**Q1. If you have any, what are your choices for increasing the comparison between different figures on the same graph?**

Ans- To increase the comparison between different figures on the same graph, you have several options:

* Adjust the scaling of the y-axis: You can modify the range of values displayed on the y-axis to emphasize the differences between the figures.
* Use different colours or patterns: Assign distinct colours or patterns to each figure to make them visually distinguishable.
* Utilize markers or data points: If you are plotting line graphs, adding markers or data points can enhance the visibility of individual data points and facilitate comparison.
* Add annotations or labels: Labelling the figures directly on the graph can provide additional context and aid in comparison.

**Q2. Can you explain the benefit of compound interest over a higher rate of interest that does not compound after reading this chapter?**

Ans- Compound interest offers the benefit of exponential growth over time compared to a higher rate of interest that does not compound. With compound interest, the interest earned is added back to the initial investment or principal, and subsequent interest calculations are based on the increased principal. This compounding effect leads to the growth of the investment at an accelerating rate. In contrast, a higher rate of interest that does not compound would result in linear growth, where the interest earned remains constant over time, leading to slower overall growth.

**Q3. What is a histogram, exactly? Name a numpy method for creating such a graph.**

Ans- A histogram is a graphical representation of the distribution of a dataset. It consists of a series of bars, where each bar represents a specific range or bin of values, and the height of the bar corresponds to the frequency or count of values falling within that range.

A numpy method for creating a histogram is numpy.histogram(). It takes an input array and the number of bins as arguments and returns an array of histogram values and an array of bin edges.

**Q4. If necessary, how do you change the aspect ratios between the X and Y axes?**

Ans- To change the aspect ratios between the X and Y axes, you can use the matplotlib.pyplot.axes().set\_aspect() function. This function allows you to set the aspect ratio explicitly by providing a value or use one of the preset options such as "equal", "auto", or "box". By adjusting the aspect ratio, you can modify the scaling of the plot along the X and Y axes.

**Q5. Compare and contrast the three types of array multiplication between two numpy arrays: dot product, outer product, and regular multiplication of two numpy arrays.**

Ans- The three types of array multiplication between two numpy arrays are:

* Dot product: The dot product is performed using the numpy.dot() function or the @ operator. It calculates the sum of the element-wise products of the corresponding elements in the arrays. The result is a scalar value or a 1-dimensional array, depending on the dimensions of the input arrays.
* Outer product: The outer product is calculated using the numpy.outer() function. It generates a new array by computing the product of each element in the first array with each element in the second array. The result is a new array with dimensions equal to the product of the dimensions of the input arrays.
* Regular multiplication: Regular multiplication between two numpy arrays is performed using the \* operator. It performs element-wise multiplication, where each element in one array is multiplied by the corresponding element in the other array. The result is a new array with the same shape as the input arrays.

**Q6. Before you buy a home, which numpy function will you use to measure your monthly mortgage payment?**

Ans- To measure the monthly mortgage payment before buying a home, you would typically use the numpy function numpy.pmt(). This function calculates the periodic payment required to pay off a loan or mortgage with a fixed interest rate and a fixed number of payments. By providing the loan amount, interest rate, and number of payments, the numpy.pmt() function returns the monthly mortgage payment.

**Q7. Can string data be stored in numpy arrays? If so, list at least one restriction that applies to this data.**

Ans- Yes, string data can be stored in numpy arrays. However, there are some restrictions like,

* The strings must have fixed length: In numpy arrays, strings are stored as fixed-length arrays of characters. Therefore, the strings must have a predetermined and fixed length. Any strings that exceed this fixed length will be truncated.
* Strings in numpy arrays are less efficient than numeric types: Operations on string arrays are generally slower and