**CGN 3405**: Applied Numerical Methods for Civil Engineering

**Topic**: Introduction to Python Programming

**Note:** Please use Colab to write your Python code. When you are ready to submit, please check out the TA's Announcement (see Webcourses) in terms of downloading "`.ipynb`." TA will run your Python codes and check the results.

## Q1: NumPy Array Creation and Operations (15 points)

1. **(4 points)** Create the following arrays using NumPy:
   - A $3 \times 4$ matrix of ones
   - A $5 \times 5$ identity matrix
   - A vector from 0 to 10 with step size 0.5 using `np.arange()`
   - A vector of 21 equally spaced points between 0 and 1 using `np.linspace()`
2. **(5 points)** Perform element-wise multiplication using NumPy:
   - Compute the element-wise multiplication between

$$\boldsymbol{x} = (1, 2, 3, 4, 5, 6, 7)^\top \qquad \boldsymbol{y} = (11, 12, 13, 14, 15, 16, 17)^\top \tag{1}$$

     Please create arrays using `np.arange()`.
   - Compute the element-wise multiplication between

$$\boldsymbol{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \qquad \boldsymbol{Y} = \begin{bmatrix} 11 & 12 & 13 \\ 14 & 15 & 16 \end{bmatrix} \tag{2}$$

3. **(6 points)** Perform matrix computations using NumPy:
   - Compute the matrix-vector multiplication between

$$\boldsymbol{K} = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 4 & 3 \\ 1 & 3 & 5 \end{bmatrix} \qquad \boldsymbol{u} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \end{bmatrix} \tag{3}$$

   - Compute the matrix-matrix multiplication between

$$\boldsymbol{K}_1 = \begin{bmatrix} 100 & -100 \\ -100 & 100 \end{bmatrix} \qquad \boldsymbol{K}_2 = \begin{bmatrix} 50 & -50 \\ -50 & 50 \end{bmatrix} \tag{4}$$

**Requirements:**

- Use `import numpy as np`.
- Write clear, commented code.
- Print the results.

## Q2: Approximation for $\cos(x)$ (20 points)

Recall the **Taylor series** expansion for the cosine function around $x = 0$:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots \tag{5}$$

The general form of the $n$-term Taylor series approximation is:

$$\cos(x) \approx \sum_{k=0}^{n-1} (-1)^k \frac{x^{2k}}{(2k)!} \tag{6}$$

1. **(8 points)** Write a Python function called `cos_taylor(x, n)` that:
   - Takes two arguments: `x` (the value at which to evaluate the cosine) and `n` (the number of terms in the Taylor series).
   - Returns the approximate value of $\cos(x)$ using the first $n$ terms of the Taylor series.
   - Use NumPy for mathematical operations.
2. **(12 points)** Test your function with $x = 1.2$ for $n = 1$ through $n = 6$ terms. Compute and display:
   - The approximate value from your function for each $n$.
   - The exact value from `np.cos(1.2)`.
   - The absolute error for each $n$:

$$\text{error} = |\text{exact} - \text{approximation}| \tag{7}$$

   - Discuss your observations about convergence and accuracy in a comment block at the end.

**Requirements:**

- Use `import numpy as np`.
- Write clear, commented code.
- Print the results.

## Q3: Computing Norms (15 points)

**(5 points for each norm)**

1. Generate a random vector of length `100` using `numpy.random.rand()` with a fixed random seed of `3`.
2. Compute the following vector norms for this random vector:
   - $\ell_1$-norm: sum of absolute values.
   - $\ell_2$-norm: square root of sum of squares.
   - $\ell_\infty$-norm: maximum absolute value.
3. Use two methods for each norm:
   - **Method A**: Define a `lambda` function for each norm.
   - **Method B**: Use the built-in `numpy.linalg.norm()` function with the appropriate parameter.
4. Print the results for both methods, showing that they match.

**Requirement:**

- Use `import numpy as np`.
- Write clean, commented code.
- For reproducibility, you need to set the random seed as `3`.
- Ensure your `lambda` functions correctly implement the mathematical definitions of each norm.
- Print the results.