L11 Tasks

L11-T1: Tuple

L11-T2: Potential pairs of elements

L11-T3: Matrix 1

L11-T4: Matrix 2

Submission:

- Submit **all** the exercises to CodeGrade via Moodle.

Note:

- If you see some "Code Structure Tests" hidden in CodeGrade but they are not mentioned in the task description, you don't need to worry about them. They are just there to make sure the code is ok.

- Be especially careful with spaces so that your output follows the sample output. Note that also each input-string in the program ends with '\n'. The reason for this is that it makes the output in CodeGrade more readable.

L11-T1: Tuple

Write a function powers(x), which returns the powers x^2, x^3, x^4, x^5 using a tuple. The main function asks the user for a number x and after calling the powers(x), it prints the results rounded to 4 decimal places according to the model answer below.

Note: “Code structure tests” will be used to make sure that the tuple structure has been used correctly. Please make sure that your program will have a function call **`powers`** with an **`x`** parameter, then it should return **4 values** using a tuple to the program. You can see the example in the slide “Differences between lists and tuples”.

Example run 1:

```
Enter a number:
56.2
Powers of 56.2:
x^2: 3158.44
x^3: 177504.328
x^4: 9975743.2336
x^5: 560636769.7283
```

L11-T2: Potential pairs of elements

Write a Python program that requests integers from a user and stores them in a list. The program should terminate the input process when the user inputs 'done'. Subsequently, the program must prompt the user to enter another integer. The program's task is to identify all potential pairs of elements from the list. These pairs should have a combined sum equal to the user-provided integer. Additionally, the pairs should consist of elements where the first element is less than or equal to the second element.

The program should not crash when user inputs something that is not an integer.

If it cannot find any pair, print the message "`No pairs found with a sum of [target number].`"

Example run 1:

```
Enter an integer (or type 'done' to finish input):
1
Enter an integer (or type 'done' to finish input):
2
Enter an integer (or type 'done' to finish input):
3
Enter an integer (or type 'done' to finish input):
4
Enter an integer (or type 'done' to finish input):
5
Enter an integer (or type 'done' to finish input):
6
Enter an integer (or type 'done' to finish input):
done
Enter the target sum:
7
Pairs with a sum of 7:
(1, 6)
(2, 5)
(3, 4)
```

L11-T3: Matrix 1

In lectures, we have learned how to install the *NumPy* library as a Python extension.

In this task, we go through the basic operation of the numpy matrix. If you don't remember how it worked, you can run `pip install numpy`

Write a program that first asks the user to provide two matrices. This is done by asking first the dimensions of a matrix. Then the program asks the user to give the entry values, one row at a time. When all rows are given, the program creates a numpy 2D array from these nested

lists. Then it tries to compute the sum and the product of the given matrices. If this is not possible, print error messages according to the examples below.

In CodeGrade NumPy is installed.

Example run 1:

```
Enter the number of rows for the first matrix:
2
Enter the number of columns for the first matrix:
2
Enter values for a 2x2 matrix:
Enter 2 values for row 1 (separated by space):
1 5
Enter 2 values for row 2 (separated by space):
3 11
This is matrix 1:
[[ 1.  5.]
 [ 3. 11.]]
Enter the number of rows for the second matrix:
2
Enter the number of columns for the second matrix:
3
Enter values for a 2x3 matrix:
Enter 3 values for row 1 (separated by space):
5 8 9
Enter 3 values for row 2 (separated by space):
2 1 -4
This is matrix 2:
[[ 5.  8.  9.]
 [ 2.  1. -4.]]
Error: sum not possible
Matrix multiplication:
[[ 15.  13. -11.]
 [ 37.  35. -17.]]
```

Example run 2:

```
Enter the number of rows for the first matrix:
2
Enter the number of columns for the first matrix:
3
Enter values for a 2x3 matrix:
Enter 3 values for row 1 (separated by space):
1 9 -3
Enter 3 values for row 2 (separated by space):
11 -2 3
This is matrix 1:
[[ 1.  9. -3.]
 [11. -2.  3.]]
Enter the number of rows for the second matrix:
2
Enter the number of columns for the second matrix:
3
Enter values for a 2x3 matrix:
Enter 3 values for row 1 (separated by space):
-11 9.3 0
Enter 3 values for row 2 (separated by space):
77 -9.3 8
This is matrix 2:
[[-11.    9.3   0. ]
 [ 77.   -9.3   8. ]]
Matrix sum:
[[-10.   18.3  -3. ]
 [ 88.  -11.3  11. ]]
Error: multiplication not possible
```

Example run 3:

```
Enter the number of rows for the first matrix:
2
Enter the number of columns for the first matrix:
2
Enter values for a 2x2 matrix:
Enter 2 values for row 1 (separated by space):
1 2
Enter 2 values for row 2 (separated by space):
3 4
This is matrix 1:
[[1. 2.]
 [3. 4.]]
Enter the number of rows for the second matrix:
2
```

```
Enter the number of columns for the second matrix:
2
Enter values for a 2x2 matrix:
Enter 2 values for row 1 (separated by space):
5 6
Enter 2 values for row 2 (separated by space):
7 8
This is matrix 2:
[[5. 6.]
 [7. 8.]]
Matrix sum:
[[ 6.  8.]
 [10. 12.]]
Matrix multiplication:
[[19. 22.]
 [43. 50.]]
```

L11-T4: Matrix 2

Follow the instructions below:

1. Create a function that takes user input and creates a matrix of the given input (see example run).
2. Make the user's defined matrix to be full of zeroes. Use numpy's zeros member function. Print the result.
3. Go through all the elements of the matrix with two nested loops (loop inside a loop). Loop first over the rows and then the columns
    a. Change the values in the matrix, so we do not have a matrix full of zeroes.
    b. Calculate the new values by taking the row_index (`row`) and column_index (`column`). The new entry value is:

        ```
        (row+1) * (column+1))
        ```

        see the example output below.
4. After this, print the filled matrix with the print command, so it follows the formatting provided by numpy.
5. Use `numpy`'s sort function to sort the numbers and remove all the axis at the same time (see np.sort() documentation on how it works)
6. Then reprint the matrix row by row with columns separated by semicolons as shown in the example run below. This format is easy to transfer to a spreadsheet program for visualization.
7. Next, create a function that takes user input and shapes the Matrix into a new shape. Our example run has 4x4-matrix that is then transformed into a 2x8-matrix.

a. Catch errors here! If the user inputs faulty dimensions (e.g., 3x3), **do not** let the program ***crash*** but ask the user to input proper values instead!

8. Next, find the individual maximum value and individual minimum value in the matrix. Calculate the sum of all entries in the matrix. Use the `numpy` member functions, min, max & sum here

9. Finally, create a function that reads a list of integers from the user. Then, remove all duplicates and sort the array to be from smallest to largest (there is a unique member variable on `numpy`!)

Use the following command to read a complete list in one row:

```
list_values = [int(item) for item in input("Enter the list items: \n").split()]
```

Example run 1:

```
This program demonstrates use of numpy-matrix.
Enter amount of rows:
4
Enter number of columns:
4
Zero-matrix of the given rows & columns is:
 [[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
Matrix printed with np-formatting:
[[ 1  2  3  4]
 [ 2  4  6  8]
 [ 3  6  9 12]
 [ 4  8 12 16]]

Matrix sorted into one array:
[ 1  2  2  3  3  4  4  4  6  6  8  8  9 12 12 16]
Matrix printed with elements separated by semicolons:
1;2;3;4;
2;4;6;8;
3;6;9;12;
4;8;12;16;

Shaping the matrix. Please, enter the new dimensions.
Enter amount of new rows:
3
Enter amount of new columns:
3
Faulty shape. Please, try again.
Shaping the matrix. Please, enter the new dimensions.
```

```
Enter amount of new rows:
2
Enter amount of new columns:
8
Newly shaped matrix is:
 [[ 1  2  3  4  2  4  6  8]
 [ 3  6  9 12  4  8 12 16]]
Largest number in the matrix is: 16
Smallest number in the matrix is: 1
Sum of all values in the matrix is: 100
Enter the list items:
11 11 12 13 13 15 16 16 18 18 22 22 18 5 5 44 3 4 55 32 34 44
Unique values are: [ 3  4  5 11 12 13 15 16 18 22 32 34 44 55]
```