IF100 – Fall 2020-2021
Take-Home Exam #4
Due January 10, Sunday, 23:59 (Sharp Deadline)

Introduction

The aim of this take-home exam is to practice on loops (for/while statements), dictionaries, functions and file operations. The use of loops, functions and file operations are necessary due to the nature of the problem; that is, you cannot finish this take-home exam without using them. You are given a main program that uses some functions and for the program to work correctly you should implement those functions. You may implement the solution without using a dictionary, but that would be less efficient and much harder for you to code. Besides, beware that if we use a larger file that contains millions of lines in auto grading, your program may fail due to inefficiency if you use lists instead of a dictionary.

For this Take-Home Exam, we have shared an incomplete Google Colab file (***THE4.ipynb***) in THE4 folder with you that only contains the main part of the assignment. Please do not modify the last code cell which has the main part, instead use the previous code cell(s) to implement necessary functions. Note that a different main program with the same function names will be used for grading purposes. If you modify the one given with the take-home exam documentation, then your grade will be affected with respect to the severity of the modification carried out. Therefore, do **not** modify (do not add/delete anything) the given main program. However, you still need to upload the entire solution while submitting your take-home exam. Your submission should include the function implementations, and also the main program that is provided together with the take-home exam documentation.

Description

In this take home exam, you will implement a Python program that will be used as an inventory management system. Together with this description document, you will also be given a ***sample*** input file, that includes the inventory information. The initial amount of stock in the inventory will be read from this text file. Your program will use this file to create a dictionary of the inventory information with `import_inventory()` function and then your program will wait for an input from the user after suggesting 3 options, if the user enters an input other than the suggested options than the program will prompt an error message and ask for valid input. Option "1" will call `sell_product()` function which will recalculate the remaining stock in the inventory. Option "2" will call `show_remaining()` function to display the remaining stock in the inventory. Option "3" will terminate the program. So, your program will keep asking for option input until the user selects option 3.

You can find the details about the Input File, Functions, Inputs and Outputs in the following section.

Input File

You will be given only one input file sample which contains product names and product counts information in a certain format. The name of the input file is `products.txt`.

Each line of the input file will contain product information as in the following format;

$$p_1\_p_1count\text{-}p_2\_p_2count\text{-}\ldots\text{-}p_n\_p_ncount$$

Ex: water_2-chocolate_3-cracker_35

- Each product information will be separated with a dash ("-") character, and there will be an underscore ("_") character between the name of the product and the count of the product as a positive integer. You can assume the file information will be given in correct format. So, you don't have to perform any format check. You may assume that product names will not contain any "-" or "_" characters.
- You cannot make assumptions about the length of the product names, or the digit count in product count information.
- There can be any number of products on the same line
- Product names are case insensitive. It means that "wATer" and "WatER" are considered the same products.
- Please keep in mind that a product can occur several times in the same line of file or in different lines.
- You can assume that there will be no empty lines in the file. However, you cannot make any assumptions on the number of lines of this file. Besides, the lines might end or start with space, tab or other whitespace characters. Keep this in mind while you're preprocessing the content of the file.

For a sample input file, you may check out the `products.txt` provided with this assignment.

Please note that *we will use another file while grading* your take-home exams. The content of the file will most probably change for grading purposes, but the name of the file will remain the same; that is, it will always be "`products.txt`".

Functions, Inputs and Outputs

Since, you cannot change the main part of the program you will need to implement functions that are being used in the main part. The details about these functions 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***.

There are three functions that you definitely will need to implement in your program.

> ➢ import_inventory() is the function to be implemented to read the data from the text file into a dictionary that holds the counts of the products. Therefore, in this function, you will create an empty dictionary, read the data from *products.txt* into this dictionary, and then return the dictionary as a returning value.
>
> In the returning dictionary, you should keep product names as keys, and total product counts with this name as integer values.
>
> ➢ sell_product() is the function to be implemented to update the counts of the products in the inventory after a sell process. This function takes only one parameter which is the inventory dictionary that was created in `import_inventory()` function. `sell_product()` function does not return any returning value.
>
> In `sell_product()` function, the names of the products and the counts of the products to be sold will be taken as a string input. Product names are again case-insensitive, which means that some of the letters could be entered in upper-case, whereas some of them could be entered in lower-case. So, you need to normalize them before searching from the dictionary.
>
> The format of the products to be sold input will be as following;
>
> $$p_1\_p_1count\text{-}p_2\_p_2count\text{-}\ldots\text{-}p_n\_p_ncount$$
>
> **Ex: WatEr_1-ChocolatE_2-waTer_21**

- Each product information will be separated with a dash ("-") character and there will be an underscore ("_") character between the name of the product and the count of the product. You can assume the file information will be given in correct format. So, you don't have to perform any format check.

- You cannot make assumptions about the length of the product names, or the digit count in product count information.
- There can be any number of products on the same line
- Please keep in mind that a product can occur several times in this input. Your program should handle each product separately even if the same product occurs several times.
- You may assume that product counts will always be an integer greater than 0.
- After selling the product if the remaining count is zero, that product should be removed from the dictionary (hint: use `pop()` method of dictionary).

After the product information to sell input is taken, this function (`sell_product()`) should compare each product count in the input and the product count in the inventory. According to this comparison, if there are enough products in inventory, then your function should update the count in the inventory by decreasing the product count by the product to be sold, and give the count of the sold product and the name of the product (with lowercase letters) as output. For example; "**`10 water sold.`**"

If we have this particular product in inventory, but this product's count value is less than users request, then your program should output an appropriate error message saying there is not enough product (with lowercase letters) in inventory. For example; "**`There is not enough water in inventory.`**"

If there is not even one product of the product requested to be sold in inventory, then your program should output an appropriate error message saying that product (with lowercase letters) doesn't exist in inventory. For example; "**`water doesn't exist in inventory.`** "

➢ show_remaining() is the function to be implemented to print the current products (with lowercase letters) and their counts in inventory in alphabetical order (in an ascending way). If there are no products in the inventory, then your program should print an appropriate message; **`"Inventory is empty!"`**. `show_remaining()` function does not return any returning value.

<u>It's very important to have the same function names, same number of parameters, and same returning values for these functions as explained in THE document. Otherwise, the given main program will not work with your</u>

functions.

You may add additional functions if you need without changing the main part.

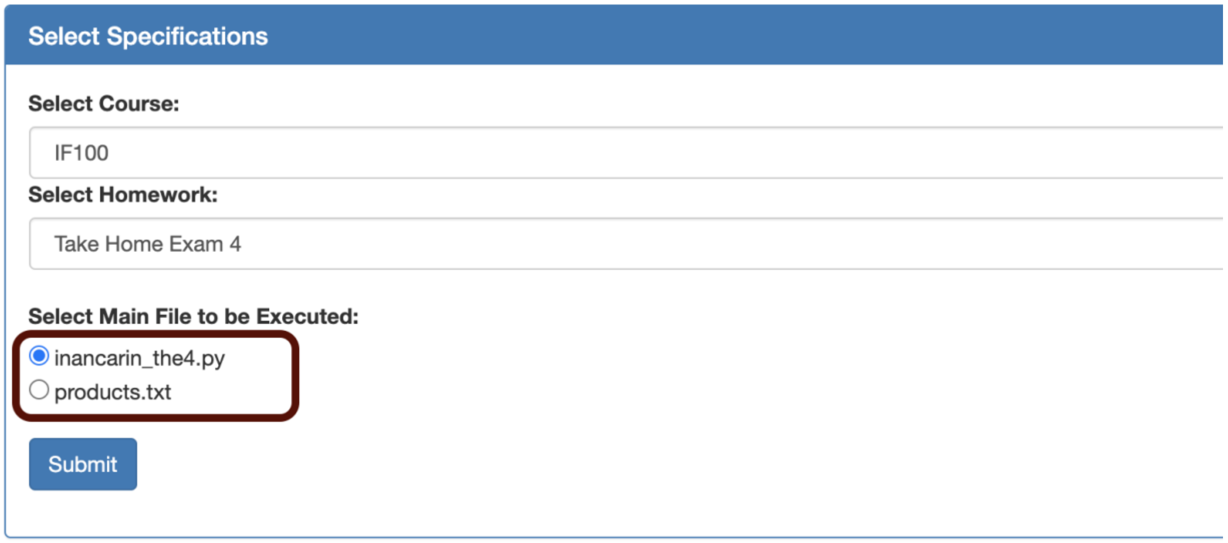
Please see the "Sample Runs" section for some examples.

Important Remarks on Working with Files

Please do NOT use the following two lines to upload the txt file to Colab. Instead use Colab's interface to upload your files as we showed in the lectures (if you use these two lines, then GradeChecker will not work with your solution):

```
from google.colab import files
uploaded = files.upload()
```

Another note on GradeChecker is that GradeChecker does not have the txt file(s) related to the take-home exams already. So, while uploading your code to GradeChecker, you should upload the required text file together with your .py file (2 files in total for this take-home exam: your python (.py) file *and* `products.txt`) and then select your .py file as the main file to be executed before pressing Submit button (see the image given below).



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. You have to display the required information in the same order and with the same words and characters as below.

Sample Run 1

```
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 3
----------------------------
Terminating...
```

Sample Run 2

```
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 4
----------------------------
Invalid input!
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 3
----------------------------
Terminating…
```

Sample Run 3

```
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 1
____________________________
Please enter products to sell: COKE_6-water_2
6 coke sold.
2 water sold.
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 3
____________________________
Terminating…
```

Sample Run 4

```
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 2
____________________________
candy : 321
cheese : 171
chips : 80
chocolate : 78
coke : 244
gum : 203
soda : 935
water : 219
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 3
____________________________
Terminating...
```

Sample Run 5

```
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 2
____________________________
candy : 321
cheese : 171
chips : 80
chocolate : 78
coke : 244
gum : 203
soda : 935
water : 219
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 1
____________________________
Please enter products to sell: COKE_6-water_2
6 coke sold.
2 water sold.
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 1
____________________________
Please enter products to sell: cracker_12-candy_5
cracker does not exist in inventory.
5 candy sold.
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 2
____________________________
candy : 316
```

```
cheese : 171
chips : 80
chocolate : 78
coke : 238
gum : 203
soda : 935
water : 217
*--------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 3
----------------------------
Terminating...
```

Sample Run 6

```
*--------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 2
----------------------------
candy : 321
cheese : 171
chips : 80
chocolate : 78
coke : 244
gum : 203
soda : 935
water : 219
*--------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 1
----------------------------
Please enter products to sell: SoDa_30-CHiPs_80-gum_33
30 soda sold.
80 chips sold.
33 gum sold.
```

```
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 1
----------------------------
Please enter products to sell: Gum_200-chocolate_12-waTer_4
There is not enough gum in inventory.
12 chocolate sold.
4 water sold.
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 2
----------------------------
candy : 321
cheese : 171
chocolate : 66
coke : 244
gum : 170
soda : 905
water : 215
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 1
----------------------------
Please enter products to sell: chips_5-candy_20-coke_200
chips does not exist in inventory.
20 candy sold.
200 coke sold.
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 3
----------------------------
Terminating...
```

Sample Run 7

```
*---------------------------

[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 1
----------------------------
Please enter products to sell: candy_321-cheese_171-chips_80
321 candy sold.
171 cheese sold.
80 chips sold.
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 1
----------------------------
Please enter products to sell: water_220-soda_935-chips_20-coke_244-
gum_203
There is not enough water in inventory.
935 soda sold.
chips does not exist in inventory.
244 coke sold.
203 gum sold.
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 1
----------------------------
Please enter products to sell: water_219-chocolate_78
219 water sold.
78 chocolate sold.
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 2
```

```
----------------------------
Inventory is empty!
*---------------------------
[1] Sell products
[2] Show remaining inventory
[3] Terminate
Please enter your decision: 3
----------------------------
Terminating…
```

How to get help?

You can use GradeChecker (https://learnt.sabanciuniv.edu/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?

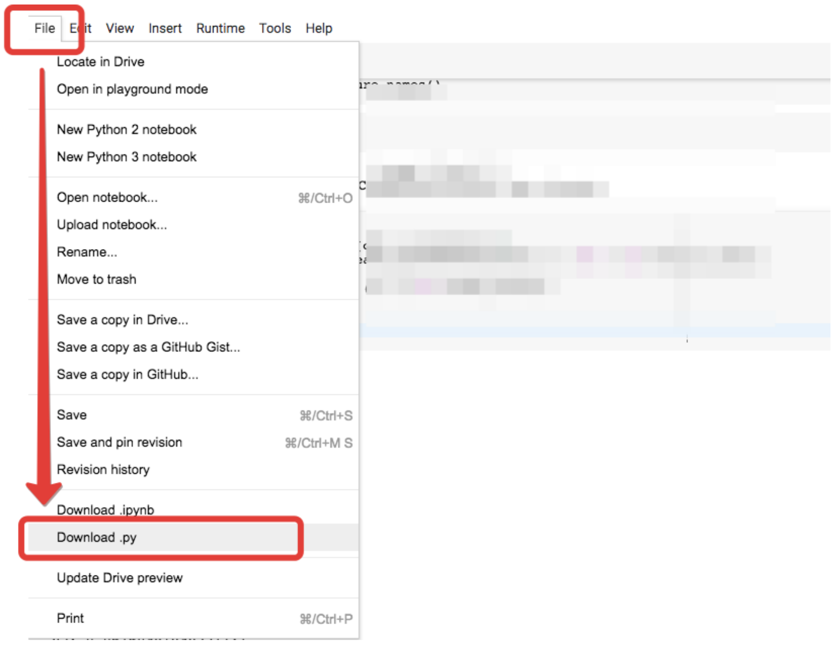
You should prepare (or at least test) your program using Python 3.x.x. We will use Python 3.x.x while testing your take-home exam.

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_the4.py"

  For example: if your SUCourse+ username is **"duygukaltop"**, then the name of the *py* file should be: **duygukaltop_the4.*py*** (please only use lowercase letters).

- Please make sure that this file is the latest version of your take-home exam program.

- You should also upload the txt file (product.txt) with your python file to GradeChecker, but uploading the txt file is not necessary for SUCourse+. You may or may not include txt file in your SUCourse submission.

- 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 Take-Home Exam Rules

- Successful submission is one of the requirements of the take-home exam. If for some reason, you cannot successfully submit your take-home exam and we cannot grade it, your grade will be 0.
- There is NO late submission. You need to submit your take-home exam before the deadline. Please be careful that SUCourse+ time and your computer time may have 1-2 minute(s) differences. You need to take this time difference into consideration.
- Do NOT submit your take-home exam via email or in hardcopy! SUCourse+ is the only way that you can submit your take-home exam.
- 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 "similar" programs!
- Plagiarism will not be tolerated. Please check our plagiarism policy given in the syllabus of the course.

Good luck!
Elif Pınar Ön, Ethem Tunal Hamzaoğlu
& IF100 Instructors