Instructions:

This assessment consists of eight simple exercises (see the next page). Each exercise asks you to program a function. The following information is given:

- The function header.
- A function description.
- Some examples of what the function should do.

Implement these eight functions in a single Python file named 'code.py' (otherwise the automatic tests will not accept your program). You can assume that the input always adheres to the assignment description (for instance, if the assignment description says that a parameter is an integer ≥ 1, you can assume that this will always be the case). Your program should not modify the value of the arguments.

You can test your program with the examples that are given in the exercise description, but be aware that the function should work for any input that adheres to the assignment description, and not just those examples.

Exercises:

1. Program a function `def f1(an_int)`. The parameter `an_int` is an integer ≥ 1. The function should return the greatest power of 3 that is smaller than or equal to `an_int`.

Examples:
`f1(1)` should return `1` (since 3^0 is 1)
`f1(10)` should return `9` (since 3^2 is 9, and 3^3 is 27)

2. Program a function `def f2(an_int)`. The parameter `an_int` is an integer ≥ 0, and represents an amount of time in seconds. The function should return a string that contains the time in the format hh:mm:ss.

Examples:
`f2(234)` should return `'00:03:54'`
`f2(7322)` should return `'02:02:02'`

3. Program a function `def f3(a_list)`. The parameter `a_list` is a list of unique integers, that contains at least 2 elements. The function should return a tuple containing the index and the value of the second smallest element in `a_list`.

Examples:
`f3([4, 5, 6])` should return `(1, 5)`
`f3([-3, -5, 7])` should return `(0, -3)`

4. Program a function `def f4(a_list1, a_list2, an_int)`. The parameters `a_list1` and `a_list2` are both a list of integers, and the parameter `n` is an integer ≥ 1. The function should return how many elements from `a_list2` appear at least `n` times in `a_list1`.

Examples:
`f4([1, 1, 2, 2, 2, 3, 3, 3, 3], [1, 2], 3)` should return `1`
`f4([1, 1, 1, 2, 2, 2, 3, 3, 3], [1, 2, 3, 4], 4)` should return `0`

5. Program a function `def f5(a_list)`. The parameter `a_list` is a list of integers. The function should return a list of slices from `a_list`. The first slice in this list should contain the first element from `a_list`, the second slice in this list should contain the first 2 elements from `a_list`, etc.

Examples:
`f5([1, 2, 3])` should return `[[1], [1, 2], [1, 2, 3]]`
`f5([0, 0])` should return `[[0], [0, 0]]`
`f5([])` should return `[]`

6. Program a function `def f6(a_str)`. The parameter `a_str` is a string. The function should return a new string, that contains all characters from `a_str` that are not a letter or digit.

Examples:
`f6('B -34;aJK+]\t>')` should return `' -;+]\t>'`
`f6('python')` should return `''`

7. Program a function def `f7(a_str)`. The parameter `a_str` is a string that contains words (consisting of lowercase letters) separated by spaces. The function should return a dictionary that contains as keys each word from `a_str`, and as values the number of times that each word appears in `a_str`.

Examples:
`f7('de het de een')` should return `{'de': 2, 'het': 1, 'een': 1}`
`f7('')` should return `{}`

8. Program a function `def f8(a_str)`. The parameter a_str is a string that consists of multiple lines with exam results. Each line either contains a grade (in the range [1.0, 10.0]), or an abbreviation (for example, 'NS' means 'No Show'). At least one line contains a grade. The function should return the average of all grades that appear in `a_str`.

Examples:
`f8('8\n8.2\n4.3\n9.5')` should return `7.5`
`f8('7.5\nNS\n7\n3.5\nNVD')` should return `6.0`