Hlutapróf 2 - practice problems

The following program calls three functions.
The first function, **get_input_list**, takes an integer as a parameter. It then asks the user to enter numbers, one at a time, as many times as the integer parameter value. If the parameter is not a positive integer the function returns None, otherwise it returns a list of all the input values from the user. The list should be exactly as long as the number of inputs. The function prompts the user for each input in clear English, but the exact phrasing of the prompts doesn't matter.
The second function, **average_of_list**, takes a list of numbers as a parameter and returns the average of all the numbers in the list.
The third function, **triple_values**, takes a list of numbers as a parameter and replaces or changes each value within the list so that each number value is 3 times as big.

```
# FUNCTION get_input_list HERE

# FUNCTION average_of_list HERE

# FUNCTION triple_values HERE

number_of_inputs = int(input("How many numbers will you input? "))
input_list = get_input_list(number_of_inputs)
if input_list == None:
    print("Not valid!")
else:
    print("The list: ", input_list)
    average = average_of_list(input_list)
    print("The average of the values is: ", average)
    triple_values(input_list)
    print("Now the list is: ", input_list)
```

<u># Example input/output 1:</u>

```
How many numbers will you input? 4
Please enter the next number: 2
Please enter the next number: 4
Please enter the next number: 5
Please enter the next number: 8
The list:  [2, 4, 5, 8]
The average of the values is:  4.75
Now the list is:  [6, 12, 15, 24]
```

<u># Example input/output 2:</u>

```
How many numbers will you input? 0
Not valid!
```

Implement all three functions, **get_input_list**, **average_of_list** and **triple_values**.

Using a dictionary

Write this whole program from scratch.
This program uses pairs of ID and name to register and then look up names.
Your program should first ask the user how many pairs will be entered.
Next, as often as the user asked for, the program prompts for an ID first, then a name.
In the example numbers are used for ID, but the IDs can be any string. There is no need to type-check the IDs or names.
When all the ID/name pairs have been entered the program runs until the user inputs ‘q’, each time asking which ID to look up and if the ID is in the collection, print out the respective name, otherwise state that the ID is not in the collection.
Once ‘q’ is entered the program prints “Quitting…” and stops running.

Example input/output:

```
How many pairs? 3
What is the ID? 4
What is the name? Anna
What is the ID? 7
What is the name? Elsa
What is the ID? 3
What is the name? Sven
What id to look up (q to quit)? 1
The id is not in the collection.
What id to look up (q to quit)? 2
The id is not in the collection.
What id to look up (q to quit)? 3
The name for  3  is  Sven
What id to look up (q to quit)? 4
The name for  4  is  Anna
What id to look up (q to quit)? 5
The id is not in the collection.
What id to look up (q to quit)? 6
The id is not in the collection.
What id to look up (q to quit)? 7
The name for  7  is  Elsa
What id to look up (q to quit)? q
Quitting...
```

Building a class

The class TickCounter has a constructor (**__init__** function) and three functions, **tick**, **get_count** and **reset**. The function **tick** can be called as many times as the user wants. The function **get_count** returns how many times **tick** has been called since the class instance was created or since the function **reset** was called.
The function **reset** restarts the counter at 0.

```
# CLASS DEFINITION FOR TickCounter HERE
```

```
if __name__ == "__main__":
    counter = TickCounter()
    for _ in range(45):
        counter.tick()
    print("The current count: ", counter.get_count())
    for _ in range(12):
        counter.tick()
    print("The current count: ", counter.get_count())
    counter.reset()
    for _ in range(31):
        counter.tick()
    print("The current count: ", counter.get_count())
```

Example output:

```
The current count:  45
The current count:  57
The current count:  31
```