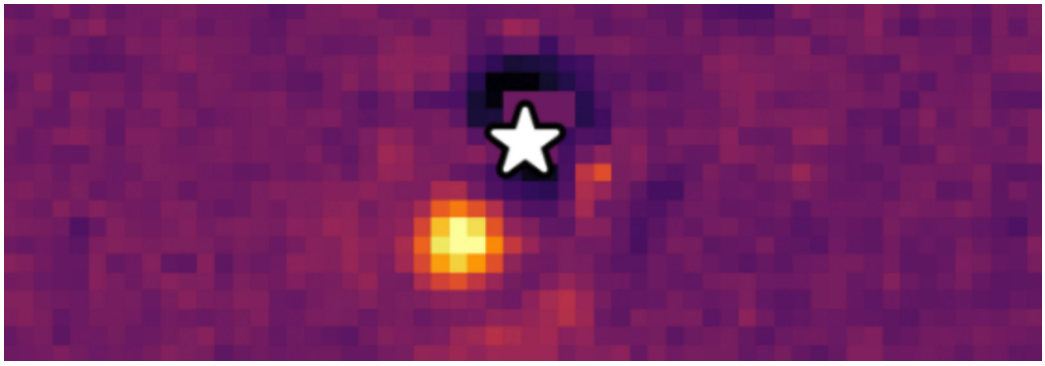
Released: October 11, 2022
Due: Friday, October 21, 2022 @ 9pm ET
Late Due Date (1-48 hours late): Sunday, October 23, 2022 @ 9pm ET

Homework 5: Exoplanets

Introduction
In this homework we will analyze a catalog of over five thousand exoplanets: planets that orbit stars other than the sun. Our dataset is from the Open Exoplanet Catalog. The dataset contains 25 interesting attributes for each planet including its mass, radius, orbital period and surface temperature. It also lists the scientific method used to discover the exoplanet as well as the year of discovery. Many of the exoplanets discovered to date are *hot Jupiters* : gas giants as large as Jupiter or larger orbiting very close to its host star. Finding Earth-like planets is much more difficult. Recently, the newly launched James Webb Telescope has managed to directly image several exoplanets. It is certainly an exciting time to be a Data Scientist collaborating with Astronomers and Astrophysicists!



Source: https://www.quantamagazine.org/webb-space-telescope-snaps-its-first-photo-of-an-exoplanet-20220901/

Assignment Goals

1. Identify the most Earth-like exoplanet using a method for measuring similarities and differences.

2. Visualize the catalog of all discovered exoplanets and their properties to see how most newly discovered exoplanets are different from the Earth and are in fact more like “hot Jupiters”: Jupiter-sized planets orbiting close to their host star.

Expectations
We expect you to write your code in a *modular style*: implement your code as a collection of well-defined functions. Your main() method should do little more than call these functions when needed in order to carry out the required analysis. We don’t want to see all of your code written inside main(). Instead, the main() function should call other defined functions, each of which should perform a singular task and should be called whenever it is needed. Breaking your code into smaller functions makes your code more readable and easier

to test. Functions should be documented with docstrings (triple-quoted strings underneath the function declaration). Your docstring should include a short description, list any input parameters and what each one represents, and declare the return value, if there is one. The instructors will demonstrate this in lecture. As always, you will be graded on a) correctness, b) code design and efficiency, and c) documentation and style.

(link to the DS 2000 style guide: https://course.ccs.neu.edu/ds2000/ds2000_fall2022_styleguide.pdf)

In this assignment, we describe a number of specific functions that we would like you to implement. You are free to add *additional* functions as appropriate.

Requirements

1. Write a function called **read_data** that reads the exoplanet data into a list of lists. (Note that our exoplanet dataset contains the solar system planets as well, including the Earth.) The input to the read_data function is the name of the file (exoplanet.csv). The return value is a list of lists containing our required data - each sublist is the data for one planet. Extract and store only the following SIX planet attributes:

    a. name (string)
    b. mass (float)
    c. radius (float)
    d. orbital period (float)
    e. semimajoraxis (float).
    f. Surface temperature (K) (float)

    Other columns may be ignored. Numeric values should be converted to floats. Note that any line that begins with a hashtag (#) can be ignored. Missing numeric values that show up as empty zero-length strings should be stored as 0.0. When storing the planetary data, make the following conversions:

    - Convert the orbital period measured in days to years. 1 year = 365.2422 days.
    - Convert the mass from Jupiter masses to Earth masses. One Jupiter mass = 317.89 Earth masses.
    - Convert the planet radius from Jupiter radii to Earth radii. One Jupiter radius = 10.97 Earth radii.

    The semi-major axis is approximately the distance of the planet to its host star, measured in Astronomical Units or AU's. (1 AU = 93 million miles, the distance from Earth to the Sun). No conversion is necessary. The surface temperature is measured in degrees Kelvin (K), where 0K represents absolute zero, the theoretical impossible-to-reach limit when atomic motion stops. No temperature conversions are necessary.

2. Write a function called **lookup_planet** that returns the data for a named planet in the form of a list. The input is the name of the planet (from the first column) and the full data set (list of lists). The output is a list of the six planet attributes for the named planet.

3. Write a function called **euclidean_distance** that measures how different two planets are. The formula we will use for measuring how different two planets are is:

$$euclid = ((\Delta Mass)^2 + (\Delta Radius)^2 + (\Delta Period)^2 + (\Delta SemimajorAxis)^2 + (\Delta SurfaceTemp)^2)^{1/2}$$

   Where the Δ-symbol means *difference.* For example, $\Delta Mass$ is the difference in mass between the two planets. The function inputs are the data for two planets (in the form of lists). The output is the Euclidean distance score.

4. Write a function called **find_most_similar_planet** which takes as input the name of a planet and the planetary dataset (list of lists). The function returns the name of the most similar planet. Use this function to determine which exoplanet is most similar to Earth. **The exoplanet with the *smallest Euclidean distance to Earth* is the exoplanet *most similar* to Earth.** Write a separate function called **generate_planet_report** that prints out this matching planet's attributes in a clear human-readable format. Add this report output to your program header so that we can see your result.

5. Write a function called **visualize_exoplanets** that takes all of the data as input and produces a scatter plot of exoplanet Semimajor Axis (y-axis) vs. exoplanet Mass. You may find it helpful to write a separate function that extracts the data for a specified column from the list of lists. You might call this function something like: **extract_column** and have it return a list of every value in that column. When visualizing the exoplanet data, be sure to label your axes. Put both the x-axis and the y-axis on a logarithmic scale:

```
plt.xscale('log')
plt.yscale('log')
```

   On your plot, specifically mark Earth with a red 'X' and add a text label ('Earth') near the 'X'.

What to Submit

Submit your program (**exoplanet.py**) and your visualization (**exoplanet.png** OR exoplanet.pdf OR exoplanet.jpg). **In your program header, include the program text output reporting the planet that is most similar to the Earth and the Euclidean distance score (in case the autograder breaks).**

It is your responsibility to verify that ALL required files have been submitted to gradescope. If you forget to submit your code or any requested outputs your grade will be affected. We cannot accept resubmissions once the grading begins. **Please remember: If you *resubmit* your assignment for any reason, your resubmission must include ALL the required files for the completed assignment. Files from previous submissions are deleted from gradescope and not accessible to the graders!**