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

***Ans***:

Here are some common techniques for increasing the comparison between different figures on the same graph:

1. Use consistent scales on the axes: By using the same scale on the axes, it becomes easier to compare the shapes and patterns of the different figures. In addition, make sure the axis labels and tick marks are clear and easy to read.

2. Use color: Color can be a powerful tool for distinguishing between different figures on a graph. You can use different colors to represent different figures or data sets.

3. Use different line styles: Different line styles (e.g., solid, dashed, dotted) can help distinguish between different figures on a graph.

4. Use annotations: Annotations (e.g., labels, text boxes) can help draw attention to specific parts of the graph and help the viewer understand the data being presented.

5. Use small multiples: Small multiples are a series of small graphs, each showing a different subset of the data. By breaking the data into smaller chunks, it becomes easier to compare and contrast different figures.

6. Use data transformations: Data transformations (e.g., logarithmic scaling) can help highlight differences in the data that might not be apparent on a linear scale.

7. Use a legend: A legend can help identify different figures on the graph, particularly when using color or different line styles.

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 is the addition of interest to the principal sum, which is the original amount of money invested or borrowed. The interest earned in each period is added to the principal, and the interest in the next period is calculated on the new total amount. This compounding effect leads to the growth of the investment or loan at an accelerating rate. In contrast, a higher rate of interest that does not compound will not benefit from the compounding effect and will result in a lower overall return.

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

***Ans*** :

A histogram is a graph that represents the distribution of numerical data by dividing it into intervals (or bins) and counting the number of data points that fall into each bin. The resulting graph consists of a series of rectangles, where the height of each rectangle represents the frequency or count of data points that fall into that bin.

NumPy provides a method called histogram that can be used to create histograms. This method takes an array of data points and returns two arrays: the first array contains the bin edges (i.e., the boundaries between the bins), and the second array contains the counts for each bin. These two arrays can then be plotted using a visualization library like Matplotlib to create a histogram graph.

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 plt.gca() method to get the current axes object and then call its set\_aspect() method with the desired aspect ratio value.

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***: There are three types of array multiplication between two NumPy arrays: dot product, outer product, and regular element-wise multiplication.

1. Dot product: The dot product of two arrays is the sum of the element-wise product of their corresponding entries. The result is a scalar value, and the operation requires that the two arrays have compatible shapes. The dot product is calculated using the dot method in NumPy.

2.Outer product: The outer product of two arrays is a new array that contains the product of each element in the first array with each element in the second array. The result is a new array with a shape equal to the product of the shapes of the two input arrays. The outer product is calculated using the outer function in NumPy.

3. Regular multiplication: Regular multiplication between two NumPy arrays multiplies the corresponding elements in each array. The two arrays must have the same shape. The result is a new array with the same shape as the input arrays. This operation is performed using the \* operator in NumPy.

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

***Ans***:

you can use NumPy's financial functions, such as np.pmt(), to calculate the periodic payment for a loan.

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 using the dtype parameter set to 'S' or 'U' (for Unicode strings). One restriction that applies to string data is that the strings must have fixed length, which needs to be specified when creating the array.