L03 Exercise: Python Programming

Complete the following exercise problems and demo the results individually or as a team.
Total score: 50

You are strongly encouraged to utilize a Large Language Model (LLM) (e.g., ChatGPT, CoPilot, Gemini) to assist with this exercise. However, consider the LLM as a learning tool or assistant to help you understand the required subjects and programming techniques. Your goal is to acquire the necessary knowledge and programming techniques, grasp the problem-solving process, and develop the skills to complete this exercise independently. This approach will prepare you to tackle more complex problems in the future.

1. **Setting up the application development environment** [10 points]
   (a) Install the Python interpreter and IDE tool (such as VS code or JupyterLab), add any necessary extensions of your choice to the IDE, and create a virtual environment "venv". Optionally, set up additional configurations such as version control and source code management for team collaboration.

   (b) Review the Python tutorial if needed and familiarize yourself with the "pip" command for installing Python packages. Ensure that all packages are installed within the virtual environment created in step (a).

   (c) Write a simple Python program "**rand_gen.py**" that:
   - Declare a one-dimensional Python array (Python built-in, not NumPy) capable of storing 10 floating-point numbers.
   - Populate the array with random values within the range [-10.0, +10.0].
   - Display all the elements of the array.

2. **Basic data engineering tasks for dataset preparation and preprocessing** [30]
   (a) Write a Python program called "**data_generator.py**" that:
   - Declares a two-dimensional Python array capable of storing 100,000,000 pairs of floating-point numbers (y, x), similar to records in a table.

- Randomly generates x values in the range [-1000.0, +1000.0] and computes the corresponding y values using the equation $y = 2.0 + 0.5x$. In the general form of a polynomial equation $y = w_0 + w_1 x_1 + \cdots + w_n x_n$, x is the "independent" variable (input), y is the "dependent" variable (output), and $[w_0, w_1, \ldots, w_n]$ (e.g., [2.0, 0.5]) are the coefficients.
- Populates the array with the generated pairs and display the first, middle, and last pairs (records).
- Saves the data in the array into two files: "**L1.json**" and "**L1.csv**". "L1.json" should be a JSON format file that stores data as key-value pairs. The key should be "linear" and the value should be a nested list of pairs, e.g., "linear: [[2.0,3.0], [0.0,1.1], …, [-1.2,3.0]]. "L1.csv" should be a comma separated CSV file. Each pair of data (record) should be stored in a separate row with two columns (in a table).
- Declares another two-dimensional Python array capable of storing 100,000,000 real value pairs (y, x), where x values are randomly generated within the range [-1000.0, +1000.0] and y values are computed using the equation $y = 2.0 + 0.5x - 3x^2$.
- Populates the array with the generated data with three columns (y, x_1, x_2) where x_1 is calculated by $0.5x$ and x_2 is calculated by $-3x^2$ and saves this new data in two CSV files: "**Q1.csv**" with two columns y and x and "**Q2.csv**" with three columns (y, x_1, x_2).

(b) Write a Python program "**data_loader.py**" to define the necessary functions that:

- Read and load the data stored in each of the file "L1.csv", "L1.json", "Q1.csv", and "Q2.csv" into the appropriate Python array data structures.

Since the functions defined in this program will be used in other Python program files, they should be written in a way that allows the "data_loader.py" file to be imported and used as a module.

(c) Write a Python program "**data_processor1.py**" that

- Uses the functions defined in "data_loader.py" to load the data from the files "Q1.csv" and "Q2.csv" into appropriate Python arrays.
- Calculates Mean and Standard Deviation for each column in the loaded data.
- Identifies and displays the values greater than two standard deviations or one standard deviation (if there are no such extreme values – two standard deviations away). These values are considered outliers.

- Removes outliers and normalize the data so that all values fall within the range [0,1], and the mean is adjusted to 0.
- Displays 10 randomly selected values from the array to ensure all values are within [0.1] range.
- Loads the normalized data into Panda DataFrames and NumPy arrays (different from Python arrays) and display 10 randomly selected values from each DataFrame and NumPy array.
- Splits the data stored in NumPy arrays into x and y values, saving them into two separate arrays: "X_data" and "Y_data".

3. **Some vector and matrix operations** [10]

   (a) Write a Python program "**dot_product.py**" that:

   - Loads the x values from "L1.csv" and from "Q1.csv" into two Python arrays, respectively.
   - Performs element-wise multiplication between the x values from both arrays using a Python loop statement.
   - Displays the sum of all the multiplications $\sum_i x_{L1} x_{Q1}$ and the time taken to complete the multiplication. If the arrays are vectors, the element-wise multiplication is called "dot-product" that results in a scalar value, indicating the similarity between the two vectors (arrays).

   - Repeats the same task but this time loads the data into NumPy arrays, perform element-wise multiplication using the NumPy function **np.dot(a,b)** without using any loop, where a and b are NumPy arrays, treated as vectors, and display the sum and the time taken for the multiplication.
   - Loads the data from "Q2.csv" into both Python arrays and NumPy arrays, compute the square of the arrays (like matrix multiplication A^2), and display the computation time taken for each multiplication.

   Briefly explains why the time taken for multiplication differs, even when using the same computer.

What to submit

- A report file in Word or PDF format that includes the name or names of the team members, and the % contribution of each member. If there is no agreement on individual contributions, briefly describe the tasks assigned and completed by each member. Different scores may be awarded based on individual contributions. If all members contributed equally, simply state “equal contribution.” The report should also include descriptions of the answers (if questions are asked) and a portion of the screenshot demonstrating the functionality of each program.
- **Each program file individually**. DO NOT submit any zip file as **Canvas cannot open them**.
- Only one report file and the program file(s) for the entire team.

Grading criteria

- Does each program successfully perform the required tasks and generate the correct results?
- Does the report demonstrate that the team has a clear understanding of the relevant concepts, programming techniques, functional requirements? Does it show how the team applied this knowledge to successfully complete the tasks?
- Is the effort put into the project clearly reflected in both the report and the program files?