IF100 – Spring 2019-2020
Take-Home Exam #3
Due April 29th, Wednesday, 23:55 (Sharp Deadline)

Introduction

The aim of this take-home exam is to practice on loops (for and while statements). The use of loops is due to the nature of the problem; that is, you cannot finish this take-home exam without using for or while statements.

Description

We all need to minimize the time we spend outside of our homes. One unavoidable need in all households is grocery shopping. A supermarket wants to help its customers by founding a program which estimates for how long the products that the customers buy for a household can meet their nutrition needs. This program should ask for the information about all the products that the customers are buying, like the names of the products, their count in the shopping cart, and the amount of fat, carbohydrate and proteins in each product. Besides, the program also needs the information about all the members of the household, like their gender, weight, height and age.

In this take-home exam, you will implement a Python program that will get 2 inputs from the user. These inputs have to be checked against a predefined format, which is further explained in the upcoming section. Then, your program should calculate the lifetime of each nutrient according to the total need of the household and display the calculated lifetime of each nutrient separately.

Your program will start with asking information about the products that the customer bought. The first input will include the products' name and their quantity, together with the amount of fat, carbohydrate and protein in each product. On the other hand, the second input will include each family member's gender, weight, height and age. Once these inputs are entered in the correct format, your program will calculate and display the lifetime of the shopping cart, for each type of nutrient. You can find the details about the inputs, calculation and outputs in the following section.

Input, Input Check, Calculation and Output

The inputs of the program and their order are explained below. It is extremely important to follow this order with the same format since we automatically process your programs. Also, prompts of the input statements to be used have to be exactly the same as the prompts of the "Sample Runs". ***Thus, your work will be graded as 0 unless the order is entirely correct***.

Your program will have multiple inputs and the specifications for these inputs are explained below.

- First input includes the products' name, their quantity, and the amount of fat, carbohydrate and protein in each product, in the following format:

  p1_quantity1_fat1_carbohydrate1_protein1,p2_quantity2_fat2_carbohydrate2_protein2,...pN_quantityN_fatN_carbohydrateN_proteinN

  - ***p1, p2, …, pN*** are the names of the products.
  - You may assume that each product's name will be given in the correct format. You may also assume that a particular product name will appear exactly once in the given input.
  - For each product, quantity shows how many of that product has been added to the chart. This information is given as ***quantityX***. You may assume that this information will also consist only of digits, the lowest value it can take is 1. Hence, you do not need to check these parts of the input.
  - For each product, its nutrient information for a single portion is given in grams as ***fatX_carbohydrateX_proteinX***. You may assume that all of the nutrient information will consist only of digits. These values can be equal to 0. However, they cannot be negative numbers. You do not need to perform an input check in this section.
  - Each product and its information should be separated by a single comma character ("**,**"). Your program should perform a check on this, as described below.
  - Product name, quantity, fat amount, carbohydrate amount and protein amount in the product information part of the input, they all should be separated by an underscore character ("**_**"). Your program should perform a check on this, as described below.
  - Your program should check the following for this input:
    - There should be at least 4 underscore characters ("**_**") in this input, and the total number of underscore characters ("**_**") should be a multiple of 4.

        - Total number of underscore characters ("**_**") should be equal to the total number of comma characters ("**,**") + 1 multiplied with 4.
            - Total number of underscore characters = (Total number of comma characters + 1) * 4
        - If the user enters an invalid input, then your program needs to display an error and <u>continue to ask for another input until the valid type of an input is entered by the user.</u>

    - You can assume that the product name will not contain any underscore characters or any commas.

- Second input includes each family member's gender, weight, height and age, in the following format:

  gender1_weight1_height1_age1,gender2_weight2_height2_age2,...genderN_weightN_heightN_ageN

    - Each family member should be separated by a single comma character ("**,**"). Your program should perform a check on this, as described below.
    - Family members' each characteristic information should be separated by an underscore character ("**_**"). Your program should perform a check on this, as described below.
    - ***weightX_heightX_ageX*** is the weight (in kg), height (in cm) and age information of a particular family member. You may assume that all of these characteristics will consist only of digits. Thus, you do not need to check these parts of input.
    - For the gender information "**M**" will be used for males and "**F**" will be used for females. You may assume that the gender information will always be entered in the correct and upper case format. So, you don't need to make any input checks for the gender information.

    - Your program should check the following for this input:
        - There should be at least 3 underscore characters ("**_**"), and the total number of underscore characters ("**_**") should be a multiple of 3.
        - Total number of underscore characters ("**_**") should be equal to the total number of comma characters ("**,**") + 1 multiplied with 3.
            - Total number of underscore characters = (Total number of comma characters + 1) * 3

    - If the user enters an invalid input, then your program needs to display an error and continue to ask for another input until the valid type of an input is entered by the user.

Once your program gets all the inputs correctly, it will calculate the total amount of fat, carbohydrate and protein in all the products bought. Thus, the nutrient amount in a portion should be multiplied with the quantity, and the total amount of carbohydrates, proteins and fats should be calculated separately.

For example; if a shopping cart consists of two (2) products, given as follows, ***p1_quantity1_fat1_carbohydrate1_protein1,p2_quantity2_fat2_carbohydrate2_protein2***, then the total amount of fat in grams should be calculated with the equation given below, and the equations for calculating the other types of nutrients would be very similar.

```
total_fat = quantity1 * fat1 + quantity2 * fat2
```

Every person, who is a member of the household in consideration, needs a different count of calories per day. According to The Harris-Benedict Equation for Basal Energy Expenditure (BEE), the calories that a person should take differs by gender. Please use the equations given below to calculate each household member's calorie needs **per day**.

Men: BEE (in calories) = 66.5 + 13.8*(Weight) + 5.0*(Height) - 6.8*(Age)

Women: BEE (in calories) = 655.1 + 9.6*(Weight) + 1.9*(Height) - 4.7*(Age)

Here are the other information that you need to know to calculate the amount of nutrients the family needs:

- A person's daily diet should consist of %50 carbohydrates, %30 fat and %20 proteins.
- 1 g fat is 9 calories.
- 1 g carbohydrates is 4 calories.
- 1 g protein is 4 calories.

Lastly, as mentioned before, your program needs to calculate for how many days each type of nutrient will be enough. If anyone of carbohydrate, fat or protein won't be enough for 1 day, then your program should print out a message which states that the user needs to add other products of that type of nutrient to the shopping cart. Otherwise, your program should print a message stating how many days they all will last. Please pay attention that your outputs should be exactly the same with the sample runs.

Please see the "Sample Runs" section for some examples.

Sample Runs

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are inputs taken from the user. <u>You have to display the required information in the same order and with the same words and characters as below.</u>

Sample Run 1

```
Products in shopping cart:
Cheese_2_13_135_130,rice_1_385_35_5,fish_2_0_130_40
Family members' informations: M_100_180_32,F_60_165_19
These products will be enough for your family in terms of fat for 3
days.
These products will be enough for your family in terms of carbohydrate
for 1 days.
These products will be enough for your family in terms of protein for
1 days.
```

Sample Run 2

```
Products in shopping cart: cheese1meat3
Invalid input for products, please enter in correct format.
Products in shopping cart: cheese1,meat3
Invalid input for products, please enter in correct format.
Products in shopping cart: cheese_1,meat_3
Invalid input for products, please enter in correct format.
Products in shopping cart: cheese_1_13_135_130fish_2_0_130_40
Invalid input for products, please enter in correct format.
Products in shopping cart: cheese_1_13_135_130,fish_2_0_130_40
Family members' informations: M_100_180_32,F_60_165_19
You need to add products with more fat to your shopping cart.
You need to add products with more carb to your shopping cart.
These products will be enough for your family in terms of protein for
1 days.
```

Sample Run 3

```
Products in shopping cart:
Cheese_2_13_135_130,rice_1_385_35_5,fish_2_0_130_40
Family members' informations: M_F
```

```
Invalid input for family information, please enter in correct format.
Family members' informations: M_12,F_60
Invalid input for family information, please enter in correct format.
Family members' informations: M,80,180,32_F,60,170,19
Invalid input for family information, please enter in correct format.
Family members' informations: M_100_180_32_44,F_60_165_19_36
Invalid input for family information, please enter in correct format.
Family members' informations: M_100_180_32,F_60_165_19
These products will be enough for your family in terms of fat for 3
days.
These products will be enough for your family in terms of carbohydrate
for 1 days.
These products will be enough for your family in terms of protein for
1 days.
```

Sample Run 4

```
Products in shopping cart:
Cheese_2_13_135_130,rice_1_385_35_5,fish_2_0_130_40,Meat_3_15_165_45
Family members' informations: M_80_179_44,F_55_164_46,F_15_80_2
These products will be enough for your family in terms of fat for 3
days.
These products will be enough for your family in terms of carbohydrate
for 2 days.
These products will be enough for your family in terms of protein for
2 days.
```

Sample Run 5

```
Products in shopping cart: cheese_1_13_135_130fish_2_0_130_40
Invalid input for products, please enter in correct format.
Products in shopping cart: cheese_1,bread_3
Invalid input for products, please enter in correct format.
Products in shopping cart: cheese_11_13_135_130,fish_13_0_130_40
Family members' informations: M_F
Invalid input for family information, please enter in correct format.
Family members' informations: M,80,180,32_F,60,170,19
Invalid input for family information, please enter in correct format.
Family members' informations: M_100_180_32,F_60_165_19
These products will be enough for your family in terms of fat for 1
days.
```

```
These products will be enough for your family in terms of carbohydrate
for 7 days.
These products will be enough for your family in terms of protein for
10 days.
```

Sample Run 6

```
Products in shopping cart: cheese_1_13_135_130
Family members' informations: M_100_180_32,F_60_165_19,M_75_180_24
You need to add products with more fat to your shopping cart.
You need to add products with more carb to your shopping cart.
You need to add products with more protein to your shopping cart.
```

How to get help?

You can use GradeChecker (http://sky.sabanciuniv.edu:8080/GradeChecker/) to check your expected grade. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

What and where to submit?

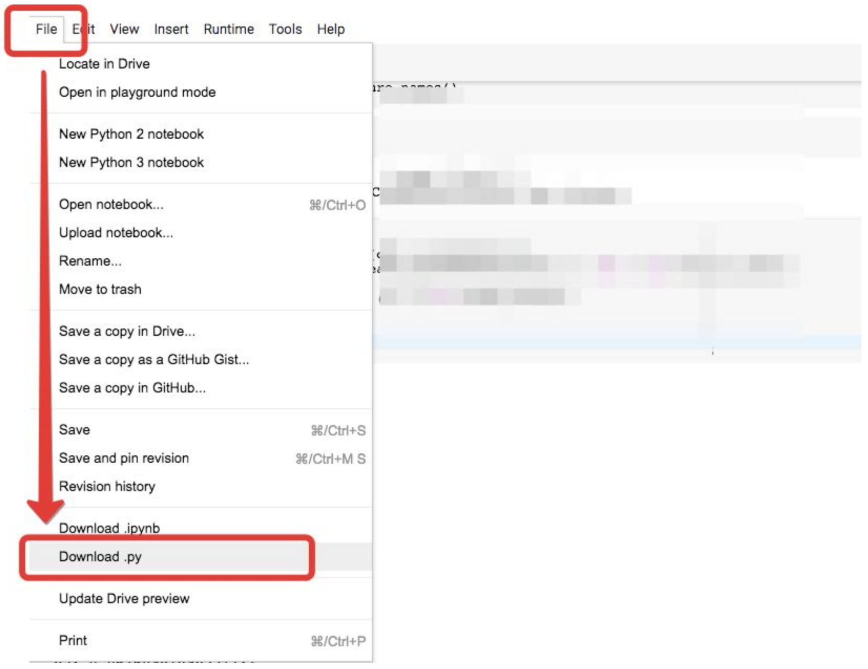
You should prepare (or at least test) your program using Python 3.x.x. We will use Python 3.x.x while testing your homework.

It'd be a good idea to write your name and lastname in the program (as a comment line of course). Do not use any Turkish characters anywhere in your code (not even in comment parts). If your name and last name is "İnanç Arın", and if you want to write it as comment; then you must type it as follows:

# Inanc Arin

Submission guidelines are below. Since the grading process will be automatic, students are expected to strictly follow these guidelines. If you do not follow these guidelines, your grade will be 0.

- Download your code as *py* file with "File" -> "*Download .py*" as below:



- Name your *py* file that contains your program as follows:

  "**username_the3.py**"

  For example: if your SuCourse username is "**duygukaltop**", then the name of the *py* file should be: **duygukaltop_the3.*py*** (please only use lowercase letters).

- Please make sure that this file is the latest version of your homework program.

- Submit your work **<u>through SUCourse only</u>**! You can use the GradeChecker only to see if your program can produce the correct outputs both in the correct order and in the correct format. It will not be considered as the official submission. You must submit your work to SUCourse.

- If you would like to resubmit your work, you should first remove the existing file(s). This step is very important. If you do not delete the old file(s), we will receive both files and the old one may be graded.

General Homework Rules

- Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.
- There is NO late submission. You need to submit your homework before the deadline. Please be careful that SUCourse time and your computer time may have a 1-2 minutes differences. You need to take this time difference into consideration.
- Do NOT submit your homework via email or in hardcopy! SUCourse is the only way that you can submit your homework.
- If your code does not work because of a syntax error, then we cannot grade it; and thus, your grade will be 0.
- Please do submit your **own** work only. It is really easy to find out "similar" programs!
- Plagiarism will not be tolerated. Please check our plagiarism policy given in the syllabus of the course.

Good luck!
Pınar Ön & Tunal Hamzaoğlu & IF100 Instructors