Homework 1

(Due: March 1, 2023)

The purpose of this homework is to get you started with programming in Python. For each question hand in a solution (i.e., program source code + program output), all zipped together into a single file.

Question 1: 5 points. Write a Python program that solves for all integer pairs $a, b \geq 0$,

$$\sqrt{a} + \sqrt{b} = \sqrt{n} \tag{1}$$

where n = 2023.

Question 2: 5 points. Figure 1 shows a two-dimensional grid of masses.

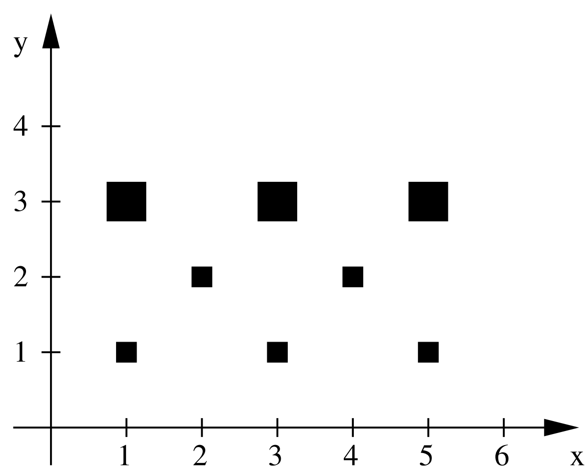

Figure 1: Two-dimensional grid of masses

If the total number of point masses is denoted by N, then the mass moments of inertia about the x- and y-axes are given by:

$$I_{xx} = \sum_{i=1}^{N} y_i^2 \cdot m_i \quad \text{and} \quad I_{yy} = \sum_{i=1}^{N} x_i^2 \cdot m_i \tag{2}$$

respectively. The polar moment of inertia is given by

$$I_{rr} = I_{xx} + I_{yy} = \sum_{i=1}^{N} \left[x_i^2 + y_i^2 \right] \cdot m_i \tag{3}$$

Now suppose that the (x,y) coordinates and masses are stored in two arrays;

```
    mass  = np.array( [ 1.0, 1.0, 1.0, 1.0, 1.0, 2.0, 2.0, 2.0 ] );

    coord = np.array( [ ( 1.0, 1.0 ),
                        ( 2.0, 2.0 ),
                        ( 3.0, 1.0 ),
                        ( 4.0, 2.0 ),
                        ( 5.0, 1.0 ),
                        ( 5.0, 3.0 ),
                        ( 3.0, 3.0 ),
                        ( 1.0, 3.0 ) ] );
```

Write a Python program to evaluate equations 2 and 3. Your program should: (1) use simple looping construct to systematically walk along the array elements, and (2) output the results of each evaluation.

Question 3: 10 points. In class we used the PrintMatrix function, i.e.,

```
def PrintMatrix(name, a):
    print("Matrix: {:s} ".format(name) );
    for row in a:
        for col in row:
            print("{:8.2f}".format(col), end=" ")
        print("")
```

to print the contents of a two-dimension matrix of numbers in a tidy format. The code is limited in at least three respects: (1) If you accidentally pass a one-dimensional array to PrintMatrix(), the code will crash because notions of row and column do not exist, (2) No effort has been made to print the row numbers along the left-hand side of the page, and the column number across the top of the page, and (3) The implementation assumes that when a matrix is printed the contents will fit on a single page (or screen). In practice, this is not true.

Extend the functionality of `PrintMatrix(name, a)` to overcome these limitations. Then, develop a test program to exercise the improved matrix print function. As an example of what you should aim for, the commands:

```
  matrix01 = numpy.arange(30).reshape((3, 10))
  PrintMatrix("matrix01 (reshaped)", matrix01)
```

would generate the output:

```
Matrix: matrix01 (reshaped)
row/col             1             2             3             4
   1      0.00000e+00   1.00000e+00   2.00000e+00   3.00000e+00
   2      1.00000e+01   1.10000e+01   1.20000e+01   1.30000e+01
   3      2.00000e+01   2.10000e+01   2.20000e+01   2.30000e+01

Matrix: matrix01 (reshaped)
row/col             5             6             7             8
   1      4.00000e+00   5.00000e+00   6.00000e+00   7.00000e+00
   2      1.40000e+01   1.50000e+01   1.60000e+01   1.70000e+01
   3      2.40000e+01   2.50000e+01   2.60000e+01   2.70000e+01

Matrix: matrix01 (reshaped)
row/col             9            10
   1      8.00000e+00   9.00000e+00
   2      1.80000e+01   1.90000e+01
   3      2.80000e+01   2.90000e+01
```

Here, the number of columns in each block can be controled by a single parameter (e.g., NoColumns = 4). The vector of 30 elements is reshaped into a (3×10) matrix. The output is organized into three blocks: (1) columns 1 through 4, (2) columns 5 through 8, and (3) columns 9 and 10. Appropriate row and column numbers are printed for each block.

Question 4: 10 points. Write a Python program that will print a list of points (x, y) on the graph of the equation

$$y(x) = \left[\frac{x^4 + \left[\frac{x}{sin(x)} \right]}{x - 2} \right] \tag{4}$$

for the range $-4 \leq x \leq 10$ in intervals of 0.25.

Note: You can approach this problem in one of two ways:

1. Detect a numerical problem before it occurs and print out an appropriate message, or
2. Evaluate $y(x)$ for all values of x and then test for various types of error.

You should find that $y(0)$ and $y(2)$ evaluate to not-a-number (NaN) and positive infinity, respectively.

Python implements these values in its standard library. For more info and examples Google: python largest double. You might also try creating a plot of y(x) vs x to see how Python handles these error conditions?

Question 5: 10 points. Rectangles may be defined by the (x,y) coordinates of corner points that are diagonally opposite.

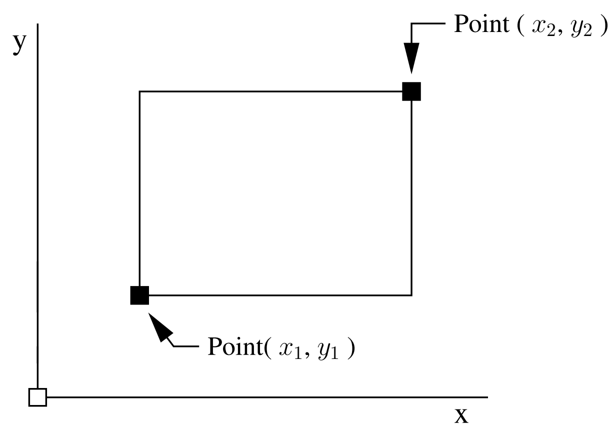

Figure 2: Definition of a rectangle via diagonally opposite corner points.

With this definition in place, develop and test a class `Rectangle.py` that uses `Point.py` to store the (x,y) coordinates of the rectangle corners. Create a handful of test rectangles and test your model by computing their perimeters and areas.

Question 6: 10 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!!

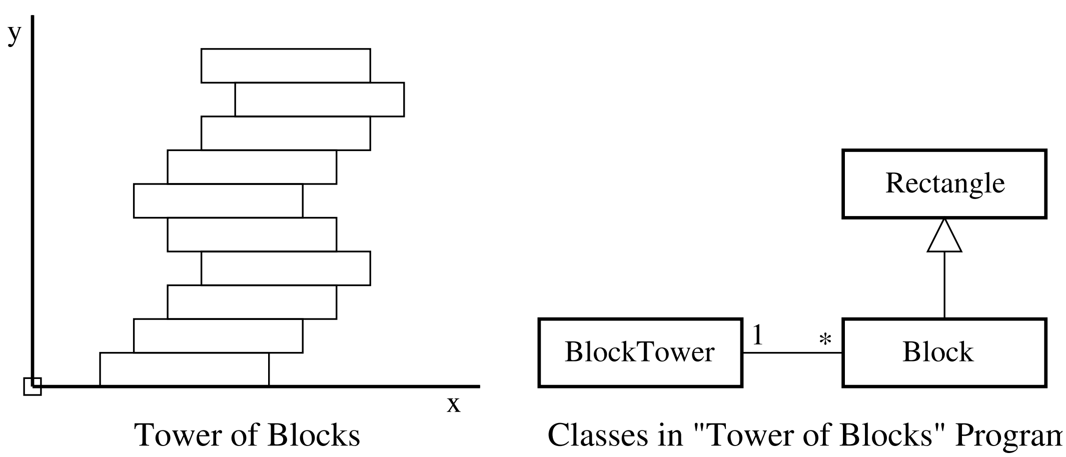

Figure 3: Schematic and classes for "Tower of Blocks"

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, 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 BlockTower 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 density of the block (this will be important in determining its weight) and methods to compute the weight and centroid of each block. The BlockTower class will use block objects to build the tower. A straight forward way of modeling the block tower is with a list of block objects. 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 five units wide and one unit high. 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:

```
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.