

**Assignment-8 The “numpy” (Numeric Python) Package**

**Subject: Computer Science Workshop - 1 (CSE 2141)**

**Session: Sep 2025 to Jan 2026**

**Branch: Computer Science and Engineering (CSE)**

**Section: All**

**Course Outcome: CO5, CO6**

**Program Outcomes: PO1, PO2, PO3, and PO5**

**Learning Levels: Remembering (L1), Understanding (L2), Application (L3), Analysis (L4)**

<b>Q no.</b>	<b>Questions</b>	<b>Learning Levels</b>
Q1.	<p>Initialize two integer 1-D vectors v1 randomly using numpy. Calculate and print the following-</p> <ol style="list-style-type: none"><li>Find the maxi and mini value</li><li>Find the number of elements present in between max and min</li></ol> <p>Note: Consider the duplicates in the array</p>	L1, L2
Q2.	<p>Initialize two integer 2-D vectors v1 and v2 randomly using numpy. Calculate and print the following-</p> <ol style="list-style-type: none"><li>Length of the vectors.</li><li>Normalized (i.e., unit length) v1Norm and v2Norm.</li><li>Angle between the vectors in degree and radian.</li></ol>	L1, L2
Q3.	<p>Use numpy.random package to generate a 2D array of size <math>1000 \times 1000</math> 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 <math>1000 \times 1000</math> 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.</p>	L2, L3
Q4.	<p>A weather monitoring system records daily temperatures (in °C) for a city over 14 days. Perform following task using the concept “Boolean masking in Numpy” :</p> <ul style="list-style-type: none"><li>Identify heat-wave days where temperature is <math>\geq 35^{\circ}\text{C}</math></li><li>Count the number of heat-wave days</li><li>Replace temperatures below <math>30^{\circ}\text{C}</math> with the value 30</li><li>Extract only comfortable days (<math>30^{\circ}\text{C}</math> to <math>34^{\circ}\text{C}</math> inclusive)</li><li>Increase all heat-wave day temperatures by <math>2^{\circ}\text{C}</math> (data correction)</li></ul>	L2, L3

Q5.	<p>A hospital stores fasting blood sugar levels (mg/dL) of multiple patients in a NumPy array. You are asked to do the following task:</p> <ol style="list-style-type: none"> <li>1. Create boolean masks using NumPy to identify patients belonging to different risk categories.</li> <li>2. Classify each patient into one of the following medical conditions:           <ol style="list-style-type: none"> <li>1. Normal → blood sugar <math>&lt; 100</math></li> <li>2. Pre-Diabetic → blood sugar between 100 and 139 (inclusive)</li> <li>3. Diabetic → blood sugar <math>\geq 140</math></li> </ol> </li> <li>3. Store the classification result in a separate NumPy array of the same size.</li> </ol> <p>Do not use loops (for / while). Use boolean masking only.</p>	L2, L4
Q6.	<p>Design a method in python to generate a NumPy array of <b>1000 observations</b> and calculate its <b>mean and standard deviation</b>.</p> <p>Design another method to analyze the distribution by counting the number of values that lie within <b>one, two, and three standard deviation intervals</b> around the mean, which will be stored in a dictionary. Print the dictionary outside of the method.</p>	L3, L4
Q7.	<p>Develop a matrix-based guessing game in which the matrix size is specified by the user. For each row of the matrix, the system randomly chooses a hidden number. The user repeatedly inputs guesses for each row until the correct number is guessed. Determine and report the number of trials taken for each row to achieve a correct guess.</p>	L2, L3
Q8.	<p>Design a method to generate a <b><math>4 \times 5</math> matrix</b> such that the elements of each row are randomly generated within a specific range defined by the row index. For the row at index <math>i</math> (0-based indexing), the values should be generated in the range <b>(<math>i + 1</math>) to (<math>i + 9</math>)</b>.</p> <p>After constructing the matrix:</p> <ol style="list-style-type: none"> <li>1. Compute the <b>mean and standard deviation row-wise and column-wise</b>.</li> <li>2. Compare the row-wise and column-wise results separately for mean and standard deviation.</li> <li>3. Determine which dimension (<b>row-wise or column-wise</b>) has the <b>larger overall value</b> for mean and for standard deviation.</li> <li>4. Return the final outcome as a <b>dictionary</b> with the keys:           <ol style="list-style-type: none"> <li>i) "mean" → indicating whether row-wise or column-wise mean is dominant</li> <li>ii) "std" → indicating whether row-wise or column-wise standard deviation is dominant</li> </ol> </li> </ol>	L2, L4

Q9.	Using NumPy, generate a three-dimensional array and normalize the data independently along each dimension. Return the resulting array after applying the normalization.	L3, L4
Q10.	Use numpy.random package to generate a 2D array of size $1000 \times 1000$ 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 $1000 \times 1000$ 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.	L3, L4
	<b>-END-</b>	