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.

Answers

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

To enhance comparisons between different figures on the same graph, you can use several techniques:

1. \*\*Dual Axis Charts\*\*: Utilize dual-axis line or bar charts to compare two datasets with different scales on the same graph.

2. \*\*Color Coding\*\*: Use distinct colors for different datasets to make them easily distinguishable.

3. \*\*Legends and Labels\*\*: Include clear legends and axis labels to indicate what each line or bar represents.

4. \*\*Overlaying Graphs\*\*: Overlay multiple graphs (e.g., line and scatter plots) on the same axes to visualize relationships between datasets.

5. \*\*Transparency\*\*: Adjust the transparency of the plots to allow overlapping data to be seen clearly.

These techniques help in effectively visualizing and comparing multiple datasets on a single graph.

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

The benefit of \*\*compound interest\*\* is that it allows interest to be calculated on both the initial principal and the accumulated interest from previous periods. This results in exponential growth of the investment over time. In contrast, a higher rate of interest that does not compound only applies to the principal amount, leading to linear growth. Over time, compound interest can significantly increase the total amount earned, especially when the investment period is long. For example, earning 5% compounded annually will yield more than earning 6% simple interest over the same period, due to the effect of interest being calculated on previously earned interest.

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

A \*\*histogram\*\* is a graphical representation that organizes a group of data points into user-specified ranges (bins). It displays the frequency distribution of numerical data, allowing for a visual interpretation of the underlying frequency distribution (shape) of the data.

In NumPy, you can create a histogram using the `numpy.histogram()` method to compute the histogram data, and then you can use Matplotlib to plot it. However, a direct method to create a histogram plot is `numpy.histogram2d()` for two-dimensional data.

Example of creating a histogram with NumPy and Matplotlib:

```python

import numpy as np

import matplotlib.pyplot as plt

data = np.random.randn(1000) # Generate random data

plt.hist(data, bins=30) # Create histogram

plt.show()

```

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

To change the aspect ratio between the X and Y axes in a plot, you can use the `set\_aspect()` method in Matplotlib. This allows you to specify the aspect ratio of the plot. For example:

```python

import matplotlib.pyplot as plt

plt.plot(x, y)

plt.gca().set\_aspect('equal', adjustable='box') # Set equal aspect ratio

plt.show()

```

You can replace `'equal'` with a specific numeric value to set a custom aspect 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.

1. \*\*Regular Multiplication (Element-wise)\*\*: This operation multiplies corresponding elements of two arrays of the same shape. If `A` and `B` are two arrays, the result is another array where each element is the product of the corresponding elements in `A` and `B`.

```python

C = A \* B # Element-wise multiplication

```

2. \*\*Dot Product\*\*: The dot product (or inner product) is a specific multiplication operation that takes two arrays and returns a single scalar value if both arrays are one-dimensional or a new array if they are two-dimensional. It computes the sum of the products of corresponding elements.

```python

C = np.dot(A, B) # Dot product

```

3. \*\*Outer Product\*\*: The outer product creates a matrix from two vectors, where each element of the first vector is multiplied by each element of the second vector. If `A` is of shape (m,) and `B` is of shape (n,), the result will be an (m, n) array.

```python

C = np.outer(A, B) # Outer product

```

In summary, regular multiplication performs element-wise operations, the dot product computes a single value (or a lower-dimensional array), and the outer product produces a higher-dimensional array.

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

To calculate your monthly mortgage payment using NumPy, you can use the `numpy.pmt()` function. This function calculates the payment amount for a loan based on constant payments and a constant interest rate. The syntax is as follows:

```python

import numpy as np

# Example values

principal = 200000 # Loan amount

annual\_rate = 0.05 # Annual interest rate

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

monthly\_payment = np.pmt(annual\_rate / 12, months, -principal)

print(monthly\_payment)

```

This will give you the monthly mortgage payment based on the principal, interest rate, and loan term.

## 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, one restriction is that NumPy arrays containing strings must have a fixed length. When creating a string array, you need to specify the maximum length of the strings, and any string longer than this length will be truncated. For example:

```python

import numpy as np

string\_array = np.array(['hello', 'world'], dtype='S5') # Fixed-length strings of 5 bytes

```

In this case, if you try to assign a longer string, it will be truncated to fit the specified length.