Programming Exercise Forum Week 6

Exercise 1

Given the following dictionary:

```
inventory = {
    'gold' : 500,
    'pouch' : ['flint', 'twine', 'gemstone'],
    'backpack' : ['xylophone','dagger', 'bedroll','bread loaf']
}
```

Try to do the following:

- Add a key to inventory called 'pocket'.
- Set the value of 'pocket' to be a list consisting of the strings 'seashell', 'strange berry', and 'lint'.
- .sort()the items in the list stored under the 'backpack' key.
- Then .remove('dagger') from the list of items stored under the 'backpack' key.
- Add 50 to the number stored under the 'gold' key.

Exercise 2

Follow the steps bellow: -Create a new dictionary called prices using {} format like the example above.

- Put these values in your prices dictionary:

  ```
  "banana": 4,
  "apple": 2,
  "orange": 1.5,
  "pear": 3
  ```

- Loop through each key in prices. For each key, print out the key along with its price and stock information. Print the answer in the following format:

  ```
  apple
  price: 2
  stock: 0
  ```

- Let's determine how much money you would make if you sold all of your food.
  - Create a variable called total and set it to zero.
  - Loop through the prices dictionaries. For each key in prices, multiply the number in prices by the number in stock. Print that value into the console and then add it to total.
  - Finally, outside your loop, print total.

Exercise 3

Follow the steps:

- First, make a list called groceries with the values "banana","orange", and "apple".
- Define these two dictionaries:

```
stock = {
   "banana": 6,
   "apple": 0,
   "orange": 32,
   "pear": 15
}

prices = {
   "banana": 4,
   "apple": 2,
   "orange": 1.5,
   "pear": 3
}
```

- Define a function compute_bill that takes one argument food as input. In the function, create a variable total with an initial value of zero. For each item in the food list, add the price of that item to total. Finally, return the total. Ignore whether the item you're billing for is in stock. Note that your function should work for any food list.
- Make the following changes to your compute_bill function:
  - While you loop through each item of food, only add the price of the item to total if the item's stock count is greater than zero.
  - If the item is in stock and after you add the price to the total, subtract one from the item's stock count.

Exercise 4

This exercise is a bit more complicate. We will review all about list and dictionaries. The aim of this exercise is to make a gradebook for teacher's students.

Try to follow the steps:

- Create three dictionaries: eren, mikasa, and armin.
- Give each dictionary the keys "name", "homework", "quizzes", and "tests". Have the "name" key be the name of the student (that is, eren’s name should be "Eren") and the other keys should be an empty list.
- Now copy this code:

```
eren = {
  "name": "Eren",
  "homework": [90.0,97.0,75.0,92.0],
  "quizzes": [88.0,40.0,94.0],
  "tests": [75.0,90.0]
}
mikasa = {
 "name": "Mikasa",
 "homework": [100.0, 92.0, 98.0, 100.0],
 "quizzes": [82.0, 83.0, 91.0],
 "tests": [89.0, 97.0]
}
```

```
armin = {
"name": "Armin",
"homework": [0.0, 87.0, 75.0, 22.0],
"quizzes": [0.0, 75.0, 78.0],
"tests": [100.0, 100.0]
}
```

- Below your code, create a list called students that contains eren, mikasa, and armin.
- for each student in your students list, print out that student's data, as follows:
    - print the student's name
    - print the student's homework
    - print the student's quizzes
    - print the student's tests
- Write a function average that takes a list of numbers and returns the average.
    - Define a function called average that has one argument, numbers.
    - Inside that function, call the built-in sum() function with the numbers list as a parameter. Store the result in a variable called total.
    - Use float() to convert total and store the result in total.
    - Divide total by the length of the numbers list. Use the built-in len() function to calculate that.
    - Return that result.
- Write a function called get_average that takes a student dictionary (like eren, mikasa, or armin) as input and returns his/her weighted average.
    - Define a function called get_average that takes one argument called student.
    - Make a variable homework that stores the average() of student["homework"].
    - Repeat step 2 for "quizzes" and "tests".
    - Multiply the 3 averages by their weights and return the sum of those three. Homework is 10%, quizzes are 30% and tests are 60%.
- Define a new function called get_letter_grade that has one argument called score. Expect score to be a number.
    - Inside your function, test score using a chain of if: / elif: / else: statements, like so:

    - If score is 90 or above: return "A"
    - Else if score is 80 or above: return "B"
    - Else if score is 70 or above: return "C"
    - Else if score is 60 or above: return "D"
    - Otherwise: return "F"
    - Finally, test your function. Call your get_letter_grade function with the result of get_average(lloyd). Print the resulting letter grade.

- Define a function called get_class_average that has one argument, students. You can expect students to be a list containing your three students.
    - First, make an empty list called results.
    - For each student item in the class list, calculate get_average(student) and then call results.append() with that result.
    - Finally, return the result of calling average() with results.
- Finally, print out the result of calling get_class_averagewith your students list. Your students should be [eren, mikasa, armin].
- Then, print the result of get_letter_grade for the class's average.