Homework 3

Dictionaries and structuring data

Out 9/21 – Due 10/6

Exercise 1. Automate the boring stuff, page 103, Character Picture Grid (25pts).
You have a list of lists where each value in the inner lists is a one-character string, like this:

```
grid = [['.', '.', '.', '.', '.', '.'],
        ['.', '0', '0', '.', '.', '.'],
        ['0', '0', '0', '0', '.', '.'],
        ['0', '0', '0', '0', '0', '.'],
        ['.', '0', '0', '0', '0', '0'],
        ['0', '0', '0', '0', '0', '.'],
        ['0', '0', '0', '0', '.', '.'],
        ['.', '0', '0', '.', '.', '.'],
        ['.', '.', '.', '.', '.', '.']]
```

You can think of grid[x][y] as being the character at the x- and y-coordinates of a "picture" drawn with text characters. The (0, 0) origin will be in the upper-left corner, the x-coordinates increase going right, and the y-coordinates increase going down. Copy the previous grid value, and write code that uses it to print the image.

```
..OO.OO..
.OOOOOOO.
.OOOOOOO.
..OOOOO..
...OOO...
....O....
```

Hint: You will need to use a loop in a loop in order to print grid[0][0], then grid[1][0], then grid[2][0], and so on, up to grid[8][0]. This will finish the first row, so then print a newline. Then your program should print grid[0][1], then grid[1][1], then grid[2][1], and so on. The last thing your program will print is grid[8][5]. Also, remember to pass the end keyword argument to print() if you don't want a newline printed automatically after each print() call. Save your code as *hw3_firstname_lastname_ex_1.py*

Exercise 2. Number of occurrences (25pts).
Develop a program that counts the number of occurrences of each character in a string, including letters, punctuations, and white spaces. Letters are case sensitive ('T' and 't' are different). Store the result in a dictionary, with the characters as the keys, and number of occurrences as their values. You can use the following string to test (but your code should work with any string input):

- The quick brown fox jumps over the lazy dog.

Note that this string contains all the letters of the English alphabet. Finally, use the module pprint for pretty printing of dictionaries.

```
import pprint
pprint.pprint(dictionary)
```

or

```
spam = pprint.pformat(dictionary) # Pretty text as a string value
print(spam)
```

Save your code as *hw3_firstname_lastname_ex_2.py*

Exercise 3. Inventory (25pts).
Imagine you have this inventory in a store:

Item	Number of item
Hand sanitizer	10
Soap	6
Kleenex	22
Lotion	16
Razors	12

1. Save the inventory in a data structure.
2. Write a function printInventory(inventory) that will take any possible inventory as the input, and display it in a nice format.
3. Write a function addItem(inventory, item) to add one new item to the data structure each time it is called. For example, calling addItem(inventory, 'Advil') for the first time will create an item Advil with a count of 1; calling it the second time will increase Advil's count to 2, and so on. Calling addItem(inventory, 'Lotion') for the first time will increase Lotion's count from 16 to 17.
4. Write a function deleteItem(inventory, item) to delete items (but don't delete a whole item group). It has the opposite effect of addItem(inventory, item). For example, calling deleteItem(inventory, 'Soap') for the first time will reduce Soap's count to 5, and so on, until the item's count reaches 0.
5. Save your code as *hw3_firstname_lastname_ex_3.py*

Exercise 4. Tic-Tac-Toe (25pts).
In this exercise we are going to create a Tic-Tac-Toe game.

1. Create the data structure
   - Nine slots that can each contain an X, an O, or a blank.
   - To represent the board with a dictionary, you can assign each slot a string-value key.
   - String values in the key-value pair to represent what's in each slot on the board:
     - 'X'
     - 'O'
     - ' '
2. Create a function to print the board dictionary onto the screen
3. Add the code that allows the players to enter their moves
4. Save your code as *hw3_firstname_lastname_ex_4.py*

Note: This isn't a complete tic-tac-toe game — for instance, it doesn't check whether a player has won — but it's enough to see how data structures can be used in programs.

Submit your code files **in a zipped archive named *hw3_firstname_lastname.zip***. Comment everything so we know you wrote the code! On top of your files write this multiline comment with your information:

```
"""
Homework 3, Exercise 1 (or 2…)
Name
Date
Description of your program.
"""
```