# Lab 3

Sol Ben-Ishay

1. **Implement/Discuss:** If you are asked to reimplement the tictactoe.py in Lab 1 can it be easily modified for the following requirements:
    a. The size of the board is not 3 by 3. It will be given as an input n.
    b. The markers will not be X/O but will be given as inputs for player 1 and player 2.
    c. Try to modify your code and add it to your project if it is easy to do so. If not discuss the reasons.
    d. **WORKED ON MODIFYING MY CODE FROM LAB 1, BUT COULD NOT FIGURE IT OUT HOW TO DO THE DIAGONALS FLEXIBLY.**
2. **Try/Discuss:** Download ***namespaces.py*** file from Nexus. Add it to your project and run. id(x) returns the address of x. Discuss the followings by considering the program and the output:
    a. Which variables are in the global namespace?
        i. HOW_MANY, n, a_list, n_new
    b. How many local namespace do we have?
        i. Two
    c. Does the n variable have the same address in the global namespace and local namespace?
        i. No, in the global namespace n is equal to 1. In function simple_data() it is equal to 6 and in function complex_data() it is equal to 2.
    d. Are the changes in the functions reflected outside for n?
        i. No
    e. Does the a_list variable have the same address in the global namespace and local namespace?
        i. No, a_list is equal to [5, 10, 15, 20] in the global namespace and becomes [5, 10, 18, 20] in function complex_data().
    f. Are the changes in the functions reflected outside for a_list?
        i. No
    g. What is the difference between n and a_list?
        i. n is an integer and a_list is a list.
3. **Implement/Discuss:** Download the ***recursive_count.py*** and ***test_recursive_count.py*** and add them to your project. Run both files and observe the results. The first one has functions for counting a target in a given list. One function considers the nested lists but not the other. It has manual tests as well in the main part. The second file is a unittest and provides a couple of the tests by using unittest. Now it is your turn. First write two functions ***recursive_len*** and ***recursive_len_nested*** in a file ***recursive_length.py.*** Both functions will have a list parameter. They will return the number of items in a given list. Nested version will count the items in the nested list as well. In your main test your functions. Then add ***unittest*** in PyCharm. The file name will be ***test_recursive_length.py***. Test your functions with simple unittests.

a. **WAS HAVING TROUBLE FIGURING OUT HOW TO CREATE THE recursive_len_nested() function. Created the recursive_len() function succesfully though.**

```
recursive_length.py


def recursive_len(my_list):
    """Gets the length of a list

    :param my_list: The list that's length will be measured
    :type my_list: list
    :returns: the length of my_list
    :rtype: int

    """

    # Empty list result is zero: Base case
    if not my_list:
        return 0
    else:
        return 1 + recursive_len(my_list[1:])


def recursive_len_nested(my_list):
    """Gets the length of the individual values in a nested list

    :param my_list: The list that's length will be measured
    :type my_list: list
    :returns: the length of my_list
    :rtype: int

    """

    # Empty list result is zero: Base case
    if not my_list:
        return 0
    # TODO Not sure how to complete recursive length function
for the nested list
    # else:


print(recursive_len([1, 3, 4, 6, 7, 8, 9]))
```

4. **Discuss:** You are assigned a project to implement a game. The definition is as follows:
    a. ***Write a board game where we have a farmer, a rabbit, a dog, and many***

***carrots in a farm. The farmer would like to harvest his carrots before the rabbit eats them. The rabbit would like to eat carrots and save its life, since the farmer and dog can catch it. The dog would like to catch the rabbit.***

b. What do you think about this definition?
    i. Although it gives a decent idea of what the game should have in it, it is not fully descriptive of what the user wants. There are still many missing requirements that if not defined, can cause different created versions of this game to look and act much different.
c. What is missing?
    i. Some of the missing information includes the size of the map, the speed of the farmer, rabbit, and dog, how the farmer and dog "catch" the rabbit, what kind of movements will the characters be able to make, what happens when the farmer or dog "catches" the rabbit.
d. Ask questions to clarify the problem.
    i. How big should the map be?
    ii. What directions will the players be able to move? X,Y plane? 3D?
    iii. How fast will each of the characters be?
    iv. What happens when the dog or farmer "catches" the rabbit? Is there a struggle or is he automatically caught? Is the game over?
    v. Countless more potential questions
e. Think about the design steps. Divide and conquer the problem.
    i. Some steps you are going to want to break down is how to represent the "map" or "game board", the movement range of the players, character actions, and much more

5.