Assignment 4: Using Python building blocks

In this assignment you will work with lists, dictionaries, loops, and functions.

Exercise A: Understanding functions

1. Read the code below. What is it doing? And what will it print when it's run?
2. Use your debugger to step through the code line by line. How many times is the second line (`augmented_string = input_string + " really"`) run?
3. What is the value of `second_variable` ?

Hand in a text file (`A.txt`) with your answers.

In [ ]:

```python
def augment(input_string):
    augmented_string = input_string + " really"
    return augmented_string


initial_variable = "I"
second_variable = augment(initial_variable)
third_variable = augment(second_variable)
print(third_variable)
fourth_variable = augment(second_variable)
fifth_variable = fourth_variable + " like icecream"
print(fifth_variable)
```

Exercise B: Using functions

Write a function that receives a number as its input, and returns that number incremented by one.

Test the function by printing some of it's results for various inputs.

Call your Python file for hand-in `B.py` .

Exercise C: Counting words (1/4)

Imagine you were to write a function called `count_word_in_text` that counts the occurrence of a single word from some text. Write down a few bullet points about how you would break this down into smaller problems. Think about this:

- What is the input?
- What is the expected output?
- What things might you need to save in variables as the program runs?

Call your text file for hand-in `C.txt` .

Exercise D: Counting words (2/4)

Now, we can move on to writing the actual function. Your function should take one input argument (the word to search for) and return the number of times that the word appears in the variable `text` (see below). Here are some examples of input and expected output:

```
'woman' -> 2 (or 3 if you count the string 'woman.')
'drawing' -> 1
'was' -> 4 (or 5 if you count the string 'was,')
```

- *Hint*: You can split a piece of text into a list where each element is a word like so:

  ```
  print(text.split()) # ['To', 'Sherlock', 'Holmes', ...]
  ```

A definition for your function could look like this:

```
def count_word_in_text(word):
    # Implement me! Write your code below
    raise NotImplementedError('Replace this line with your code!')
```

In [ ]:

```
text = '''
To Sherlock Holmes she is always the woman. I have seldom heard him
mention her under any other name. In his eyes she eclipses and
predominates the whole of her sex. It was not that he felt any
emotion akin to love for Irene Adler. All emotions, and that one
particularly, were abhorrent to his cold, precise but admirably
balanced mind. He was, I take it, the most perfect reasoning and
observing machine that the world has seen, but as a lover he would
have placed himself in a false position. He never spoke of the softer
passions, save with a gibe and a sneer. They were admirable things
for the observer--excellent for drawing the veil from men's motives
and actions. But for the trained reasoner to admit such intrusions
into his own delicate and finely adjusted temperament was to
introduce a distracting factor which might throw a doubt upon all his
mental results. Grit in a sensitive instrument, or a crack in one of
his own high-power lenses, would not be more disturbing than a strong
emotion in a nature such as his. And yet there was but one woman to
him, and that woman was the late Irene Adler, of dubious and
questionable memory.
'''
```

Call your Python file for hand-in `D.py` .

Exercise E: Counting words (3/4)

Write a second function, that iterates over all the words in the `text` variable and adds them as a key-value pair of the word and the number 0 to a dictionary. The result should be a dictionary where all the keys are the unique words in the text, and all the values are numbers.

- *Hint*: You can create a dictionary like so:

  ```
  my_dictionary = {}
  ```

- *Hint*: You will have to create a value in your dictionary *if it does not already exist*. Two ways to do this is by iterating through the text twice, or use the `setdefault` method (google knows what this means). If you have a better idea, feel free to try it out!

Call your Python file for hand-in `E.py` .

Exercise F: Counting words (4/4)

Write a third function that counts the occurrence of all the words in the text. The output should be a dictionary where the keys are all the words in the text, and the values are a number, indicating how many times that word occurred in the text.

- *Hint*: Inspect your dictionary from before. What are the keys? Can you exploit that somehow?
- *Hint*: You can iterate over a dictionary like so (print() statements simply to show what .items() does):

  ```
  my_dict = {'one_key': 2, 'some_key': 3, 'a_key': 5, 'still_key': 7}
  for key, value in my_dict.items():
      print(key)
      print(value)
      print()
  ```

Call your Python file for hand-in `F.py` .

Exercise G: Printing hearts

This is the practice project "Character Picture Grid" from chapter four of "*Atomate the Boring Stuff with Python*", see bottom of [http://automatetheboringstuff.com/chapter4/ (http://automatetheboringstuff.com/chapter4/)](http://automatetheboringstuff.com/chapter4/)

Say you have a list of lists where each value in the inner lists is a one-character string, like this:

```
In [ ]:
```

```python
grid = [['.', '.', '.', '.', '.', '.'],
        ['.', 'O', 'O', '.', '.', '.'],
        ['O', 'O', 'O', 'O', '.', '.'],
        ['O', 'O', 'O', 'O', 'O', '.'],
        ['.', 'O', 'O', 'O', 'O', 'O'],
        ['O', 'O', 'O', 'O', 'O', '.'],
        ['O', 'O', 'O', 'O', '.', '.'],
        ['.', 'O', 'O', '.', '.', '.'],
        ['.', '.', '.', '.', '.', '.']]
```

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.

Use the previous grid value, and write a function that uses it to print the image like so:

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

Hint: You will need to use a loop inside another 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 do not want a newline printed automatically after each `print()` call.

Call your Python file for hand-in `G.py`.