Lab. 10: Robot

Operator Overloading

Python supports operator overloading. To overload the arithmetic operators, +, −, ⋆, and /, just define `__add__`, `__sub__`, `__mul__`, and `__truediv__`, respectively. If you define the following `Point` class, for example, you may define `__add__` as follows to add two points using the operator +:

```
1  class Point:
2      def __init__(self, x, y):
3          self.x, self.y = x, y
4      def __add__(self, q):
5          return Point(self.x + q.x, self.y + q.y)
6      def __repr__(self):
7          return f"Point({self.x}, {self.y})"
8
9  if __name__ == "__main__":
10     ps = [Point(1, 2), Point(2, 3)]
11     ps += [ps[0] + ps[1]]
12     for p in ps:
13         print(p)
```

Programming Lab 10: robot.py

Given a sequence of two-dimensional displacements, calculate the new locations of the robot located at the origin accumulating the displacement, say (x_d, y_d) where x_d and y_d are the displacements to the x- and y-coordinates, respectively. Given the following five displacements for example,

$$(2,3), (-1,2), (0,-5), (3,3), (1,1)$$

the sequence of the robot's locations is changed as follows:

$$(0,0) \to (2,3) \to (1,5) \to (1,0) \to (4,3) \to (5,4)$$

Write a program to calculate the closest location of the robot from the origin in a two-dimensional Cartesian plane. For the above example, the nearest location of the robot is (1, 0) where the distance from the origin is 1.0. Beware that the locations during the movement should be ignored.

Your program should define a class `Loc`, representing the robot's location, with the constructor, the method `__str__`, and other methods as needed. You may define other methods as needed.

Your program is to read from standard input. The input consists of more than a line, containing the x- and y-coordinates of each displacement, separated by a space. Your program is to write to standard output. The output consists of two lines where the first line contains the closest location of the robot in the form of `(X, Y)` where `X` and `Y` denote the x- and y-coordinates of the location. In the second line of the output, the distance of this location from the origin should be printed. The coordinates and the distance should be rounded to the 2nd digit after the decimal point. If there are multiple such locations, print the location whose x-coordinate is larger than the other. If the x-coordinates are also the same, print the location whose y-coordinate is larger than the other.

Additional requirements for bonus points

1. Attach docstrings to the classes and the functions including methods.

2. Use the function `reduce` in the the module `functools`.

Input	**Output**
2 3 -1 2 0 -5 3 3 1 1	(1,00, 0.00) 1.00
1 1 -2 0 0 -2 2 2	(1.00, 1.00) 1.41