Assignment #4

Math 1800: Mathematical Programming in Python

Instructions: Choose 3 of the following problems to work on. Submit your responses as Python text files, with the extension `.py`. Each file should include your name and the problem number. Problem set 4 is due Thursday, February 09, by midnight.

- *Problem 4.0:* Our discussion of the Gaussian Prime Spiral included the following pseudocode to trace out a spiral from initial point `cstart` and initial direction `d`. Convert this pseudocode to a working Python program. You can include the `is_gaussian_prime()` code that was given in the notes.

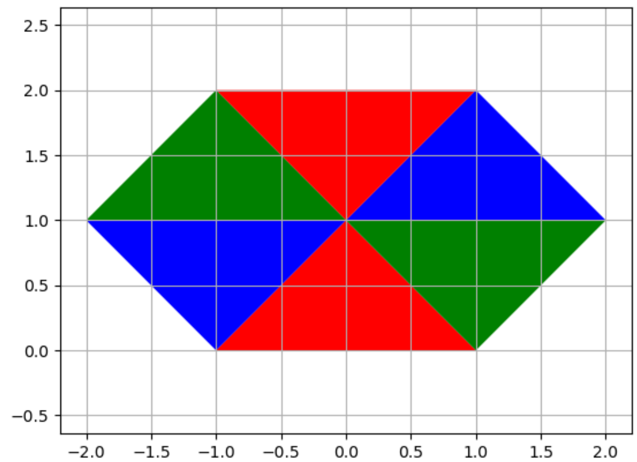
```
gaussian_prime_spiral ( cstart, d )

  step = 0
  loop forever

    if step == 0
      c = cstart
    else
      c = c + d
      if c is gaussian_prime
        d = d * 1j

    alist = alist + real c
    blist = blist + imaginary c

    if ( 0 < step and c == cstart )
      break

    step = step + 1

  plot ( alist, blist )
```

- *Problem 4.1:* Using 1-based indexing, an $n \times n$ matrix A is defined , for indices $1 \leq i, j \leq n$ by

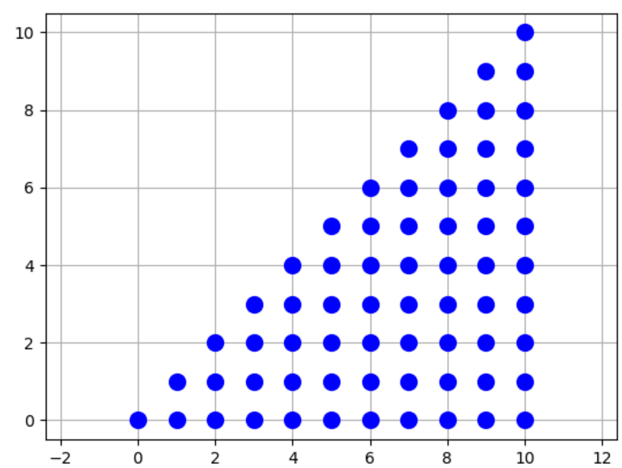
$$A_{i,j} = \sqrt{\frac{2}{n+1}} \sin(\frac{ij\pi}{n+1})$$

  Write a Python function (which uses 0-based indexing!) to define this matrix for any size n. Now use the `numpy` function `C=matmul(A,B)` to verify that `I=A*A`, that is, that A times A returns the identity matrix. (Actually, the resulting matrix will have some small numerical errors.) You can choose $n = 5$ for your example.

- *Problem 4.2:* Use the `matplotlib.pyplot` command `plt.fill()` to create the following plot:



- *Problem 4.3:* Use a pair of `for()` loops and the `matplotlib.pyplot` command `plt.plot()` to create the following grid of blue dots:



- *Problem 4.4:* Plot the "sinc()" function, $f(x) = \frac{\sin(x)}{x}$, over the domain $-10 \leq x \leq 10$. Add a red line across the plot at the level $y = 0.8$, as though we were marking a danger zone.

- *Problem 4.5:* Carry out some operations in complex arithmetic. Recall that a complex number $z = x + yi$ can also be written in polar form as $z = re^{i\theta}$, where $r = \sqrt{x^2 + y^2}$ and $\theta = \tan^{-1}(y/x)$.
  - Let $a = -5 + 4i, b = 2 - 3i$
  - Compute and print $c = a * b$;
  - Use the `r=abs(z)` and `theta=atan2(y,x)` functions to compute the polar forms of a, b, c;
  - Verify that `rc = ra * rb`, and `thetac = thetaa + thetab`;
  - For a, b, c, verify that the polar form equals the rectangular form. That is, evaluate $r * \exp(theta * i)$ for each of the three cases.

- *Problem 4.6:* Over the interval $0 \leq x \leq 1$, consider the function $f(x) = \cos(7.0 * x) + 5 * \cos(11.2 * x) - 2 * \cos(14.0 * x) + 5 * \cos(31.5 * x) + 7 * \cos(63.0 * x)$. Evaluate $y = f(x)$ at 101 evenly spaced points. Use the `np.min()` and `np.max()` functions to determine the minimum and maximum values of y that you observe. Now use the `np.argmin()` and `np.argmax()` functions to find the indexes of y at which these values occur. Finally, use these indexes to report the values of x at which the minimum and maximum occur.

- *Problem 4.7:* Write a one line statement that is:

  - True if vector v is monotonically increasing;
  - True if vector v is *strictly* monotonically increasing.

  Test your two statements on the following vectors:

  - x = [1, 2, 3, 6, 8]
  - y = [-1, 2, 3, 3, 7]
  - z = [1, 3, 7, 2, 9]

- *Problem 4.8:* Hill problem P6.1.2, page 227. The shoelace algorithm for calculating the area of a simple polygon proceeds as follows. Write down the (x, y) coordinates of the n vertices in an $n \times 2$ array, and repeat the coordinates of the first vertex as the last row, to make an $(n + 1) \times 2$ array. Now:

  1. Multiply each x coordinate in the first n rows by the y coordinate in the next row down, and take the sum: $S_1 = x_1 * y_2 + ... + x_n * y_{n+1}$
  2. Multiply each y coordinate in the first n rows by the x coordinate in the next row down, and take the sum, $S_2 = y_1 * x_2 + ... + y_n * x_{n+1}$
  3. The area of the polygon is $0.5 * (S_1 - S_2)$.

  Implement this algorithm, and apply it to compute the area of the polygon whose vertices are:

  ```
  x  y
  0, 0
  3, 0
  3, 3
  2, 3
  2, 1
  1, 1
  1, 2
  0, 2
  ```

  I would suggest you use Python lists for this exercise. You may use Python `for()` loops even though Hill says not to!

- *Problem 4.9:* Carry out the task in problem 4.8, but now do not use `for()` loops. Use `numpy` arrays with index slicing. That means you have to use `np.append()` and `np.sum()` as well.

  If you want to show off, you can write the two summations as a single statement.