ENCE 201 Engineering Information Processing, Spring Semester, 2024

Homework 1
(Due: February 16, 2024)

The purpose of this homework is to get you started with programming in Python + Jupyter Notebook. Submit to gradescope either: (1) a Jupyter Notebook of your code + program output, plus a pdf file exported from Jupyter, or (2) a zip file of python code for each problem + program output.

Question 1: 10 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 = 2024.

Hint: The prime factorization of 2024 is $2 \cdot 2 \cdot 2 \cdot 11 \cdot 23$. You can use this fact and a little bit of number theory to see that there will only be 3 solutions to the problem, i.e., (a,b) pairs.

Question 2: 10 points. Leibnez's series is given by:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} + \cdots \tag{2}$$

Write a Python program to compute Leibniz's series summation for 1000 terms. First, use a while-loop construct to compute the series summation. Repeat the experiment using an array for the series coefficients, appropriate matrix element-level operations for the alternating signs. and `sum()` for the summation of series terms.

Use the Python package `time` (note, packages `datetime` and `timedelta` may also work) to monitor the time needed to compute each implementation. Conduct a simple experiment to see how the relative speed of the two methods varies as a function of the number of terms in the series?

Question 3: 10 points. The modulo operator, `%`, computes the remainder that occurs after an integer `m` has been divided by a second integer `n`. For example,

```
m = 5, n = 3, 5 = 1*3 + 2  --> 5%3 evaluates to 2
m = 6, n = 3, 6 = 2*3 + 0  --> 6%3 evaluates to 0
m = 7, n = 3, 7 = 2*3 + 1  --> 7%3 evaluates to 1
m = 8, n = 3, 8 = 2*3 + 3  --> 8%3 evaluates to 2
```

and so forth. It is important to notice that `m%n` will always return an integer between 0 and (n-1).

Now let A be a (7×7) matrix whose elements are given by

$$\mathrm{A}[i][j] = \left[i^2 + j^2)\right] \%7. \tag{3}$$

Things to do:

1. Write a short Python program to populate a (7x7) matrix with the results of equation 3.

2. Now let `p` and `q` be integers that cover the interval 0 through 100. Extend your Python program to find combinations of `p` and `q` where $p^2 + q^2$ will be divisible by 7.

3. Prove that if $p^2 + q^2$ is divisible by 7, then it will also be divisible by 49.

Hint. You should use numpy to store the matrix. You can also find a fancy implementation for printing matrices and vectors in the python code distributed in class. The last part of this problem is not as difficult as it looks. Write `p` as $7 * p_1 + r_1$ and `q` as $7 * q_1 + r_2$ and then an expression for $p^2 + q^2$. The result follows directly from the expression and the matrix element values in equation 3.

Question 4: 10 points. Figure 1 is a schematic of an irregular polygon having seven sides.

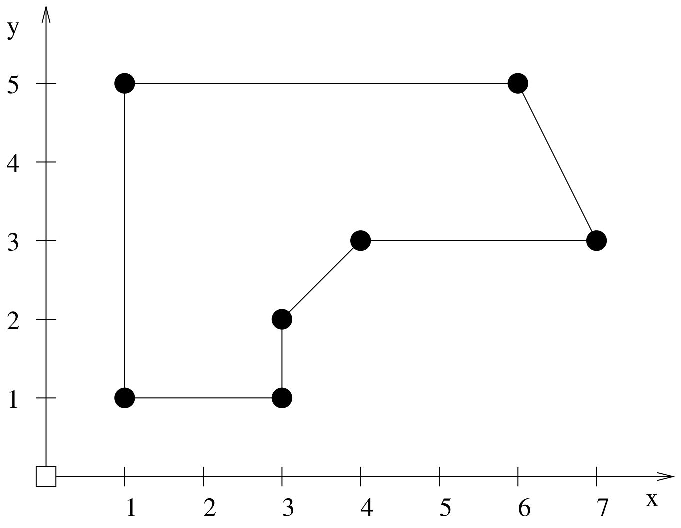

Figure 1: Seven-sided irregular polygon.

Suppose that the x and y vertex coordinates are stored as two columns of information in the array

```
coord = np.array( [ ( 1.0, 1.0 ),
                    ( 1.0, 5.0 ),
                    ( 6.0, 5.0 ),
                    ( 7.0, 3.0 ),
                    ( 4.0, 3.0 ),
                    ( 3.0, 2.0 ),
                    ( 3.0, 1.0 ) ] );
```

Write a Python program that will compute and print

1. The minimum and maximum polygon coordinates in both the x and y directions.
2. The minimum and maximum distance of the polygon vertices from the coordinate system origin.
3. Write functions perimeter() and area() to compute the perimeter and area of the polygon, respectively.

Note. For Parts 1 and 2, use the `max()` and `min()` methods in python. In Part 3, use the fact that the vertices have been specified in a clockwise manner.

Question 5: 10 points. Write a Python program that will compute and print a list of (x, y) pairs for:

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

over the range $-10 \le x \le 10$ and in intervals of 0.25.

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