Exercise 1

Due Date: 1.4.2020, 12:00

In the following exercise you will implement few Python functions. You are not allowed to use any packages at all. You do not need to check the type of the arguments unless asked to do so.

Note: Every pair should upload two files to the Moodle: a code file (ex1.py) and a docx\pdf file. The docx\pdf will include answers that are not code. Every code file that you send is being checked automatically. Therefore, it is very important not to change the functions' names and the input arguments. In all exercises, you will be given few testing examples. Use them to check your code. Try to think of additional tests that might be interesting to check.

Every code file should start with two comment lines, the first line should include your IDs (separated by a comma) and the second should include the number of hours you worked on this exercise. For example:

```
# id1, id2
# 3.5
```

1. For this question you should download the "DebuggingExampleExercise.py" file from the Moodle. Use the debug mode to answer the following questions: (answers in the docx\pdf)
    a. Put breakpoint on line 9. What is the value of *first_appearance* after running this line for the first time?
    b. What is the value of *i* in the last iteration of the loop in *iter_l* function?
    c. Explain what happens when running *iter_l* function
    d. What is the value of *cur_list* after running the code?

2. In this question, you will program two solutions for the same problem. You goal is to write two functions (*min_count1* and *min_count2*) which calculate the minimum value within a list. The functions return a list containing the minimum value and the number of times it appears in the list. (answers in the code file)

    Testing example:
    min_count1([1,4,2,10, 102]) -> [1, 1]
    min_count2([1,4,2,10, 102]) -> [1, 1]
    min_count1([-1,2,-1,3]) -> [-1, 2]
    min_count2([-1,2,-1,3]) -> [-1, 2]

3. In this question, you will learn a new function called *filter.*
    a. Google it and explain what the function does. (answers in the docx\pdf file)
    b. Use filter to write a function called *only_even* with two arguments. The function goes over the numbers from argument 1 to argument 2 (including argument 1 and not including argument 2) and returns a list with only even numbers in the given range. The list should be sorted from smallest to largest (answers in the code file)
    c. Use filter to write a function called *only_strings* which returns a list with only the strings which appeared in the original list, in the same order as in the original list. (answers in the code file)

    Testing example:
    only_even(3,10) -> [4,6,8]
    only_strings([1,'a',2]) -> ['a']

Good Luck!