823G5 Programming in Python

Aim:

The aim of these exercises is to develop practical skills in scientific computing using Python's math, cmath, and numpy modules. Participants will practice solving real-world problems, including calculating areas of geometric shapes, solving quadratic equations, performing basic matrix operations, and finding the inverse of matrices to solve systems of linear equations. These exercises will help participants build a strong foundation in fundamental scientific computing concepts and gain hands-on experience with essential Python libraries used in data science, engineering, and mathematics.

Exercises

Download the **Lab8Exercises** ZIP file for solutions. The code is a set of Python source code file (.py). You will need to open a new PyCharm project and import the .py files into it. Refer to the Lab 1 instructions if you cannot remember how to do this.

Coing Challenge 1: Basic Scientific Computing

In this challenge, we will explore basic scientific computing using Python's math module.

1) **Calculate Area**
   Write a Python program that calculates the area of geometric shapes. The program should ask the user to choose a shape (e.g., circle, rectangle) and input the required parameters. You should write separate functions for each shape (e.g., calculate_circle_area, calculate_rectangle_area) and call the appropriate function based on the user's choice.

2) **Solve Quadratic Equation**
   Write a Python program that solves a quadratic equation of the form $ax^2 + bx + c = 0$. The program should ask the user to input the coefficients a, b, and c, and then solve the equation. You should handle cases with real and complex roots and print the solutions.

Challenge 2: Advanced Scientific Computing

In this challenge, we will delve into advanced scientific computing using Python's numpy module.

1) **Matrix Operations**
   Write a Python program that performs matrix operations using the numpy library. The program should ask the user to choose an operation (e.g., addition, subtraction, multiplication) and input the required matrices. You should write separate functions for each operation (e.g., matrix_addition, matrix_subtraction, matrix_multiplication) and call the appropriate function based on the user's choice.

2) **Matrix Inverse and System of Equations**
   Write a Python program that finds the inverse of a square matrix and solves a system of linear equations using the numpy library. The program should ask the user to input a square matrix and a column vector. You should find the inverse of the matrix and solve the equation Ax = B, where A is the matrix and B is the column vector.