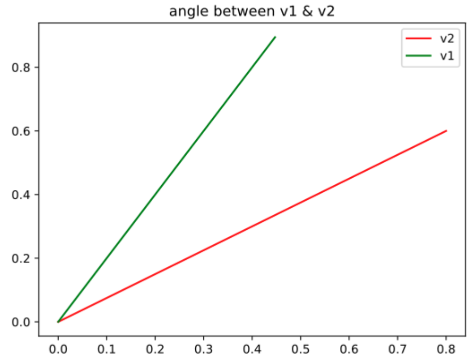
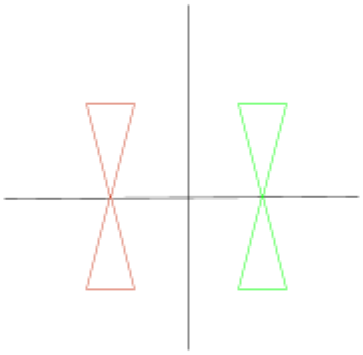
Assignment-9 for Numpy and Plotting

Subject: CSW1 (CSE 2141)
Session: Sep 2025 to Jan 2026
Branch: Computer Science and Engineering (CSE)
Section: All
Course Outcome: CO6
Program Outcomes: PO1, PO2, and PO5
Learning Levels: Remembering (L1), Understanding (L2), Application (L3), Analysis (L4)
Evaluation (L5), and Creation (L6)

Q no.	Questions	Learning Levels
Q1.	Initialize two integer 2-D vectors v1 and v2 randomly using numpy. Calculate and print the following- a. Length of the vectors. b. Normalized (i.e., unit length) $v1_{Norm}$ and $v2_{Norm}$. c. Angle between the vectors in degree and radian. Example- v1=[1, 2], v2=[4, 3]. $L(v1)=\sqrt{5}$, $L(v2)=5$; $v1_{Norm}$ = [$1/\sqrt{5}$, $2/\sqrt{5}$] $v2_{Norm}$=[4/5,3/5] If angle is θ then $\cos\theta$ = ([$1/\sqrt{5}$, $2/\sqrt{5}$] **dot_product** [4/5, 3/5])	L2, L3
Q2.	Use **numpy.random package** to generate a 2D array of size **1000 × 1000** filled with random floating-point numbers in the range **[0, 1)**. Measure and display the execution time required for this operation. Next, generate an equivalent 2D array of size **1000 × 1000** manually using Python's built-in **random.random()** function with **nested for loops**. Measure the execution time for this approach as well. Finally, compare the two execution times and clearly print which method (**NumPy** or **Python nested loops**) is faster.	L2, L3

Q3.	Generate **24 random integers** between 1 and 20 using NumPy, and reshape them into a matrix **A** of size 6×4.Using **numpy array slicing**, take the **first two rows** of matrix A to form matrix **B** of size 2×4, and the **last four rows** of matrix A to form matrix **C** of size 4×4. Perform matrix multiplication of B × C to produce the resulting matrix **D** of size 2×4. Print all the matrices **A**, **B**, **C**, and **D**.	L2, L3
Q4.	Given a 2D integer matrix of size m×n, perform the following tasks using numpy: a. Create a function **matrix_stats(matrix)** to find and return the matrix statistics including **min(), max(), mean(), median(), size(), std(),** and **sum()**. b. Create another function **separate_even_odd(matrix)** that generates two matrices of the same shape as the original: an **EvenMatrix** where all odd numbers are replaced with zero, and an **OddMatrix** where all even numbers are replaced with zero. Call these functions in your program to print the original matrix, display the statistical results with labels, and the two matrices.	L2, L3
Q5.	Generate a NumPy 1D array of random integers between 0 to 99. Then using a Boolean array, mask out all integers except those meeting any of the three conditions: a. If number n = 1 b. If number n is a multiple of 3 c. If number n > 75 After masking all conditions, print the array.	

Q6.	Plot the normalized vectors of Q1 from the origin. Color v1 green and v2 red. Add legend.  	L2, L3
Q7.	Draw the following figure using matplotlib. 	L2, L3
Q8.	Visualize the sine and cosine functions using NumPy and Matplotlib. Generate the x values from 0 to 2π with an interval of 0.1, and compute their corresponding sine and cosine values. Plot both curves on the same graph, where the **sine** function should appear in **blue with a solid line** and the **cosine** function in **red with a dashed line**. Label the x-axis as “X values (radians)” and the y-axis as “Function values”, add the title “Sine and Cosine Functions”, and include a legend to clearly distinguish between the two curves.	L2, L3

Q9.	Create arrays for time periods from 0 to 30 years in monthly increments. Calculate and plot the future value of an initial investment of $10,000 under three different compounding frequencies: annual (rate 5%), quarterly, monthly. Display each scenario separately, then overlay them in a plot for comparison. Use proper colors & legends.	L2, L3
Q10.	Creates a 3D mesh grid using NumPy. Defines a mathematical function z=sin($\sqrt{x^{2} + y^{2}}$ to visualize. Plots the surface using plot_surface() from mpl_toolkits.mplot3d. Uses a color map (viridis) for visual appeal.	L2, L3
Q11.	Generate a random 4x4 matrix. Compute its determinant. If the determinant is non-zero, compute the inverse and verify by multiplying M with its inverse to check if it approximates the identity matrix (with tolerance 1*e-6). If singular, adjust one element to make it invertible and recompute. Use Matplotlib to create heatmaps side-by-side: one for the original matrix M (using plt.imshow(M, cmap='Blues')) and one for the inverse matrix (if invertible) or the adjusted matrix. Add colorbars, label the subplots, and include a title like "Matrix and Its Inverse Visualization" with a text annotation showing the determinant value.	L2, L3
Q12.	Define two square matrices. Compute their Kronecker product using np.kron. Flatten the resulting matrix into a 1D array and create a histogram of its values using with 10 bins & 50 bins. Customize the histogram with edge colors, add a title, and label the axes to show the distribution of Kronecker product elements. Compute statistical measures like the mean, variance, and skewness of the flattened array , and overlay a normal distribution curve on the histogram for comparison.	L2, L3

Q13.	Make a financial analysis of credit card transactions done by Indians. [The file name is “*Credit_card_spending.txt*.” The column names are **‘index’, ’City’, ’Date’, ‘Card Type’, ’Exp Type’, ’Gender’, ’Amount’]** a. Count the number of transactions for both genders and create a bar plot. b. The system displays the number of cardholders for each type of card and plots this information in a bar graph. c. Display the top 3 cities with the highest transaction by plot. d. Draw the year-wise amount spent on the cardholder type **‘signature.’** e. Show the total amount spent per cardholder type in a pie chart.	L2, L3
Q14.	Design and analyze the 40 students' mark database. Each student has five subjects like **Math-I, English, Chemistry, OB, and ICP**. Subjects' marks are generated by randint () method, and the database is made. Then Analysis a. Identify the names of the top two subjects that achieved the highest marks. b. Compare and analyze each subject's mark using a distribution plot. c. Find out the maximum and minimum number of students who pass and fail in each subject. And visualize it. d. Analyze and plot the average pass and fail in each subject. e. Display the students' marks from highest to lowest.	L3, L4
Q 15.	Create a function that generates a 2x2 matrix with elements based on a parameter t, such as [[t, 1], [2, t+1]], where t ranges from -10 to 10 using np.linspace. Compute the determinant for each t using np.linalg.det. Plot the determinants against t as a line graph using Matplotlib, marking points where the determinant is zero with red dots. Include grid lines, axis labels, and a legend.	L3, L4
	-END-	