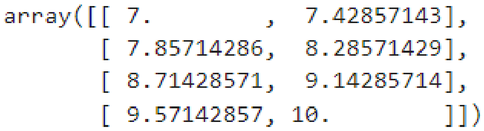
HW2

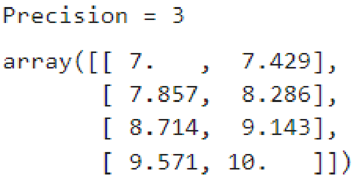
- **Submission date: 02/07/2025 (11:59 PM)**
- Read your class notes *thoroughly* before attempting the assignment.
- Also refer to Sec. 1.3 in the Scientific Python lectures (link)
- You can submit ***well-commented*** code in either `.py` or `.ipynb` formats.
- Submitting a PDF explaining your code is *highly* encouraged.

1. **NumPy practice.**
   a. Create a NumPy array `np_array`1 of integers from 0 to 25 and reshape it into a (5,5) matrix.
   b. Create a NumPy array `np_array2` of size 25 of equally spaced numbers from 5.0 to 10.0 (both endpoints inclusive) and print the array.
   This clearly shows too many digits after the decimal point. Write a function called `matrix_round()` that takes in a NumPy array and a desired precision, and returns another array of the same shape with all entries rounded to the desired precision.
   E.g. If the input is a (4, 2) matrix of the form



```
array([[ 7.        ,  7.42857143],
       [ 7.85714286,  8.28571429],
       [ 8.71428571,  9.14285714],
       [ 9.57142857, 10.        ]])
```

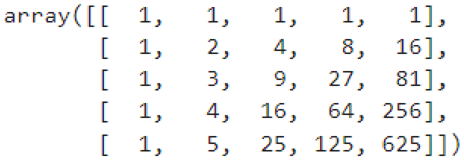
   The output of passing this matrix and a precision of 3 must return (you don't have to print the precision)



```
Precision = 3

array([[ 7.   ,  7.429],
       [ 7.857,  8.286],
       [ 8.714,  9.143],
       [ 9.571, 10.   ]])
```

   Remember, the only two inputs to the function should be the NumPy array and desired precision.
   c. Read the properties of matrix inversion in the class notes. Compute the inverse of the matrix V with the following entries (the `linalg` package might help here)



```
array([[  1,   1,   1,   1,   1],
       [  1,   2,   4,   8,  16],
       [  1,   3,   9,  27,  81],
       [  1,   4,  16,  64, 256],
       [  1,   5,  25, 125, 625]])
```

   Print V^{-1} to 3 decimal places. Compute $V^{-1}V$ and $V\,V^{-1}$ and print them to 3 decimal places. What is the name given to the matrices $V^{-1}V$ and $V\,V^{-1}$? What NumPy command is used to generate such matrices?

2. **Speeding up Matrix Multiplication Using NumPy.**
   a. Write a program to multiply two 700×700 matrices using nested for loops. Using the `tqdm()` function from the `tqdm` package, display a progress bar showing the progress of your program (short tutorial on using the `tqdm` package).
   b. Import the `time` module and use the `time()` method within to measure the time taken to multiply the two matrices (documentation on the time module).
   c. Report the average time taken to multiply two 700×700 matrices using nested for loops (compute the average over 30 randomly generated pairs of matrices).
   d. Repeat the above procedure, but use the NumPy method `numpy.dot()` instead of the two for loops.
   e. What speedup do you observe with NumPy?

3. **MatplotLib.** (Review the MatplotLib tutorial thoroughly before answering. Additionally, refer to Sec. 1.4 in the Scientific Python lectures (link)) solve the problems related to the MatplotLib library in the Jupyter Notebooks named `matplotlib_questions_Part1.ipynb` and `matplotlib_questions_Part2.ipynb`.