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

In Seaborn, we will plot multiple graphs in a single window in two ways. First with the help of Facetgrid() function and other by implicit with the help of matplotlib.

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 causes your wealth to grow faster. It makes a sum of money grow at a faster rate than simple interest because you will earn returns on the money you invest, as well as on returns at the end of every compounding period.

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

Compound interest causes your wealth to grow faster. It makes a sum of money grow at a faster rate than simple interest because you will earn returns on the money you invest, as well as on returns at the end of every compounding period.

NumPy has a numpy. histogram() function that is a graphical representation of the frequency distribution of data. Rectangles of equal horizontal size corresponding to class interval called bin and variable height corresponding to frequency.

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

Setting figure sizes is one of those things that feels like it should be very straightforward. However, it still manages to show up on the first page of stackoverflow questions for both matplotlib and seaborn. Part of the confusion arises because there are so many ways to do the same thing - this highly upvoted question has six suggested solutions:

* manually create an Axes object with the desired size
* pass some configuration paramteters to seaborn so that the size you want is the default
* call a method on the figure once it's been created
* pass hight and aspect keywords to the seaborn plotting function
* use the matplotlib.pyplot interface and call the figure() function
* use the matplotlib.pyplot interface to get the current figure then set its size using a method

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.

here are three main ways to perform NumPy matrix multiplication:

* np.dot(array a, array b): returns the scalar or dot product of two arrays
* np.matmul(array a, array b): returns the matrix product of two arrays
* np.multiply(array a, array b): returns the element-wise matrix multiplication of two arrays

Scalar multiplication or dot product with numpy.dot:

Scalar multiplication is a simple form of matrix multiplication. A scalar is just a number, like 1, 2, or 3. In scalar multiplication, we multiply a scalar by a matrix. Each element in the matrix is multiplied by the scalar, which makes the output the same shape as the original matrix.

With scalar multiplication, the order doesn’t matter. We’ll get the same result whether we multiply the scalar by the matrix or the matrix by the scalar.

Let’s take a look at an example:

import numpy as np

A = 5

B = [[6, 7],

[8, 9]]

print(np.dot(A,B))

Now, let’s multiply a 2-dimensional matrix by another 2-dimensional matrix. When multiplying two matrices, the order matters. That means that matrix A multiplied by matrix B is not the same as matrix B multiplied by matrix A.

import numpy as np

A = [[6, 7],

[8, 9]]

B = [[1, 3],

[5, 7]]

print(np.dot(A,B))

print("----------")

print(np.dot(B,A))

Matrix product with numpy.matmul:

The matmul() function gives us the matrix product of two 2-d arrays. With this method, we can’t use scalar values for our input. If one of our arguments is a 1-d array, the function converts it into a NumPy matrix by appending a 1 to its dimension. This is removed after the multiplication is done.

If one of our arguments is greater than 2-d, the function treats it as a stack of matrices in the last two indexes. The matmul() method is great for times when we’re unsure of what the dimensions of our matrices will be.

Let’s look at some examples:

Multiplying a 2-d array by another 2-d array

import numpy as np

A = [[2, 4],

[6, 8]]

B = [[1, 3],

[5, 7]]

print(np.matmul(A,B))

Multiplying a 2-d array by a 1-d array

import numpy as np

A = [[5, 0],

[0, 5]]

B = [5, 2]

print(np.matmul(A,B))

One array with dimensions greater than 2-d

import numpy as np

A = np.arange(8).reshape(2, 2, 2)

B = np.arange(4).reshape(2, 2)

print(np.matmul(A,B))

Element-wise matrix multiplication with numpy.multiply

The numpy.multiply() method takes two matrices as inputs and performs element-wise multiplication on them. Element-wise multiplication, or Hadamard Product, multiples every element of the first NumPy matrix by the equivalent element in the second matrix. When using this method, both matrices should have the same dimensions.

Let’s look at an example:

import numpy as np

A = np.array([[1, 3, 5, 7, 9], [2, 4, 6, 8, 10]])

B = np.array([[1, 2, 3, 4, 5], [5, 4, 3, 2, 1]])

print(np.multiply(A,B))

We can pass certain rows, columns, or submatrices to the numpy.multiply() method. The sizes of the rows, columns, or submatrices that we pass as our operands should be the same. Let’s look at an example:

import numpy as np

A = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

B = np.array([[11, 12, 13, 14, 15], [16, 17, 18, 19, 20]])

print(np.multiply(A[ 0,:], B[ 1,: ]))

print("----------")

print(np.multiply(A[ 1,:], B[ 0,:]))

Outer product of two arrays:

To get the Outer product of two arrays, use the numpy.outer() method in Python. The 1st parameter a is the first input vector. Input is flattened if not already 1-dimensional. The 2nd parameter b is the second input vector. Input is flattened if not already 1-dimensional. The 3rd parameter out is a location where the result is stored.

Steps

At first, import the required libraries-

import numpy as np

Creating two numpy One-Dimensional array using the array() method −

arr1 = np.array([5, 10, 15])

arr2 = np.array([20, 25, 30])

Display the arrays −

print("Array1...\n",arr1)

print("\nArray2...\n",arr2)

Check the Dimensions of both the arrays −

print("\nDimensions of Array1...\n",arr1.ndim)

print("\nDimensions of Array2...\n",arr2.ndim)

Check the Shape of both the arrays −

print("\nShape of Array1...\n",arr1.shape)

print("\nShape of Array2...\n",arr2.shape)

To get the Outer product of two arrays, use the numpy.outer() method in Python −

print("\nResult (Outer Product)...\n",np.outer(arr1, arr2))

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

In order to calculate the monthly mortgage payment, you will use the numpy function . pmt(rate, nper, pv) where: rate = The periodic (monthly) interest rate. nper = The number of payment periods (months) in the lifespan of the mortgage loan.

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

The elements of a NumPy array, or simply an array, are usually numbers, but can also be boolians, strings, or other objects.

The dtype of any numpy array containing string values is the maximum length of any string present in the array. Once set, it will only be able to store new string having length not more than the maximum length at the time of the creation