Week 11

Write and run your programs with IDLE editor. Submit finished programs to CodeGrade. Note that some tasks have several steps (A, B, C, …) in CodeGrade.

IMPORTANT: End each input-command string with a newline symbol \n. For example:

```
variable = input("Some text:\n")
```

Task 1. Create a module `my_stats`, which contains three functions `my_mean(L)`, `my_variance(L)`, `my_mode(L)`, returning the mean, the (population) variance and the mode of the integers in the list `L`, respectively. You can assume that L is always nonempty.

- `my_mean(L)` is calculated by adding up all the numbers in the set and dividing the sum by the total number of values.
- `my_variance(L)` is calculated by computing the average of the squared differences between each data point in the population and the mean.
- `my_mode(L)` is the value that appears most frequently in a dataset. If there are several of them, return the biggest.

When implementing these functions, you are **not allowed** to use any modules. Submit the module file `my_stats.py` to CodeGrade and the system tests your module. There is no example run in this task.

Task 2. Write a function `powers(x)`, which returns the powers x^2, x^3, x^4, x^5 using a tuple. The main function asks 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.

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
```

Task 3: 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.

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)
```

Task 4. Write a program that first ask 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
```