Solving complex problems with Python

Objectives

Development of Python modules to solve complex problems

- Develop Python modules and classes
- Work with standard and compound data types in Python
- Use test-driven development
- Familiarize with special libraries e.g. matplotlib

Deadlines

- ***Lab 6****:* all features from Iteration 1 and specified features from Iteration 2 *(work during the same lab)*
  Upload your solution online before the end of the Lab6.

- ***Lab 7****:* extra features added to Iteration 2 *(work during the same lab)*
  Upload your solution online before the end of the Lab7.

- ***Lab 8:*** deadline to finalize the entire application
  Upload your solution online before the start of the Lab8.

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 solution should ensure:

- Providing at least 10 data examples in the application
- Documentation and testing of each function
- 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 simple operations with points in two-dimensional space.

Iteration 1

A point (class ***MyPoint***) is identified by the following properties:

- *coord_x* given as a number
- *coord_y* given as a number
- *colour* given as string (possible values 'red', 'green', 'blue', 'yellow' and 'magenta')

The following features are to be provided (at the level of class ***MyPoint***):

1. Get and set the value of all properties for a point.
2. Provide the string representation of a point.
   For example, for a point with coordinates coord_x = 1, coord_y = 2 and colour = "red", the string format should be *"Point (1, 2) of colour red."*

Iteration 2

The program manages several points (class ***PointRepository***) and allows operations such as:

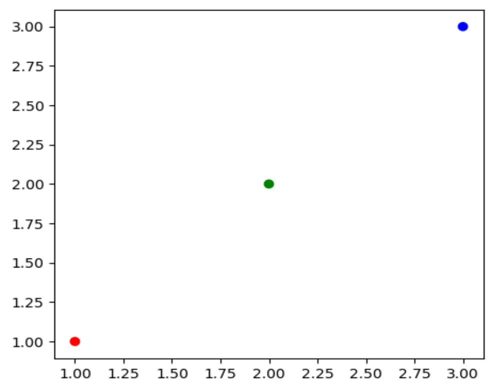
1. *Add a point to the repository*
2. *Get all points*
3. *Get a point at a given index*
4. Get all points of a given colour
5. *Get all points that are inside a given square (up-left corner and length given)*
6. Get the minimum distance between two points
7. *Update a point at a given index*
8. *Delete a point by index*
9. Delete all points that are inside a given square
10. Plot all points in a chart (using library *matplotlib*)

Note

Matplotlib (https://matplotlib.org/index.html) is a special library useful for creating quality figures such as plots, bar charts, scatterplots and histograms.

For example, to plot 3 points with coordinates (1,1), (2,2), (3,3) you can use a code like:

```
import matplotlib.pyplot as plt
x = [1, 2, 3]
y = [1, 2, 3]
col = ["red", "green", "blue"]
plt.scatter(x, y, c = col)
plt.show()
```



PointRepository: examples of extra features

Iteration 2

The program manages several points (class ***PointRepository***) and allows operations such as:

1. Get all points that are inside a given rectangle (up-left corner, length and width given)
2. Get all points that are inside a given circle (centre of circle, radius given)
3. Get the maximum distance between two points
4. Get the number of points of a given colour
5. Update the colour of a point given its coordinates
6. Shift all points on the x axis
7. Shift all points on the y axis
8. Delete a point by coordinates
9. Delete all points that are inside a given circle
10. Delete all points within a certain distance from a given point