Due date: 9/18/2024, end of the day.

For question 1&2 (programming question), please submit an .ipynb file via Canvas

Programming Problem:

1. **Basic Python [18 points]**
   In this problem, you will need to follow the instructions in given .ipynb file (hw1.1 Basic Operation of Numpy and Matplotlib). The file includes a set of expected outcomes that this question aims to achieve. Your task is to develop your own code which should be capable of producing results similar to the provided expected outcomes.

2. **Polynomial Fitting [42 *points*]**
   In this problem, we write a program to estimate the parameters for an unknown polynomial using the polyfit() function of the numpy package.

   1) Please plot the noisy data and the polynomial you found (in the same figure). You can use any value of *m* selected from 1, 2, 3, 4, 5, 6. After plotting, explain how the choice of *m* influences the fit, considering concepts of overfitting and underfitting.
   2) Plot MSE versus order *m*, for *m* = 1, 2, 3, 4, 5, 6 respectively. Identify the best choice of *m*.
   3) Change variable *noise_scale* to 150, 250, 350, 550, 750, and 950 respectively, re-run the algorithm, and plot the polynomials with the best *m* found in (2). Discuss the impact of increasing noise on the accuracy and reliability of the polynomial parameters. How does the model's performance degrade with increasing noise?
      [You need to plot a figure like in (1) for EACH choice of noise scale.]
   4) Change variable *number_of_samples* to 100, 80, 40, 30, 20, and 10 respectively, re-run the algorithm, and plot the polynomials with the *m* found in (ii). Discuss the impact of the number of samples on the accuracy of the returned parameters.
      [You need to plot a figure like in (i) for EACH choice of number_of_samples.]
   5) Implement regularization techniques (such as ridge regression or lasso) in the polynomial fitting process. Discuss the regularized models with the unregularized ones in terms of MSE. Discuss the benefits and drawbacks of using regularization in this context. When is regularization necessary, and how does it help improve model performance?

   Please use the following code at the beginning of your program to generate the data.

   Simulated data is given as follows in Python:

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
noise_scale = 100
number_of_samples = 50
x =  30*(np.random.rand(number_of_samples, 1) - 0.5)
y = 6 * x + 7 * x**2 + 3 * x**3 + noise_scale*np.random.randn(number_of_samples, 1)
plt.plot(x,y,'ro')
```