1 Point and Line
Create two classes: one class representing a point on a plane, with x and y coordinates as attributes and constructor arguments. The second class representing a line with attributes a and b, defining the equation y = ax + b. Add a method to the Point class that takes an object of the Line class as an argument and returns True or False depending on whether the point lies on the line or not. Add a method to the Line class that finds the point where the line intersects the x-axis or prints a message that such a point does not exist.

2 Rectangle
Create a class representing a rectangle. The constructor of this class must take two objects of the Point class as arguments. The two points defining the rectangle must lie on a diagonal. Implement a method that calculates the sides of the rectangle based on the points provided to the constructor. The class should have methods to calculate the area and perimeter of the rectangle. The class should also have a method that draws the defined rectangle using the matplotlib library and marks the points with which it was defined.

3 Note and Notebook
Create classes Note and Notebook. The Note class stores the author, content, and creation time (author and content are provided as constructor arguments, while the time is obtained and saved when the object is created). The constructor of the Notebook class takes no arguments but creates an empty list to which objects of the Note class will be added. The Notebook class must have methods that allow you to add a new note, add an existing note, check how many notes have been added, and display all added notes. Additionally, the case where the notebook is empty must be handled.

4 Fraction
Write a class representing a fraction in the form a/b, where a is the numerator and b is the denominator. The resulting class should meet the following conditions and demonstrate the operation of the implemented class:

• We create an object from two numbers (numerator and denominator). When creating a fraction class, it should be reduced (you can use the gcd method from the math module). If the denominator is zero, a ZeroDivisionError exception should be thrown.
• Two methods for printing objects of our class: __repr__ and __str__. The representation must return commands with which our object can be created, and conversion to a string must show the fractional form.
• We store the numerator and denominator as read-only variables (private variable and @property).
• Objects of the class must support basic arithmetic operations (operator overloading): addition, subtraction, multiplication, division, absolute value. A new object should be created when performing an operation. All operations are performed only on objects of our class (there is no need to implement multiplication by a number, etc.).
• Objects of this class can be compared (6 operators to overload).
• The class must have implementations of casting methods to other data types (float, int, bool).
• The class must have an implementation of the __round__ method to enable rounding while casting to float.