Homework 1

(Due: September 19, 2025)

This homework assignment will get you started with programming in Python + Jupyter Notebook. Submit to gradescope a pdf file of the problem descriptions + your solution.

Question 1: 10 points. The four fours puzzle (Google: four fours) is most commonly formulated as follows: Given no more than 4 instances of the digit "4," represent all integers using a finite number of mathematical symbols and operators in common use. Acceptable symbols and operators include:

```
+,-      addition subtraction
*,/      multiplication, division
sqrt,^   square root, power
!        factorial, eg: n! = n * (n-1) * (n-2) * ... * 2 * 1
.        decimal point, e.g., .4
.4'      repeating decimal, eg: .4' = .444444444...
```

There are, however, a few constraints – you cannot use representations for other numbers (e.g., use of π as in $sin(\pi/4) = 1$) or an infinite number of operations (e.g., sqrt(sqrt(sqrt(...4))) = 1).

It can be shown that all integers from 0 through 100 can be represented in this way. Some expressions are far from obvious, for example,

$$\text{the integer 73 corresponds to} \qquad \sqrt{\sqrt{\sqrt{4^{4!}}}} + \frac{4}{\dot{.}4}. \tag{1}$$

Solutions to the first few integers are shown in Table 1. Notice that these expressions are not unique. For example, zero and one can also be expressed as (4-4)/(4+4) and (4+4)/(4+4), respectively.

Write a Python program to evaluate and print the arithmetic expressions for the four-fours problem for integers 0 through 19 and 73. The abbreviated output might look like:

```
--- Value  Expression                    Evaluation ...
--- ================================================ ...
---     0: 4/4 - 4/4                 -->   0.000000 ...
---     1: 44/44                     -->   1.000000 ...
```

Number	Expression	Number	Expression
0	$4/4 - 4/4$	10	$(44 - 4)/4$
1	$44/44$	11	$4/.4 + 4/4$
2	$4/4 + 4/4$	12	$4! - (4 + 4) - 4$
3	$\sqrt{4 * 4} - 4/4$	13	$(4!\sqrt{4} + 4)/4$
4	$4! - (4 * 4) - 4$	14	$4/.4 + \sqrt{4} + \sqrt{4}$
5	$\sqrt{4 * 4} + 4/4$	15	$(4 * 4) - 4/4$
6	$(4 + 4)/4 + 4$	16	$(4 + 4 + 4 + 4)$
7	$4!/(4 + 4) + 4$	17	$(4 * 4) + 4/4$
8	$(4 + 4) * 4/4$	18	$44 * .4 + .4$
9	$(4 + 4) + (4/4)$	19	$4! - 4 - 4/4$

Table 1: Solutions to the four fours problem

```
... lines of output removed ...

--- ============================================== ...
```

Hint: Python has a factorial function ... so you can write `math.factorial(4)` in place of 4!. Also, while the expressions in Table 1 are written in terms of integers, in practice you will need to work with double precision numbers (e.g., to handle 4/.4). You should expect tiny errors because the ensuing arithmetic calculations may not be exact.

Question 2: 10 points. Write a Python program that solves for all positive integer pairs, i.e., x, $y \geq 0$,

$$x + xy + y = 2025. \tag{2}$$

You can solve this problem by simply looping over all potentially good (x,y) values and evaluating equation 2. However, if you factor the left-hand side of equation 2, and then look at the prime factors of the resulting equation, it should be evident there are only a small number of (x,y) pairs that can work. Print your results in a tidy table.

Hint: Python has a package called prettytable (i.e., pip3 install prettytable) which you might find useful. A small test program for pretty tables can be found in: python-code.d/basics/TestPrettyTable01.py.

Question 3: 10 points. Suppose that during squally conditions, regular one second wind gusts produce a forward thrust on a yacht sail corresponding to

$$F(t) = \begin{cases} 10 + 25 \cdot t - 15 \cdot t^3 & 0.0 \leq t \leq 0.4, \\ (900 - 150t)\,/100 & 0.4 < t \leq 1.0 \end{cases} \tag{3}$$

F(t) has units kN.

Write a Python program that computes and prints $F(t)$ for $0 \leq t \leq 4$ seconds, and then creates a line plot of the wind force vs time.

The textual output should look something like:

```
           Time          Thrust
          (sec)            (kN)
==============================
           0.00           10.00
           0.25           16.02
           0.50            8.25
           0.75            7.88
           1.00           10.00

... lines of output removed ...

           3.00           10.00
           3.25           16.02
           3.50            8.25
           3.75            7.88
           4.00           10.00
==============================
```

Hint: Equation 3 is periodic. For values of time, t, greater than one second, you can compute the fractional part of the time with: t - math.floor(t).

Question 4: 10 points. Figure 1 shows a two-dimensional grid of masses. If the total number of point masses is denoted by N, then the total mass of the grid, M, is given by

$$M = \sum_{i=1}^{N} m_i \tag{4}$$

The coordinates of the grid centroid, ($\bar{x}$, $\bar{y}$), are defined by:

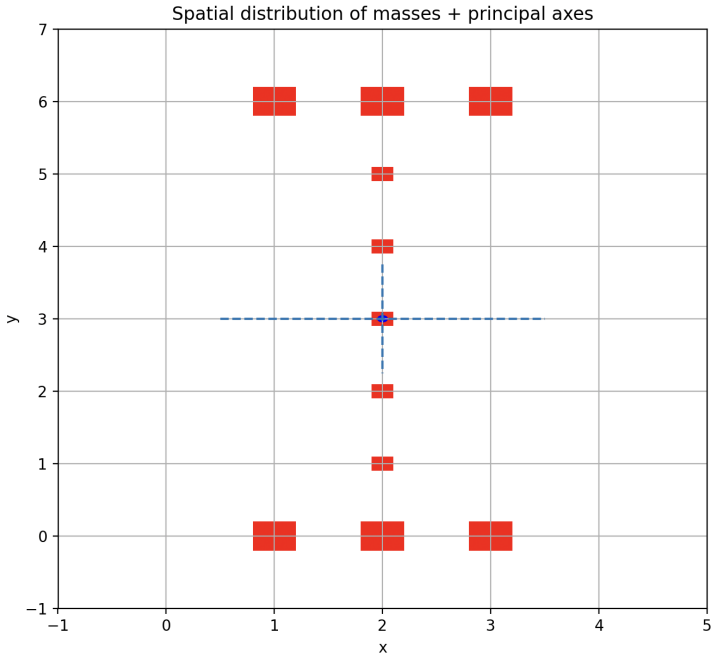

Figure 1: I-shaped grid of masses + principal axes.

$$M\bar{x} = \sum_{i=1}^{N} x_i \cdot m_i \quad \text{and} \quad M\bar{y} = \sum_{i=1}^{N} y_i \cdot m_i \tag{5}$$

The moments of inertia about the x- and y-axes are given by:

$$I_{xx} = \sum_{i=1}^{N} y_i^2 \cdot m_i \quad \text{and} \quad I_{yy} = \sum_{i=1}^{N} x_i^2 \cdot m_i \tag{6}$$

respectively. Similarly the cross moment of inertia is given by

$$I_{xy} = \sum_{i=1}^{N} x_i \cdot y_i \cdot m_i \tag{7}$$

With solutions to equations 5 - 7 in hand, the corresponding moments of inertia about the centroid are given by the parallel axes theorem (Google: parallel axis theorem moments of inertia). Finally, the orientation of the principle axes are given by

$$\tan(2\theta) = \left[\frac{2I_{xy}}{I_{xx} - I_{yy}}\right] \tag{8}$$

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

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

coord = np.array( [ ( 1.0, 0.0 ), ( 2.0, 0.0 ), ( 3.0, 0.0 ),
                    ( 2.0, 1.0 ), ( 2.0, 2.0 ), ( 2.0, 3.0 ),
                    ( 2.0, 4.0 ), ( 2.0, 5.0 ), ( 2.0, 6.0 ),
                    ( 1.0, 6.0 ), ( 3.0, 6.0 ) ] );
```

Write a Python program to evaluate equations 4 – 8, and create a plot in Python similar to Figure 1. Add the centroid and principal axes (drawn with the appropriate orientation) to your plot.

Question 5: 10 points. Figure 2 shows the cross-section of a T-shaped beam (also called T-beam). Reinforced concrete T-beams are commonly found in buildings and highway bridges.

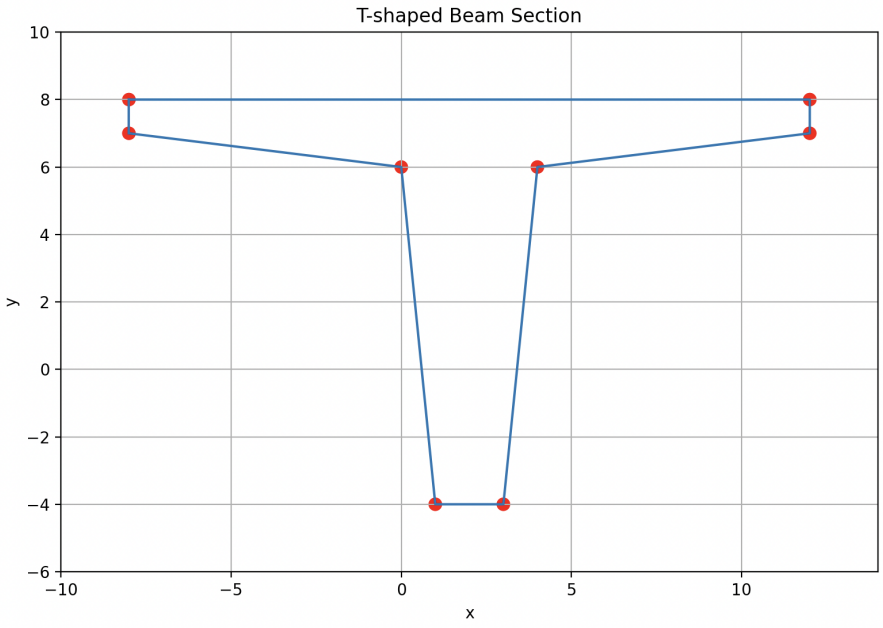

Figure 2: T-shaped beam cross section.

Under service load conditions, T-beams are expected to behave elastically, with very small displacements and no long-term damage. From a mechanics standpoint, the associated elastic analysis procedures require a knowledge of the section area and centroid, and moments of inertia. The purpose of this question is to take a first step toward the development of python code that will compute these section properties automatically.

Later on (i.e., homeworks 2 and 3) we will step things up a bit by adding holes to the cross section, and modeling the whole cross section as an object.

Getting Started. The T-beam shown in Figure 2 has (x, y) coordinates stored as two columns of a numpy array:

```
coord = np.array( [ (  -8.0,  8.0 ),
                    (  12.0,  8.0 ),
                    (  12.0,  7.0 ),
                    (   4.0,  6.0 ),
                    (   3.0, -4.0 ),
                    (   1.0, -4.0 ),
                    (   0.0,  6.0 ),
                    (  -8.0 , 7.0 ) ] );
```

Write a Python program that will:

1. Compute and print the minimum and maximum polygon coordinates in both the x and y directions.
2. Compute and print the minimum and maximum distance of the polygon vertices from the coordinate system origin.
3. Create a plot of the T-beam similar to Figure 2.
4. Write functions perimeter() and area() to compute the perimeter and area of the T-beam, respectively.

Hints. For Parts 1 and 2, use the `max()` and `min()` methods in Python. One way of creating Figure 2 is to draw the vertices as circle objects (i.e., from matplotlib.patches import Circle) and the edges as objects of type Line2D (i.e., from matplotlib.lines import Line2D). To compute the perimeter and area of the T-beam, use the fact that the vertices have been specified in a clockwise manner. You should be able to systematically walk around the perimeter of the T-beam and compute the required values of interest.

Question 6: 10 points. Write a Python program that will compute and print a list of (x, y) pairs for:

$$y(x) = \left[\frac{(x^3 - 25x)}{(x-4)(x+5)\sin(x)} \right] \tag{9}$$

over the range $-10 \leq x \leq 10$ and in intervals of 0.25. You should find that $y(-5)$ and $y(0)$ evaluate to not-a-number (nan), and that $y(4)$ evaluates to positive infinity.

Python 3 provides remarkably good builtin support for handling of run-time errors. Create a plot of $y(x)$ vs x – you should find that errors will be automatically handled within the matplotlib.pyplot environment.