Written by Max Paul

11/16/2022

Question 1

1) Find the number of entries in an array of integers that are divisible by a given integer. Your function should have two input parameters – an array of integers and a positive integer – and should return an integer indicating the count using a return statement.

Run your algorithm on the problem instances:

- [20, 21, 25, 28, 33, 34, 35, 36, 41, 42] number of entries that are divisible by 7
- [18, 54, 76, 81, 36, 48, 99] number of entries that are divisible by 9

Question 1 method

```
def count_divisible(data: list, divisor: int):
    """
    :method:
    count_divisible: a method where we use two inputs;
    to determine the total count of whole divisible integers

    :input:
    data: an array / list of integers
    divisor: a positive int

    :return:
    count: the count of divisible without remainder
    """
    counter = 0
    for i in data:
        if i%divisor==0:
            counter+=1

    return counter
```

Question 1 output for [20, 21, 25, 28, 33, 34, 35, 36, 41, 42] number of entries that are divisible by 7

```
print(count_divisible([20, 21, 25, 28, 33, 34, 35, 36, 41, 42], 7))
```

```
4
```

Question 1 output for [18, 54, 76, 81, 36, 48, 99] number of entries that are divisible by 9

```
print(count_divisible( [18, 54, 76, 81, 36, 48, 99], 9))
```

```
5
```

Question 2

2) Given an array of real numbers, without sorting the array, find the smallest gap between all pairs of elements (for an array A, the absolute value of the difference between elements Ai and Aj). Your function should have one input parameter – an array of numbers – and should return a non-negative number indicating the smallest gap using a return statement.

Run your algorithm on the problem instances:

- <50, 120, 250, 100, 20, 300, 200>
- <12.4, 45.9, 8.1, 79.8, -13.64, 5.09>

Question 2 Method

```
def find_smallest_gap(data: list):
    """
    :param data: a list of real numbers
    :return the smallest gap of all nums
    """
    # out final list of gaps between elements
    gapped_data = []

    for i in data:
        """
         temp_data restarts for
         each element in the list

         for each run we store the gaps;
```

```
            then take the lowest from this list
            """
            temp_data = []
            for y in data:
                """
                if we dont ignore the case where i == y index,
                then we will always have a 0
                forcefully removing 0 is not acceptable as its
                very valid to have a list with 2 same nums
                we need to handle this edge case
                by checking if the index is the same we
                can determine if we want to check or not
                this will make out alg n-1 faster :)
                """

                if data.index(i) != data.index(y):
                    temp_data.append(abs(i-y))
            gapped_data.append(min(temp_data))

    return min(gapped_data)
```

Question 2 output for <50, 120, 250, 100, 20, 300, 200>

```
print(find_smallest_gap([50, 120, 250, 100, 20, 300, 200]))
```

```
20
```

Question 2 output for <12.4, 45.9, 8.1, 79.8, -13.64, 5.09>

```
print(find_smallest_gap([12.4, 45.9, 8.1, 79.8, -13.64, 5.09]))
```

```
3.01
```

Question 3

1) Given an integer n>=2 and two nxn matrices A and B of real numbers, find the product AB of the matrices. Your function should have three input parameters – a positive integer n and two nxn

matrices of numbers– and should return the nxn product matrix using a return statement.

Run your algorithm on the problem instances:

```
[
  [2,7],
  [3,5]
]

X
[
  [8, -4],
  [6,6]
]


  [
    [1,0,2],
    [3,-2,5],
    [6,2,-3]
  ]

  X

  [
    [.3,.25,.1],
    [.4,.8,0],
    [-.5,.75,.6]
  ]
```

Question 3 method

```
def multiply_matrices(X, Y, n):
    """
    :param X: a matrix of numbers
    :param Y: a matrix of numbers
    :param n: the size of matrices

    :return the product of X and Y in Matrix size n
    """

    Matrix = [[0 for x in range(n)] for y in range(n)]

    # iterate through rows of X
    for i in range(len(X)):
```

```
    for i in range(len(X)):
       # iterate through columns of Y
       for j in range(len(Y[0])):
           # iterate through rows of Y
           for k in range(len(Y)):
               Matrix[i][j] += X[i][k] * Y[k][j]
    return Matrix
```

Question 3 output for

```
[
  [2,7],
  [3,5]
]

X
[
  [8, -4],
  [6,6]
]
```

```
print(multiply_matrices([[2,7],[3,5]], [[8,-4],[6,6]], 2))
```

```
    [[58, 34], [54, 18]]
```

Question 3 output for

```
[
  [1,0,2],
  [3,-2,5],
  [6,2,-3]
]

X

[
  [.3,.25,.1],
  [.4,.8,0],
  [-.5,.75,.6]
]
```

+ Code + Text

```
print(multiply_matrices([[1,0,2],[3,-2,5],[6,2,-3]] , [[.3,.25,.1],[.4,.8,0],[-.5,.75,
```

```
[[-0.7, 1.75, 1.3], [-2.4000000000000004, 2.9, 3.3], [4.1, 0.8500000000000001, -
```


✓ 0s completed at 8:28 AM