Homework 3

(Due: November 07, 2025)

The purposes of this homework are to give you experience in working with classes and objects, and computing numberical solutions to roots of equations. Submit to gradescope a pdf writeup containing the problem description for each problem, your Python code, and approprite textual and graphical output.

Question 1: 10 points. As shown in Figure 1 below, rectangles may be defined by the (x,y) coordinates of corner points that are diagonally opposite.

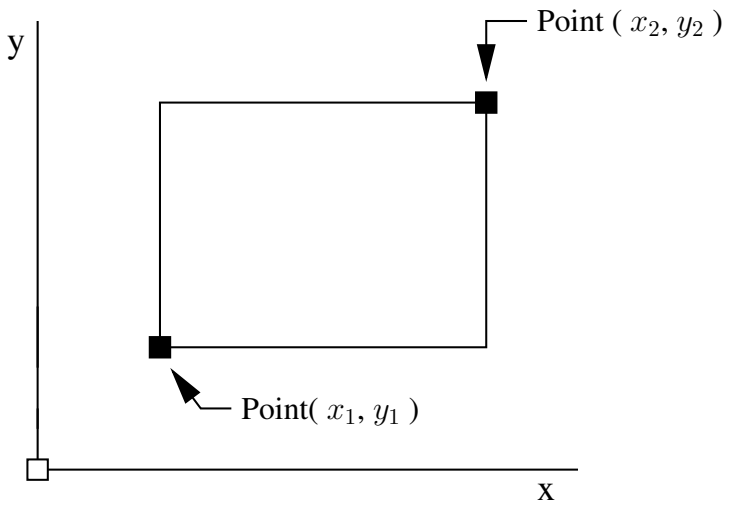

Figure 1: Definition of a rectangle via diagonally opposite corner points.

With this definition in place, the following script of code is a basic implementation of a class for creating and working with rectangle objects.

```
# ========================================================================
# Rectangle01.py: Very basic implementation of rectangle objects, where
# corner points are defined by varibles (x1, y1) and (x2, y2).
#
# Written by: Mark Austin                                    September, 2024
# ========================================================================

class Rectangle:
```

```
  def __init__(self, x1, y1, x2, y2 ):
    self.x1 = x1;
    self.y1 = y1;
    self.x2 = x2;
    self.y2 = y2;
    self.name = "";

  # Set rectangle name ...

  def setName(self, name ):
    self.name = name;

  # Compute perimeter of rectangle ..

  def getPerimeter (self):
     perimeter = 2*(abs(self.x2-self.x1) + abs(self.y2-self.y1))
     return perimeter

  # Compute area of rectangle ..

  def getArea (self):
     area = abs(self.x2-self.x1)*abs(self.y2-self.y1);
     return area

  # String represention of rectangle ...

  def __str__(self):
     rectangleinfo = [];

     ... details removed ...

     return "".join(rectangleinfo);
```

The Rectangle class uses variables x1, y1, x2, and y2 to define the corner points, and has a method to create rectangle objects (i.e, __init__), convert the details of a rectangle object into a string format (i.e., __str__), and compute the rectangle area and perimeter (i.e., getArea() and getPerimeter()).

A script of test program usage is as follows:

```
prompt >>
prompt >> python3 TestRectangle01.py
--- Enter TestRectangle01.main()    ...
--- ============================== ...
--- Part 1: Create and print rectangle A ...

--- Rectangle: A ...
--- ------------------------------------------------
---   Corner Point (x1,y1) = (  1.00,   2.00) ...
---   Corner Point (x2,y2) = (  3.00,   4.00) ...
---   Perimeter =   8.00 ...
---   Area      =   4.00 ...
```

```
--- ------------------------------------------------

--- Part 2: Create and print rectangle B ...

--- Rectangle: B ...
--- ------------------------------------------------
---   Corner Point (x1,y1) = (  0.00,   0.00) ...
---   Corner Point (x2,y2) = (  6.00,   5.00) ...
---   Perimeter =  22.00 ...
---   Area      =  30.00 ...
--- ------------------------------------------------

--- ============================== ...
--- Finished TestRectangle01.main()    ...
prompt >> exit
```

Source Code: The rectangle code and test program can be found in: python-code.d/objects/classes/ ...

Question: Now suppose that instead of using the variables x_1, y_1, x_2 and y_2 to define the corner points, we use the class Point:

```
import math

class Point:

    def __init__(self, xCoord=0, yCoord=0):
        self.__xCoord = xCoord
        self.__yCoord = yCoord

    # get x coordinate

    def get_xCoord(self):
        return self.__xCoord

    ..... details of other functions removed ...
```

The appropriate modification for Rectangle is:

```
from Point import Point

class Rectangle:

   def __init__(self, x1, y1, x2, y2 ):
       self.pt1 = Point(x1,y1)     # <-- create lower corner point ...
       self.pt2 = Point(x2,y2)     # <-- create upper corner point ...

   ..... details rectangle removed ....
}
```

The arrangement of Rectangle and Point classes can be visualized as follows:

Figure 2: Test program script and classes in a rectangle system.

What to do? Fill in the missing details (i.e., constructors and __str__ method) of class Point. Modify the code in Rectangle to use the Point class. The resulting program should have essentially the same functionality as the original implementation (v1) of Rectangle.

Question 2: 20 points. The left-hand side of Figure 3 shows the essential details of a domain familiar to many children. One by one, rectangular blocks are stacked as high as possible until they come tumbling down – the goal, afterall, is to create a spectacular crash!!

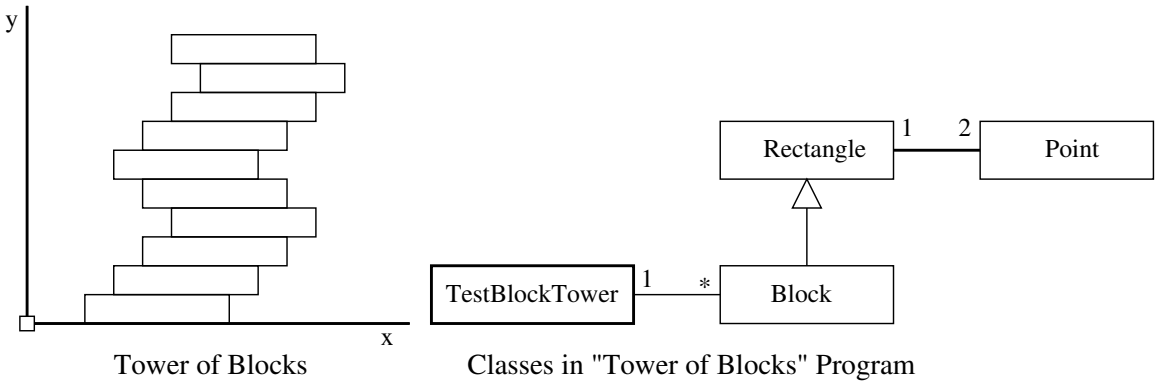

Figure 3: Schematic and classes for tower of blocks problem.

Suppose that we wanted to model this process and use engineering principles to predict incipient instability of the block tower. Consider the following observations:

1. Rather than start from scratch, it would make sense to create a Block class that inherits the properties of Rectangle (previous question), and adds details relevant to engineering analysis (e.g., the density of the block).

2. Then we could develop a BlockTower class that systematically assembles the tower, starting at the base and working upwards. At each step of the tower assembly, analysis procedures should make sure that the tower is still stable.

The right-hand side of Figure 3 shows the relatioship among the classes. One TestBlockTower program (1) will employ many blocks, as indicated by the asterik (*).

Develop a Python program that builds upon the Rectangle class written in the previous questions. The class Block should store the depth and density of the block – this will be important in determining the

mass and centroid of each block. The TestBlockTower class will use block objects to build the tower. A straight forward way of modeling the block tower is with a List. After each block is added, the program should conduct a stability check. If the system is still stable, then add another block should be added. The simulation should cease when the tower of blocks eventually becomes unstable.

Note. To simplify the analysis, assume that adjacent blocks are firmly connected.

Stability Considerations. If the blocks are stacked perfectly on top of each other, then from a mathematical standpoint the tower will never become unstable. In practice, this never happens. There is always a small offset and, eventually, it's the accumulation of offsets that leads to spectacular disaster.

For the purposes of this question, assume that blocks are six units wide, one unit high, and depth of one unit. When a new block is added, the block offset should be one unit. To make the question interesting, assume that four blocks are stacked with an offset to the right, then three blocks are added with an offset to the left, then four to the right, three to the left, and so forth. This sequence can be accomplished with the looping construct:

```
# Compute incremental offset for i-th block .....

offset = math.floor((BlockNo - 1)/5.0) + (BlockNo-1)%5;
if ((BlockNo-1)%5 == 4 ):
   offset = offset - 2;
```

The tower will become unstable when the center of gravity of blocks above a particular level falls outside the edge of the supporting block.

What to do? Write a Python program that will:

1. Determine how many blocks can be added to the stack before it crashes.
2. Create a figure of the block configuration and centroid position immediately before collapse.

Question 3: 10 points. Figure 4 plots the function

$$f(x) = \left[x - \frac{1}{x}\right]^3 + \left[x - \frac{1}{x}\right] - 30. \tag{1}$$

over the range $[-4, 4]$.

Theoretical considerations indicate that equation has two real roots:

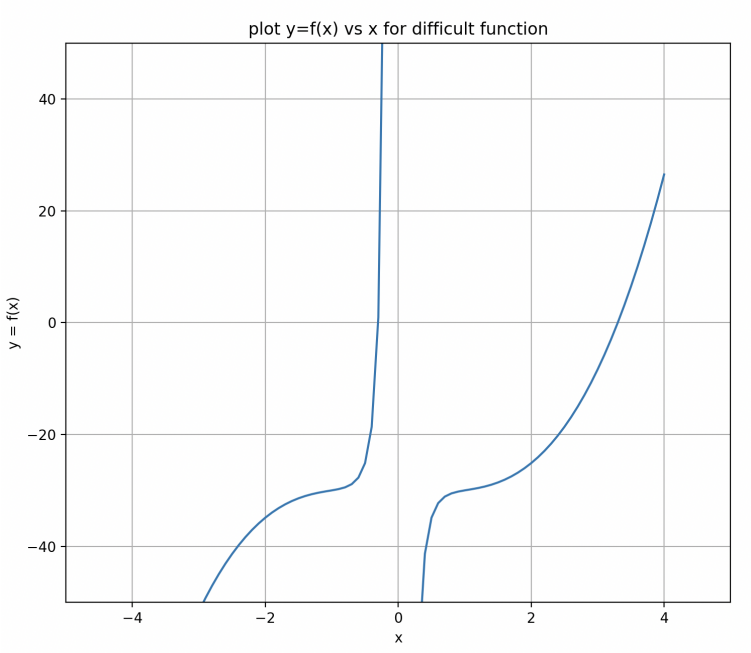

Figure 4: Plot y = f(x) vs x.

$$[r_1, r_2] = \frac{3}{2} \pm \frac{\sqrt{13}}{2} \tag{2}$$

Now suppose that we have Figure 4, and can see that the two roots lie in the intervals $[-1, 0]$ and $[3, 4]$, but for some reason don't know about equation 2.

Write a Python program that:

1. Uses the **method of bisection** to compute the lower and upper roots to equation
2. Computes the roots with the **method of Newton Raphson** iteration.
3. Investigates behavior of **Newton Raphson iteration** when we seek solutions to the lower root and the starting value x_o = -2.

Question 4: 10 points. For values $x > 0$, the nonlinear function

$$f(x) = 1 + \sin(x) \tag{3}$$

has roots at x = $3\pi/2$, $7\pi/2$, $11\pi/2$, and so forth.

Write a Python program to plot equation 3 over the range [0, 4π], and then demonstrate that while the **method of Newton Raphson** struggles to find value(s) of x for which f(x) = 0 (why?), the **method of Modified Newton Raphson** computes the same roots with ease.

Question 5: 10 points. Consider the family of matrix equations AX = B defined by

$$\begin{bmatrix} 1 & 2 & -3 \\ 3 & -1 & 5 \\ 4 & 1 & a^2 - 14 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ a + 2 \end{bmatrix}. \tag{4}$$

Determine the values of 'a' for which matrix A will be singular. Then,

1. Develop a program that uses NumPy to store matrices A and B, and then systematically evaluates the determinant and rank of A, and the rank of augmented matrix [A | B] for the values of 'a' that make A singular.

1. Develop a second program that uses SymPy to store matrices A and B symbolically, and then computes symbolic solution to the matrix equations 4. For the values of 'a' that make A singular, evaluate the determinant and rank of A, and the rank of augmented matrix [A | B].

You should find that the Python module SymPy is considerably more powerful than its numerical counterpart NumPy.