Solving complex problems with Python

Objectives

Development of Python modules to solve complex problems

- Develop Python modules and classes
- Use test-driven development
- Learn how to work with exceptions
- Familiarize with special libraries e.g. numpy, matplotlib

Deadlines

- ***Lab 8****:* specified features from Iteration 1 and Iteration 2 *(work during the same lab)*
- ***Lab 9****:* all features from Iteration 1 and Iteration 2 with tests in PyUnit *(homework from Lab 8)* and add new features in Iteration 2 *(work during the same lab)*
- ***Lab 10****:* add controller layer and implement Iteration 3 *(homework)*

Requirements

1. Implement a solution for the following problem using classes and feature driven development
2. The solution should offer a console type interface that allows the user to input the data and visualize the output
3. Use only the standard and compound data types available in Python

The application should be developed along several iterations and the solution should ensure:

- Providing at least 10 data examples in the application
- Documentation and testing of each function (at least 5 assertions)
- Validation of data – when the user introduces invalid commands or data, a warning should be generated

Problem specification

A **math teacher** needs a program that helps **students** perform different vector operations.

Iteration 1:

A vector (class ***MyVector***) is identified by the following properties:

- *name_id* given as a string/int
- *colour* given as one letter (possible values ‘r’, ‘g’, ‘b’, ‘y’ and ‘m’)
- *type* given as a positive integer greater or equal to 1
- *values* given as a list of numbers

The following features are offered by the program (to be implemented in class **MyVector**):

1. **Scalar operations**
    a. Add a scalar to a vector
    ```
    e.g. [1,2,3] + 2 = [3,4,5]
    ```
2. **Vector operations**
    a. Add two vectors
    ```
    e.g. [1,2,3] + [4,5,6] = [5,7,9]
    ```
    b. Subtract two vectors
    ```
    e.g. [1,2,3] - [4,5,5] = [-3,-3,-2]
    ```
    c. Multiplication
    ```
    e.g. [1,2,3] * [4,5,5] = 29
    ```
3. **Reduction operations**
    a. Sum of elements in a vector
    ```
    e.g. for [1,2,3] sum is 6
    ```
    b. Product of elements in a vector
    ```
    e.g. for [1,2,3] product is 6
    ```
    c. Average of elements in a vector
    ```
    e.g. for [1,2,3] average is 2
    ```
    d. Minimum of a vector
    ```
    e.g. for [1,-2,3] minimum is -2
    ```
    e. Maximum of a vector
    ```
    e.g. for [1,2,-3] maximum is 2
    ```

Iteration 2:
The program manages several vectors (class ***VectorRepository***) and allows operations such as:

1. Add a vector to the repository
2. Get all vectors
3. Get a vector at a given index
4. Update a vector at a given index
5. Update a vector identified by name_id
6. Delete a vector by index
7. Delete a vector by name_id
8. Plot all vectors in a chart based on the type and colour of each vector (using library *matplotlib*). Type should be interpreted as follows: 1 – circle, 2 – square, 3 – triangle, any other value – diamond.

Add the following features:

9. Get the sum of elements in all vectors.
10. Get the vector which represents the sum of all vectors.
11. Get the list of vectors having a given sum of elements.
12. Get the list of vectors having the minimum less than a given value.
13. Get the sum of all the elements in those vectors having a given colour.
14. Get the max of all vectors having the sum greater than a given value.
15. Get the min of all vectors.

16. Get a list of values representing the multiplication of consecutive vectors in the repository.
17. Delete all vectors from the repository.
18. Delete all vectors for which the colour is a given value.
19. Delete all vectors for which the product of elements is greater than a given value.
20. Delete all vectors that are between two given indexes.
21. Delete all vectors for which the max value is equal to a given value.
22. Update all vectors by adding a given scalar to each element.
23. Update a vector identified by name _id.
24. Update the colour of a vector identified by name_id.
25. Update all vectors having a given type by setting their colour to the same given value.

Iteration 3:
Implement all features from iteration 1 using special libraries e.g. numpy.