameter takes a string of the name of the table in which you want to store your data.

**calculate\_volume\_neo(db):** Takes a database as a parameter. Pulls the asteroid ID and diameter from the database, then calculates the volume using the formula V = 4/3 \* pi \*R^3. It then adds the volumes to a list, and applies the ID as the X-axis and volume as the Y-axis. The result is output as a text and a scatter plot.

**calculate\_volume\_sentry(db):** Takes a database as a parameter. Pulls the asteroid ID and diameter from the database, then calculates the volume using the formula V = 4/3 \* pi \*R^3. It then adds the volumes to a list, and applies the ID as the X-axis and volume as the Y-axis. The result is output as a text and a scatter plot.

CAD\_Calc.py

**Overview:**

CAD\_Calc.py connects to the database and selects the velocity for each close approach object near planetary bodies during May and October 2018. For the velocities of objects near the Earth and Moon, there is only one list for each of them since their average velocities are so close. After the velocities are calculated, they are put into a dictionary and then formatted into CAD\_Calc.txt. Finally, the data is arranged into a double bar chart in order of planets closest to the sun.

**averageVelocity(body):** Takes in the list of velocities for objects near a celestial body and computes the average. Since the Earth and Moon had two velocities for each tuple, an if statement had to be used so that the second value would be included in the computation.

**autolabel(rects):** Takes in the rectangles that are being graphed and writes a label above each one showing the exact height.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Issue Description | Resource Location | Result |
| 12/16/2019 | Asteroid volume formula | <https://spacemath.gsfc.nasa.gov/Modules/7Module9.html> | Found volume formula for calculation. |
| 12/2/2019 | Needed API for well-known objects | <http://www.bluebulbprojects.com/MeasureOfThings/api/default.php#rItems> | Requested API key. No response received. |

|  |  |  |  |
| --- | --- | --- | --- |
| 12/2/-12/19/2019 | General Python syntax and reference | <https://docs.python.org/3/> | Syntax refreshers and knowledge. |
| 12/18/2019 | Bar Chart Documentation | <https://matplotlib.org/3.1.1/index.html> | Parameters and other needed to create double bar chart as well as autolabel(rects) function. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |