At the beginning, some work with lists. Some of the exercises, and especially the last one, need a little thought; if you don't know how to resolve it, try to look on the internet or ask other colleagues to solve it together.

1. Make a list of pets. You will need it in later exercises. Known pets are: 'dog', 'cat', 'rabbit', 'snake'

2. Write a function that returns the name of pets (list given as the argument) that are shorter than 5 letters.

3. Write a function that returns the names of pets (list given as the argument) that begin with 'd'.

4. Write a function that gets a word and detects if it's in the pet list. “Detects” means that the function returns *True* or *False*.

5. Write a function that gets two lists of animal names and returns three lists:
(a) the animals that are on both lists (common ones),
(b) animals which are only in the first list,
(c) animals that are only in the second list
Write the test to verify that it works properly.

6. Write a program that sorts the list of pets by alphabet.

7. Write a function that sorts the animals alphabetically, but ignores the first letter
Returns: ['rabbit', 'cat', 'snake', 'dog']
Procedure:
- you have a list of values that you want to sort by a key. The key can be calculated from each value.
- create a list of pairs (key, value)
- sort this list of pairs – the pair is first sorted according to the first element, then the second one
- finally, create a list of values from the list of pairs

8. Write the following functions. Each one gets the birth number string as the argument and somehow analyzes it:
(a) is it in the correct format: 6 digits, slash, 4 digits? (returns True or False)
(b) is it divisible by 11? (returns True or False)
(c) What birthday is encoded in the number? (returns three numbers – day, month, year)
(d) What gender is encoded in the number? (returns 'man' or 'woman')
Write tests to verify that the functions work properly.
For the purpose of this exercise, it is enough to be able to process the numbers issued after 1985. Real birth numbers can be more complicated.

9. Write a program that asks the user for the birth number and outputs the results.

One classical programming exercise, which is probably going to be a bit tricky. It's optional, so if you don't have a few hours to spare, you can skip it

10. Write a function that converts the Roman numerals to Arab numbers (integers)
tip: first write tests to verify that (and what) (not) works properly.

Exercises 11-17 depend on each other, and provides the solution gradually. If you do them, you can play

the game! This section is not easy (and last 2 ex. are particularly challenging). You can try to join forces with other participants of the course.

11. Write a function that gets a list of coordinates (pairs of numbers smaller than 10, that specify the column and row) and output them as a map: a 10 x 10 grid, where the fields that are in the list are written in X, and elsewhere, dots. For example:
draw_map([(0,0),(1,0),(2,2),(4,3),(8,9),(8,9)])

```
X X . . . . . .
. . . . . .
. . X . . . . . .
. . . . X . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . X .
```

How to do it?
- make a table (list of lists) with the same dots, something like:
[['.','.','.'],['.','.','.'],['.','.','.']]
- replace the dots with X in the appropriate places
- write the table using 2 nested *for* loops

12. Write a movement function, that get the coordinates as a list and the direction keyword ('n','s','e','w') and adds to that list, the last point “shifted” in that direction. Ex:

```
s,j,v,z
coordinates=[(0,0)]
movement(coordinates, 'e')
print(coordinates)     # → [(0,0),(1,0)]
movement(coordinates, 'e')
print(coordinates)     # → [(0,0),(1,0),(2,0)]
movement(coordinates, 's')
print(coordinates)     # → [(0,0),(1,0),(2,0),(2,1)]
movement(coordinates, 'n')
print(coordinates)     # → [(0,0),(1,0),(2,0),(2,1),(2,0)]
```

This function should return nothing. It just changes an existing list.
Don't forget about the tests.

13. Write a loop, that will ask the user for the map coordinates, then asks for the movement, and then draws the list as a map. Then it will ask again, etc.
Start with the list of [(0,0),(1,0),(2,0)]

14. Change the movement function so that it changes the coordinates of the first point according to the direction. The resulting list must have the same length as the initial one.
Change the tests accordingly.

15. Update the movement function to prevent:
- moving out of the map,
- move to a point that is already in the list,
A good exception for these situations is ValueError('Game Over')
Update the tests.

16. Add snake food to the game. Here are the rules for a vegetarian snake, but you can change them according to your taste:
- in the beginning, the fruit list contains one fruit in a field where there is no snake (for ex. [(2,3)]
- this means one fruit in position [(2,3)]. When the snake eats the fruit, it grows (the tail 'does not disappear', meaning don't do what you added in exercise 14), and if there is no other fruit on the map, new one grows in a random place (where there is no snake)
- every 30 moves, new fruit will create itself
- this mysterious fruit is displayed on the map as a question mark (?)

17. The map can have any size greater than 4x1. Can be 20x20 or 10x30

And now something to think about.

18. Can the list contain itself? Try to make the list as simple as possible so that this applies:
list[5][5][5][5][5][5][5][5][5][5][5][5][5][5][5][5][5][5][5][5][0] == 5