Mandatory Assignment –MMN16
Course: Programming and Data Analysis using Python
Assignment Topics: Files and Data Analysis
Relevant Units: Units 13–14
Assignment Weight: 8 Points
Number of Questions: 1
Submission Deadline: June 22, 2025
Semester: 2025A

Please Note:

- The programs **must include internal documentation** (in English only), explaining clearly at the beginning of each script what the program does overall and, throughout the code, what each part does. The documentation must adhere to the **PEP 8** standard: https://peps.python.org/pep-0008/
- You may add auxiliary functions beyond those explicitly required in the assignment, provided they contribute meaningfully.
- Focus on the **functional completeness** of your submission. You are encouraged to present graphical outputs (charts/plots), provided they are as aesthetically clear as possible **and meet the task's requirements**.
- **Do not use advanced material not covered in the course.**
- **Use constants where applicable.**
- Be meticulous about **proper indentation (alignment)** and use **meaningful variable names (in English)** consistent with course conventions.
- Ensure that your output **strictly follows** the format specified in the question—correct spelling, case sensitivity (uppercase/lowercase), spacing, etc.
- Assignments may be submitted **only** through the **online course assignment system**.
- Don't forget to **save the submission confirmation number** after uploading your assignment.

Question 1 (100 Points)

On the course website, you will find a file named **nasa.csv** containing information about asteroids in space that have had close approaches to Earth. The file is in **CSV format**.

Units **13.1** and **13.4** provide a detailed explanation of how to read and process data from CSV files.

As part of this assignment, you are required to **prepare and clean the raw data** from the CSV file, **analyze** it, and **present key findings**.

Stage 1: Data Cleaning and Preparation

A. Write a function named `load_data` that:

- Accepts a CSV file name.
- Returns a DataFrame using **Pandas** (denoted as `df`).
- Handles relevant exceptions (e.g., file not found, invalid file format, file unavailable) and displays an appropriate message to the standard output.

Assumption: The file `nasa.csv` is located in the same directory as the Python script for this project.

Note for students using Google Colab: You must upload the file manually to the **sample_data** folder in your notebook environment.
To do this, right-click the `sample_data` folder on the left panel and choose **"Upload"**, then use the file path in your program (via **copy path**).

Be aware that files may disconnect or be removed between sessions, so you might need to re-upload the file again later.

B. Write a function named `mask_data` that:

- Accepts a DataFrame (`df`).
- Filters and returns a new DataFrame containing only **asteroids with a close approach date from the year 2000 onward**.

C. Write a function named `data_details` that:

- Accepts a DataFrame (`df`).
- Cleans it by selecting the following columns only:
    - `Neo Reference ID`
    - `Orbiting Body`
    - `Equinox`

- Returns a static list of tuples representing rows from the table, **including three rows, column headers, and the total number of rows**.

Stage 2: Data Analysis

(This stage should be performed after the data cleaning step above.)

D. Write a function named `max_absolute_magnitude` that:

- Accepts a DataFrame (`df`).
- Returns a **tuple** where:
  - The first element is the **name of the asteroid** with the **highest absolute magnitude** (`Absolute Magnitude` column).
  - The second element is the **value** of this maximum magnitude.

E. Write a function named `closest_to_earth` that:

- Accepts a DataFrame (`df`).
- Returns the **name** of the asteroid that came **closest to Earth**, based on the column `Miss Dist.(kilometers)`.

F. Write a function named `common_orbit` that:

- Accepts a DataFrame (`df`).
- Returns a **dictionary** where:
  - The **keys** are orbit identifiers (`Orbit ID`).
  - The **values** are the **number of asteroids** associated with each orbit.

G. Write a function named `min_max_diameter` that:

- Accepts a DataFrame (`df`).
- Returns the **number of asteroids** whose **maximum estimated diameter** in kilometers
  (`Est Dia in KM(max)`) exceeds the **average** of all maximum diameters.

Stage 3: Data Presentation

*(This stage should be carried out **after** completing the data cleaning and analysis stages.)*

General Guidelines:

- Use the **updated DataFrame (`df`)** prepared during the data cleaning stage.
- Use the **matplotlib** package to present data visually.
- Each chart must include:
  - **Title**, **legend**, and **axis labels** with explanations of the values displayed.
- Whenever possible, use the functions from the **Data Analysis** stage to support the visuals.
- You may also write and use additional **helper functions**, as needed.
- Graphical outputs should be presented **as clearly and aesthetically as possible**, but the **main focus** should be on their **functional value** in addressing the assignment's objectives.

H. Write a function named `plt_hist_diameter` that:

- Accepts a DataFrame (`df`).
- Displays a **histogram** showing the **number of asteroids** according to their **average diameter in km**, divided into **intervals of 100 km**.

The average diameter for each asteroid is the **mean** of its minimum and maximum estimated diameters:
(`Est Dia in KM(min)` and `Est Dia in KM(max)`)

I. Write a function named `plt_hist_common_orbit` that:

- Accepts a DataFrame (`df`).
- Displays a **histogram** showing the **number of asteroids per orbit**, based on their `Minimum Orbit Intersection` (MOID).

The graph should be divided into **continuous orbit range bins of size 10**, starting from the minimum MOID value up to the maximum.

J. Write a function named `plt_pie_hazard` that:

- Accepts a DataFrame (`df`).
- Displays a **pie chart** showing the **percentage** of asteroids classified as **hazardous** versus **non-hazardous**, based on the `Hazardous` field in the DataFrame.

Page 5 – Translation:

K. Write a function named `plt_linear_motion_magnitude` that:

- Accepts a DataFrame (`df`).
- Checks whether there is a **linear correlation** between:
  - The asteroid's **absolute magnitude** (`Absolute Magnitude`), and
  - The asteroid's **miss distance from Earth in kilometers** (`Miss Dist.(kilometers)`),
  - As well as its **velocity in miles per hour** (`Miles per hour`).
- Create a **graph** to illustrate the relationship.
- Additionally, in your own words, **explain** in the function's documentation whether there appears to be a **simple linear regression** pattern or **correlation** between these variables.

Documentation Guidelines:

For each function, follow the documentation conventions outlined in **Unit 1.9**.
Include a **docstring** that clearly explains the function's purpose, inputs, and outputs.

Submission Instructions

1. The TMA must be submitted **electronically only**, via the assignment submission system.
2. You must submit a file named `py_dsnasa_asteroid.py`.
   Alternatively, you may also submit the `nasa.csv` file (optional).
3. If you choose to work in **Google Colab**, download the notebook in `.py` format and submit that file.
4. Package your solution file into a **RAR archive** (not ZIP!) and upload it.
5. Don't forget to **save the submission confirmation number** provided by the system.

If you did not receive a confirmation number, the submission was **not** received.

Note: You may resubmit multiple times up until the deadline. Each new submission **overwrites** the previous one.

However, if the teaching assistant has already downloaded your assignment, you **won't** be able to send an updated version.

Good luck! 🍀