National College of Ireland

National College of Ireland
Programming for Artificial Intelligence (H9PAI)

Programme: MSc in Artificial Intelligence - MSCAI1B
Continuous Assessment (30%)

Release Date: 16th October 2024
Submission Date: 15th November 2024

General Instructions:

- **API Key Usage:** Do not hardcode your API key directly in the script. Instead, use environment variables to store sensitive information such as API keys. You can retrieve the API key in your script using Python's os library:

```
import os
api_key = os.getenv('API_KEY')
```

  Before submission, ensure that your API key is removed from the code and provide instructions on how to set up environment variables in the *README.md* file.

- Comment your code thoroughly. Each function, loop, and significant code block must be clearly explained.
- Ensure your code is modular and functions are properly used to separate tasks. Meaningful variable and function names should be used to make your code readable and easy to follow.
- It is strongly encouraged that you encapsulate the main logic in a block like the following:

```
if __name__ == "__main__":
    main()
```

  Alternatively, you can also submit the responses in a jupyter notebook with appropriate labelling through markdown

- When reading from or writing to files (e.g., JSON, CSV), ensure that files are not overwritten unnecessarily. Append data to existing files where applicable.
- Do not use libraries that completely automate tasks you're supposed to learn, such as APIs or libraries that abstract away JSON handling or API requests.
- Use of AI tools (e.g., ChatGPT, GitHub Copilot, or any other AI-based coding assistance tools) is strictly prohibited.
- This is an **individual assessment**, and all work must be completed independently. Submissions will be checked for academic integrity violations.

- **Submission Requirements**: Submit a .zip file containing:
  - Your Python scripts (in `.py` or `.ipynb` format)
  - A short report (in PDF or Word format) that includes:
    * The outputs for each part of the assessment (e.g., printed data, screenshots of visualizations, CSV or JSON file excerpts, etc.).
    * Explanations of key outputs or decisions made (e.g., any challenges faced in fetching data, handling errors, or processing large files).
  - A README.md file that explains how to set up the environment (including how to configure environment variables for the API key).

Short Report Guidelines

- Your report should be concise (maximum 4 pages) and contain:
- A brief explanation of your approach for each part of the assessment as well as answer to the questions asked in Problem-4.
- Screenshots or excerpts showing the outputs (where applicable).
- For each section, highlight any challenges faced or important decisions made during implementation (e.g., handling large JSON files, error handling techniques, etc.).

1 Problem 1: NASA APOD Data Retrieval and JSON File Processing (33 marks)

You will use the NASA APOD API to retrieve data about NASA's Astronomy Picture of the Day for an extended range of dates and store the data in a JSON file.

1. **API Call (10 marks)**
   - Write a Python function `get_apod_data(api_key, date)` that sends a GET request to the NASA APOD API to fetch Astronomy Picture of the Day data for the given date. The function should return:
     - Date of the picture
     - Title of the picture
     - URL of the image or video
     - Explanation of the picture
     - Media type (image or video)
   - The function should handle potential exceptions (e.g., network issues, invalid date, or incorrect API key) and print meaningful error messages.
2. **Fetching Data for a Date Range (15 marks)**
   - Write a function `fetch_multiple_apod_data(api_key, start_date, end_date)` to retrieve APOD data for all dates within a large enough date range (e.g., one year's worth of data, from January 1, 2020, to December 31, 2020).
   - Ensure that you respect the API rate limits by adding a 1-second delay between requests.
   - Use a loop to iterate through the date range and call `get_apod_data()` for each date. The data for each date should be appended to a JSON file without overwriting the previous data.
3. **Saving Data to a JSON File (8 marks)**
   - Modify the `fetch_multiple_apod_data()` function to append the retrieved data to a JSON file named `apod_data.json`. If the file doesn't exist, create it. Each entry should include the date, title, URL, explanation, and media type.
   - Ensure proper exception handling when writing to the file (e.g., handle file I/O errors such as permission issues).
   - **Important:** Ensure that the file size is significant by fetching data for at least 200-300 days to generate a large JSON file.

2 Problem 2: JSON Data Reading, Looping, and Processing (27 Marks)

In this part, you will use the JSON file `apod_data.json` (which should now contain data for 200-300 days) to perform various data analysis tasks. Implement exception handling and use loops for file reading and data processing.

1. **Reading and Loading JSON Data with Exception Handling (10 marks)**

- Write a function `read_apod_data()` that reads the `apod_data.json` file and loads its content into a Python dictionary.
- Use a try-except block to handle potential file errors such as file not found, permission denied, or empty file. Print appropriate messages for each case.
- Once loaded, loop through the JSON data and print the date and title of each entry.

2. **Processing and Summarizing Data Using Loops (10 marks)**
   Using a loop, write a function `analyze_apod_media()` that reads the `apod_data.json` file and:
   - Counts the total number of images and videos in the file.
   - Identifies the date with the most detailed explanation (i.e., the entry with the longest explanation text).
3. **Extracting and Writing Data to a CSV File (7 marks)**
   - Extract the following fields for each entry: date, title, media type, and URL. Write this information to a CSV file named `apod_summary.csv`. Include appropriate column headers. Ensure the program can append new entries to the CSV if it already exists. Use a loop to write each record to the CSV file.

Problem 3 - Numpy Array Manipulation and Statistical Functions (18 marks)

1. Create a 2D NumPy array with 100 random integers between 10 and 100, arranged into 20 rows and 5 columns. The array must satisfy the following conditions: **(7 marks)**
   - The sum of each row must be even.
   - The sum of all values in the array must be a multiple of 5.

   Hint: Use conditional operations to adjust the random values where necessary.
2. Using array indexing and loops, perform the following tasks: **5 marks)**
   - Extract and print all elements in the array that are divisible by both 3 and 5.
   - Replace all elements in the array that are greater than 75 with the mean of the entire array.
3. Perform some statistical operations on the array. Calculate the mean and standard deviation of all the values in the array. Find the median value of the array. Compute the variance for each column in the array. **(6 marks)**

Problem 4 - Working with Pandas DataFrames (22 marks)

In this problem, you'll work with pandas DataFrames, reading them into and out of memory, changing their contents, and performing aggregation operations. For this problem, you'll need to download the celebrated `iris` dataset, available as a .csv file from Moodle. Note: for the sake of consistency, please use this version of the CSV and not one from elsewhere (e.g., downloaded from the UC Irvine repository).

1. Download the iris data set from the link above. Please include this file in your submission. Read iris.csv into Python as a pandas DataFrame. Note that the CSV file includes column

headers. How many data points are there in this data set? What are the data types of the columns? What are the column names? The column names correspond to flower species names, as well as four basic measurements one can make of a flower: the width and length of its petals and the width and length of its sepal (the part of the pant that supports and protects the flower itself). How many species of flower are included in the data? (**4 marks**)

2. It is now known that this data contains errors in two of its rows (see the documentation at https: `https://archive.ics.uci.edu/dataset/53/iris`. Using 1-indexing, these errors are in the 35th and 38th rows. Correct the 35th row to 4.9, 3.1, 1.5, 0.2, "setosa" and the 38th row to 4.9, 3.6, 1.4, 0.1, "setosa". Confirm the changes and display the corrected rows. (**3 marks**)
3. Add two new features to the dataset: (**2 marks**)
   - **Petal Ratio:** Defined as the ratio of petal length to petal width.
   - **Sepal Ratio:** Defined as the ratio of sepal length to sepal width.

   Save the modified DataFrame as `iris_corrected.csv`.
4. Calculate the pairwise correlation between all numeric columns (including the new Petal Ratio and Sepal Ratio columns). Identify and interpret the two pairs of features that have the highest positive correlation and the highest negative correlation. What can you infer from this? (**4 marks**)
5. Create a scatter plot with Sepal Ratio on the x-axis and Petal Ratio on the y-axis, coloring the points according to species. Additionally, add a linear regression line for each species to show the trend between these two ratios. Save the plot as `iris_scatter_with_regression.pdf`. (**5 marks**)
6. Create a pair plot (using Seaborn or an equivalent tool) for the four original numeric features (sepal length, sepal width, petal length, petal width) and the two new ratio features. This will show the relationships between all features, colored by species. (**4 marks**)