ASSIGNMENT 23

# Q1

Adjusting the scaling: we can manually set the range or limits of the axes to ensure that the data points are appropriately scaled for comparison.

Using multiple axes: we can create multiple y-axes on the same graph to represent different figures. Each axis can have a different scale, making it easier to compare the values.

Utilising different line styles or markers: By using distinct line styles or markers for each figure, we can visually differentiate them more effectively.

Adding annotations or labels: Including labels or annotations on the graph can provide additional information and context, making it easier to understand and compare the figures.

# Q2

Compound interest has the benefit of exponential growth over time. With compound interest, the interest earned in each period is added to the principal amount, resulting in a higher base for interest calculation in the subsequent periods. This compounding effect leads to the exponential growth of the investment or debt. In contrast, a higher rate of interest without compounding would only apply to the initial principal amount and would not generate additional interest based on the accumulated interest. Over a longer duration, compound interest can significantly amplify the growth or cost compared to a simple higher interest rate without compounding.

# Q3

A histogram is a graphical representation of the distribution of a dataset. It consists of bars, where each bar represents a range or bin of values, and the height of the bar corresponds to the frequency or count of data points falling within that bin. In NumPy, the numpy.histogram method is used to create a histogram. It takes an array of data and bins as input and returns the frequency counts and bin edges.

# Q4

To change the aspect ratios between the X and Y axes, we can use the matplotlib.pyplot.axes method and set the aspect ratio using the set\_aspect function. For example:

import matplotlib.pyplot as plt

# Create a plot

fig, ax = plt.subplots()

# Set the aspect ratio

ax.set\_aspect('equal') # equal aspect ratio

ax.set\_aspect('auto') # automatic aspect ratio based on the data

# Customise the aspect ratio by providing a scalar value

ax.set\_aspect(0.5) # e.g., aspect ratio of 1:2 (Y:X)

# Q5

Dot product: The dot product is performed using the numpy.dot function or the dot operator (@). It calculates the sum of the element-wise product of corresponding elements in the arrays. The dot product results in a scalar value or a 1D array, depending on the dimensions of the input arrays.

Outer product: The outer product is calculated using the numpy.outer function. It computes the multiplication of each element in the first array with every element in the second array, resulting in a new array with the shape (M, N), where M is the size of the first array and N is the size of the second array.

Regular multiplication: Regular multiplication of two NumPy arrays is performed using the \* operator. It performs element-wise multiplication between the corresponding elements of the arrays, resulting in a new array with the same shape as the input arrays.

# Q6

To measure the monthly mortgage payment before buying a home, we would typically use financial functions rather than specific NumPy functions. One common function for mortgage payment calculations is numpy.pmt from the numpy.financial module. It allows us to calculate the periodic payment for a loan based on the principal, interest rate, and loan duration.

# Q7

Yes, string data can be stored in NumPy arrays using the numpy.array function with the dtype parameter set to 'str' or 'object'. However, there are some restrictions when working with string data in NumPy:

Fixed-length strings: NumPy arrays require a fixed length for each element. Therefore, if we allocate a longer string to a NumPy array with fixed-length strings, the string will be truncated to fit the specified length.

Performance considerations: Compared to numerical data, manipulating and performing operations on string data in NumPy arrays can be less efficient due to the variable length and additional overhead associated with string handling.

Limited string-specific operations: NumPy provides fewer string-specific operations compared to other libraries like Pandas. For more advanced string manipulation, we may need to consider using libraries specifically designed for text data processing.