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

Q2. Can you explain the benefit of compound interest over a higher rate of interest that does not

compound after reading this chapter?

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

Q4. If necessary, how do you change the aspect ratios between 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.

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

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

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

To increase the comparison between different figures on the same graph, you can use the following techniques:

1. **Use of multiple subplots:** Plot different figures side by side using subplots (plt.subplot() in Matplotlib) to allow easy comparison.
2. **Overlaying plots:** Overlay multiple datasets on the same graph with different colors, markers, or line styles to distinguish them.
3. **Annotations:** Add labels, legends, and annotations to clearly identify and explain different figures.
4. **Normalization:** Normalize data to the same scale to make comparisons more meaningful.
5. **Dual axes:** Use dual axes (plt.twinx() in Matplotlib) to compare datasets with different ranges on the same plot.
6. **Heatmaps:** Use a heatmap to visually compare the intensity of different figures across a grid.
7. **Interactivity:** Use interactive plotting tools (like Plotly) that allow users to hover over, zoom in, or filter data points for better comparison.

### **Q2. Can you explain the benefit of compound interest over a higher rate of interest that does not compound?**

Compound interest has the advantage of earning interest on both the initial principal and the accumulated interest from previous periods. This results in exponential growth of the investment over time, as opposed to simple interest, which only earns interest on the principal. Even if the rate of simple interest is higher, compound interest can eventually surpass it, especially over a long period, due to the compounding effect.

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

A histogram is a graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable and is made up of bins, each representing the frequency (count) of data points that fall within a certain range of values.

The NumPy method for creating a histogram is numpy.histogram(). However, to display the histogram graphically, you typically use matplotlib.pyplot.hist().

### **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 in a Matplotlib plot, you can use the set\_aspect() method of the Axes object. For example:

python

import matplotlib.pyplot as plt

ax = plt.gca() # Get current axis

ax.set\_aspect('equal', adjustable='box') # Set aspect ratio to be equal

This will make the scales on both axes equal, or you can pass a specific numerical value to set\_aspect() to set a custom ratio.

### **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.**

* **Dot Product (numpy.dot() or @ operator):**
  + **Definition:** Computes the dot product of two arrays. For 1D arrays, it results in a scalar value, and for 2D arrays (matrices), it results in matrix multiplication.
  + **Use Case:** Commonly used in linear algebra for operations like matrix multiplication and finding projections.

**Example:**python  
  
import numpy as np

a = np.array([1, 2])

b = np.array([3, 4])

np.dot(a, b) # Output: 11 (1\*3 + 2\*4)

* **Outer Product (numpy.outer()):**
  + **Definition:** Computes the outer product of two vectors, resulting in a matrix where each element is the product of the corresponding elements of the input vectors.
  + **Use Case:** Used in various mathematical applications, including tensor products and in generating matrices from two vectors.

**Example:**python  
  
np.outer(a, b) # Output: array([[3, 4], [6, 8]])

* **Regular Multiplication (\* operator or numpy.multiply()):**
  + **Definition:** Performs element-wise multiplication of two arrays of the same shape.
  + **Use Case:** Used in element-wise operations where each element of one array is multiplied by the corresponding element of the other array.

**Example:**python  
  
np.multiply(a, b) # Output: array([3, 8])

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

To calculate your monthly mortgage payment, you can use the numpy.pmt() function, which is used to compute the payment against loan principal plus interest.

Example:

python

import numpy as np

rate = 0.05 / 12 # Monthly interest rate (annual rate divided by 12)

nper = 30 \* 12 # Total number of payments (30 years \* 12 months)

pv = 300000 # Present value (loan principal)

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

print(payment) # Output: -1610.46 (monthly payment)

The result is 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:

* **Fixed Length:** The length of strings in a NumPy array is fixed once the array is created. If you assign a longer string to an element, it will be truncated to the pre-defined length.
* **Uniform Type:** All elements in a NumPy array must have the same data type and size. This means that if one string is shorter than the defined length, it will be padded with null characters (\x00) or spaces.

Example:

python

import numpy as np

arr = np.array(['apple', 'banana', 'cherry'], dtype='U6') # Array of strings with max length 6

print(arr) # Output: ['apple' 'banana' 'cherry']

If you try to assign a longer string:

python

arr[0] = 'pineapple'

print(arr) # Output: ['pineap' 'banana' 'cherry'] (truncated)