**Assignment Week 6**

Please do all assignments below with the following rules:

1. Please create a repository in your github account with name “Assignment Week 6”
2. Please add readme file in your repository that has information about your FullName and BinusianID
3. Please store each assignment below into each folder in your “Assignment Week 6” repository. So, it means there will be 9 different folders in your repository with name according to the assignment number
4. Please add my github account: “**bagzcode**” into your github repository as one of the collaborators in your repository. Failed to do this means I cannot access your repository and give the marks for your assignment
5. The deadline is next **Thursday, 29th of October 2020 at 15:00**. So, I am expecting no more update or push being recorded in your repository pass the deadline.
6. Please don’t wait until the deadline to push your answer into your repository and
7. You are allowed to discuss with friends in the WA group (discuss means no copy paste answer ya)
8. Don’t forget to give any necessary comments in your code.
9. Please always read the question carefully and try it first before you ask any questions to me or the others
10. And Have fun

**Assignment 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.

**Assignment 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.

**Assignment 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.

**Assignment 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.

**Assignment 5**

A hapax legomenon (often abbreviated to hapax) is a word which occurs only once in either the written record of a language, the works of an author, or in a single text. Define a function that given the file name of a text will return all its hapaxes. Make sure your program ignores capitalization. [open <http://www.gutenberg.org/> and download an e-book as plain text, use the file for texting your program]

**Assignment 6**

Write a program that given a text file will create a new text file in which all the lines from the original file are numbered from 1 to n (where n is the number of lines in the file).

**Assignment 7**

Write a program that will calculate the average word length of a text stored in a file (i.e the sum of all the lengths of the word tokens in the text, divided by the number of word tokens). [open <http://www.gutenberg.org/> and download an e-book as plain text, use the file for texting your program]

**Assignment 8**

A certain children’s game involves starting with a word in a particular category. Each participant in turn says a word, but that word must begin with the final letter of the previous word. Once a word has been given, it cannot be repeated. If an opponent cannot give a word in the category, they fall out of the game. For example, with "animals" as the category,

Child 1: dog

Child 2: goldfish

Child 1: hippopotamus

Child 2: snake

...

Your task in this exercise is as follows: Take the following selection of 70 English Pokemon names and generate the/a sequence with the highest possible number of Pokemon names where the subsequent name starts with the final letter of the preceding name. No Pokemon name is to be repeated.

audino bagon baltoy banette bidoof braviary bronzor carracosta charmeleon cresselia croagunk darmanitan deino emboar emolga exeggcute gabite girafarig gulpin haxorus heatmor heatran ivysaur jellicent jumpluff kangaskhan kricketune landorus ledyba loudred lumineon lunatone machamp magnezone mamoswine nosepass petilil pidgeotto pikachu pinsir poliwrath poochyena porygon2 porygonz registeel relicanth remoraid rufflet sableye scolipede scrafty seaking sealeo silcoon simisear snivy snorlax spoink starly tirtouga trapinch treecko tyrogue vigoroth vulpix wailord wartortle whismur wingull yamask

**Assignment 9**

A sentence splitter is a program capable of splitting a text into sentences. The standard set of heuristics for sentence splitting includes (but isn't limited to) the following rules:

Sentence boundaries occur at one of "." (periods), "?" or "!", except that.

Periods followed by whitespace followed by a lower-case letter are not sentence boundaries.

* + - 1. Periods followed by a digit with no intervening whitespace are not sentence boundaries.
      2. Periods followed by whitespace and then an upper-case letter but preceded by any of short list of titles are not sentence boundaries.
      3. Sample titles include Mr., Mrs., Dr., and so on.
      4. Periods internal to a sequence of letters with no adjacent whitespace are not sentence boundaries
      5. (for example, www.aptex.com, or e.g).
      6. Periods followed by certain kinds of punctuation (notably comma and more periods) are probably not sentence boundaries.

Your task here is to write a program that given the name of a text file can write its content with each sentence on a separate line. Test your program with the following short text:

Mr. Miyagi bought cheapsite.com for 1.5 million dollars, i.e. he paid a lot for it. Did he mind? Adam Jones Jr. thinks he didn't. In any case, this isn't true... Well, with a probability of .9 it isn't.

The result should be:

Mr. Miyagi bought cheapsite.com for 1.5 million dollars, i.e. he paid a lot for it.

Did he mind?

Adam Jones Jr. thinks he didn't.

In any case, this isn't true...

Well, with a probability of .9 it isn't.