```
In [52]: import math
         import numpy as np
         import sympy as sp
         import matplotlib.pyplot as plt
         import sklearn
```

Portfolio Problem 1: Week 1 Advanced Problem 3

My first week's advanced problem had to do with finding polynomials that went through certain points. This code block below defines a matrix of the following meaning: The first four columns are the x values of the points raised to the 0th, 1st, 2nd, and 3rd power respectively. The final column is their respective y values. This is treated as an augmented column. After defining this matrix, we put it into reduced row echelon form and we can see what each of the coefficients needs to be to go through those points.

```
In [ ]: # part A

        m1 = sp.Matrix(
            [[1, -3, 9, -27, -13],
             [1, -1, 1, -1, 2],
             [1, 1, 1, 1, -3],
             [1, 3, 9, 27, -4]
             ])

        sp.pprint(m1.rref())
```

```
⎛⎡1  0  0  0  1/2⎤              ⎞
⎜⎢               ⎥              ⎟
⎜⎢0  1  0  0  -3 ⎥              ⎟
⎜⎢               ⎥, (0, 1, 2, 3)⎟
⎜⎢0  0  1  0  -1 ⎥              ⎟
⎜⎢               ⎥              ⎟
⎝⎣0  0  0  1  1/2⎦              ⎠
```

In this case, we can see that a_0 (or the coefficient of x^0) is $\frac{1}{2}$, a_1 is -3, a_2 is -1, and a_3 is $\frac{1}{2}$ from the augmented component of our matrix printout.

For part B, we do the same proceedure of setting up a matrix with all x values raised to certain powers, but add another column before the y values. Once reduced, this will result in a matrix that represents a system of linear equations with a free variable.

```
In [7]: # part B

        m2 = sp.Matrix(
            [[1, -3, 9, -27, 81, -13],
             [1, -1, 1, -1, 1, 2],
             [1, 1, 1, 1, 1, -3],
             [1, 3, 9, 27, 81, -4]
             ])
```

```
sp.pprint(m2.rref())
```

```
⎛⎡1  0  0  0  -9  1/2⎤              ⎞
⎜⎢                   ⎥              ⎟
⎜⎢0  1  0  0  0   -3 ⎥              ⎟
⎜⎢                   ⎥, (0, 1, 2, 3)⎟
⎜⎢0  0  1  0  10  -1 ⎥              ⎟
⎜⎢                   ⎥              ⎟
⎝⎣0  0  0  1  0   1/2⎦              ⎠
```

Here, we can see that our coefficients will be defined as $a_0 = \frac{1}{2} + 9a_4$, $a_1 = -3$, $a_2 = -1 - 10a_4$, $a_3 = \frac{1}{2}$, and $a_4 \in \mathbb{R}$. There exists a solution for any a_4.

Finally, for part C, we're tasked with writing a function that applies this system generally (although this function does not allow us to find *all* polynomials of a certain degree; rather, it finds the single unique polynomial of minimal degree that passes through the unique points provided.)

```
In [139…
# part C
def findPolynomial(points: list) -> str:
    """
    Finds the polynomial of degree len(points)-1 that passes through
    all points given and outputs a string representation of that
    polynomial
    """

    # unpack points
    xs = list(zip(*points))[0] # get all x values
    ys = list(zip(*points))[1] # get all y values

    # check precondition
    if len(xs) != len(ys):
        raise Exception("Your xs and ys are not the same length!")

    # create rows of matrices
    degree = len(xs)
    matrixList = []
    ys = list(ys[::-1]) # allows popping off a y for every x iteration
    for x in xs:
        row = [x**i for i in range(degree)] # powers of x_m
        matrixList.append(row)
        row.append(ys.pop())

    # create sympy Matrix object and reduce
    echelonMatrix = sp.Matrix(matrixList).rref()

    # get coefficients from reduced matrix
    coefficients = []
    for i in range(degree):
        coefficients.append(echelonMatrix[0].row(i)[-1])

    # create String representation of the polynomial
    stringRep = ""
    exp = degree-1
```