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

There are several ways to increase the comparison between different figures on the same graph:

1. Change the y-axis limits: By changing the y-axis limits, you can make the differences between the figures more pronounced.
2. Use different colors: You can use different colors to make each figure stand out more. However, it's important to use colors that are easy to distinguish from one another.
3. Use different line styles: You can use different line styles, such as solid lines and dashed lines, to make each figure more distinct.
4. Use markers: You can use markers, such as dots or squares, to make each data point more visible.
5. Add a legend: By adding a legend, you can label each figure and make it easier for the viewer to understand the differences between the figures.

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

Compound interest is the interest that is added to the principal of an investment over time, and then the interest is also calculated on the new total amount. This means that over time, the interest earned on the investment grows exponentially. In contrast, a higher rate of interest that does not compound does not earn interest on the interest that has already been earned, so the growth is not exponential.

The benefit of compound interest is that it can lead to significant growth in an investment over time. The effect of compounding becomes more pronounced the longer the investment is held, so the earlier one starts investing, the greater the benefit of compounding will be.

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

A histogram is a graph that represents the distribution of a set of data. It shows how many times each value in the data set appears, or how many values fall into each range of values. The x-axis shows the range of values and the y-axis shows the frequency of values falling within each range.

In NumPy, the **histogram** method can be used to create a histogram graph. This method takes an array of values and returns two arrays: the frequency of the values and the range of values.

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

To change the aspect ratios between the x and y axes, you can use the **aspect** parameter in Matplotlib. The **aspect** parameter takes a value that represents the aspect ratio of the plot, which is the ratio of the height to the width of the plot.

For example, to make the plot wider, you can increase the aspect ratio to make the height smaller relative to the width:

import matplotlib.pyplot as plt

# create a plot

fig, ax = plt.subplots()

# set the aspect ratio to 2:1

ax.set\_aspect(2)

# plot the data

ax.plot(x, y)

# show the plot

plt.show()

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.

There are three types of array multiplication between two numpy arrays: dot product, outer product, and regular multiplication. Here are the differences:

1. Regular multiplication: This type of multiplication between two numpy arrays is the simplest form of multiplication. When we multiply two numpy arrays, the corresponding elements are multiplied together to form a new array of the same shape. This is known as element-wise multiplication.
2. Dot product: This type of multiplication is used to multiply two arrays of the same shape. It is the most common type of multiplication used in linear algebra. In dot product multiplication, the corresponding elements of the two arrays are multiplied together and then added up to form a single value. The resulting array has a shape of (m, n) where m is the number of rows in the first array and n is the number of columns in the second array.
3. Outer product: This type of multiplication is used to find the outer product of two arrays. The outer product of two arrays is a matrix that contains the product of each element of the first array with each element of the second array. The resulting array has a shape of (m, n) where m is the number of elements in the first array and n is the number of elements in the second array.

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

To calculate the monthly mortgage payment, we can use the numpy function called "pmt". The pmt function calculates the payment on a loan based on constant payments and a constant interest rate.

The syntax of the pmt function is as follows:

numpy.pmt(rate, nper, pv, fv=0, when='end')

Here, rate is the interest rate, nper is the total number of payments, pv is the present value or the total amount of the loan, fv is the future value, and when is the timing of the payments.

For example, to calculate the monthly mortgage payment for a loan of $200,000 with an interest rate of 4% for 30 years, the numpy pmt function can be used as follows:

import numpy as np

rate = 0.04/12

nper = 30\*12

pv = -200000

pmt = np.pmt(rate, nper, pv)

print(pmt)

This will output 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.

Yes, string data can be stored in numpy arrays. However, there are some restrictions that apply to this data:

1. All the elements in the array must have the same length.
2. The array must have a fixed length.
3. The maximum length of the string is determined by the size of the array.

For example, if we want to store string data in a numpy array, we can create a numpy array of type 'S' (for string) with a fixed length as follows:

import numpy as np

data = np.array(['John', 'Mary', 'Tom'], dtype='S4')

Here, we have created a numpy array of type 'S4', which means that each string in the array has a length of 4 characters.