IF100 – Spring 2020-2021
Take-Home Exam 3
Due May 7th, 2021, Friday, 23:59 (Sharp Deadline)

Introduction

The aim of this take-home exam is to practice on loops (for/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 loop statements.

Description

SUInvest asked you to write a program to keep track of cryptocurrency wallets. Your program will calculate the current state of the wallet by taking the purchase history from its users, and at the end, it will display the wallet total in currency.

In this take-home exam, you will implement a Python program that will get three inputs from the user:

- The first input of your program is the purchase history: Exchange code of cryptocurrencies and amounts, written in a predefined format. Here, you should keep in mind that a cryptocurrency might occur several times in this input, in the case of which your program should sum up all the amounts declared that are associated with the same cryptocurrency. In this input, information of cryptocurrencies will be separated with semicolons.
- The second input of your program is the exchange code of the currency for which the user wants to display the wallet total.
- The third input is the exchange rates of the currencies. In this input, information of cryptocurrencies will be separated with semicolons.

Your program needs to print an appropriate output according to some conditions in your program.

- If there is no cryptocurrency in the wallet that the user wants to sell, then your program needs to print out a message accordingly.
- If there is not as much as cryptocurrency in the wallet that the user wants to sell, then your program needs to print out a message accordingly.

You can find the details about the inputs and outputs in the following section.

Input, Process 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 purchase history; exchange codes of cryptocurrencies and amounts of cryptocurrencies bought or sell in the following format:

  ExchangeCode#1:ExchangeCode#1amount;ExchangeCode#2:ExchangeCode#2amount;...;ExchangeCode#N:ExchangeCode#Namount

  You may assume that each exchange code will be given correctly and there will be no semicolon ("**;**") and colon ("**:**") characters used in the exchange codes. Keep in mind that the same exchange code might occur more than once in the given input.

  - You may assume that there will be only one semicolon ("**;**") between each exchange code and amount of information pairs and only one colon ("**:**") between the exchange codes and amounts of cryptocurrencies. Also, you may assume that all exchange codes will be given in all uppercase.
  - There will not be any semicolon or colon characters in the beginning or in the end of the input.
  - Amount of cryptocurrency information will be given in float type. Amounts could be both positive or negative. Positive numbers indicate buying operation where negative amounts indicate selling operation.
  - You don't need to perform any input checks for this input.
  - You may assume that ExchangeCodes will not contain semicolon or colon characters.
- The second input of your program is the exchange code of the currency for which the user wants to display the wallet total.
  - You may assume that this input will be given in all uppercase letters.
  - You may assume the exchange rate of all currencies in your wallet and the desired currency will be contained by the third input.
  - You don't need to perform any input checks for these inputs as well.

- The third input is the exchange rates of the currencies in the following format:

  ExchangeCode#1_ExchangeCode#2:ExchangeRateA;ExchangeCode#3_ExchangeCode#2:ExchangeRateB;ExchangeCode#2_ExchangeCode#4:ExchangeRateC;ExchangeCode#3_ExchangeCode#1:ExchangeRateD;...

  - You may assume that there will be only one semicolon ("**;**") between each exchange code and exchange rate pairs and only one colon ("**:**") between the exchange codes and exchange rates. Also, you may assume that all exchange codes will be given in all uppercase.
  - Each exchange code pair is separated with underscore ("**_**").
  - "***ExchangeCode#1_ExchangeCode#2:ExchangeRateA***" indicates value of 1 ExchangeCode#1 equals to ExchangeRateA times ExchangeCode#2. In other words,

    ExchangeCode#1 = ExchangeRateA * ExchangeCode#2
  - You may assume that this input will contain the values of all currencies in your wallet at the desired exchange rate, which means all the information you need to calculate the total price of the wallet in desired currency will be given by the user. But there may also be some exchange rates that you will not use while calculating the total price.
- You may assume that exchange rates will be given in float type as positive values.

Once your program gets the first input, it should process each transaction indicated in the input. If the float value is larger than 0, then it means a buying operation; thus an indicated amount of cryptocurrency should be added to the wallet. In this condition, your program should output;

"*ExchangeCode#1amount ExchangeCode#1* **is bought"**

If the float value is smaller than 0, then it means a selling operation; thus an indicated amount of cryptocurrency will be decreased from the wallet. In this condition, your program should output;

"*ExchangeCode#1amount ExchangeCode#1* **is sold"**

Keep in mind that the selling operation should only be carried out if there is enough cryptocurrency to be sold in the current wallet. In the case of less amount, your program should output;

"Not enough *ExchangeCode#1***"**

If the cryptocurrency to be sold doesn't exist in the current wallet, then your program should output;

"You don't have *ExchangeCode#1***"**

You should also keep in mind that the order of the operations are important, which means if there is not enough cryptocurrency to be sold in the current wallet, that operation should not affect the wallet. In such cases, your program should only give the output indicated above.

After processing all of the transactions, your program should get the second and third input in order to calculate the total price of your wallet in the desired type of currency. After calculating the total price, your program should output;

"You have *total_amount desired_currency***(s)."**

Total_amount must be displayed as a real number *with exactly two decimal places* (*Hint*: use ***format*** function that was explained in the recitation materials).

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

Sample Run 1

```
Please enter your purchase history: SUC:0.1;ASX:1.2;SUC:0.2;SUC:0.3;ETR:2
0.1 SUC bought
1.2 ASX bought
0.2 SUC bought
0.3 SUC bought
2.0 ETR bought
Please enter the currency type: EPA
Please enter prices: SUC_EPA:4;TSE_XETRA:0.012;EPA_ASX:0.05;ETR_EPA:40
You have 106.40 EPA(s).
```

Sample Run 2

```
Please enter your purchase history:
SUC:0.2;ETR:1.2;SUC:-0.1;SUC:0.3;ETR:2;ETR:-1.1
0.2 SUC bought
1.2 ETR bought
0.1 SUC sold
0.3 SUC bought
2.0 ETR bought
1.1 ETR sold
Please enter the currency type: EPA
Please enter prices: EPA_SUC:0.25;ASX_EPA:20;EPA_ETR:0.025;IRE_XETRA:0.03
You have 85.60 EPA(s).
```

Sample Run 3

```
Please enter your purchase history:
SUC:0.2;ETR:1.2;SUC:-0.3;SUC:0.3;ETR:2;ETR:-1.1
0.2 SUC bought
1.2 ETR bought
Not enough SUC
0.3 SUC bought
2.0 ETR bought
1.1 ETR sold
Please enter the currency type: EPA
Please enter prices: SUC_EPA:4;HIRE_XETRA:0.02;ASX_EPA:20;ETR_EPA:40
You have 86.00 EPA(s).
```

Sample Run 4

```
Please enter your purchase history:
SUC:-0.2;ETR:1.2;SUC:0.1;SUC:0.3;ETR:2;ETR:-1.1
You don't have SUC
1.2 ETR bought
0.1 SUC bought
0.3 SUC bought
2.0 ETR bought
1.1 ETR sold
Please enter the currency type: EPA
Please enter prices: EPA_SUC:0.25;EPA_ASX:0.05;EPA_ETR:0.025
You have 85.60 EPA(s).
```

Sample Run 5

```
Please enter your purchase history:
HIRE:0.2;IRE:1.2;TSE:23;HIRE:-0.3;HIRE:0.3;IRE:2;IRE:-1.1;TSE:31
0.2 HIRE bought
1.2 IRE bought
23.0 TSE bought
Not enough HIRE
0.3 HIRE bought
2.0 IRE bought
1.1 IRE sold
31.0 TSE bought
Please enter the currency type: XETRA
Please enter prices:
HIRE_XETRA:0.02;SUC_EPA:4;IRE_XETRA:0.03;TSE_XETRA:0.012
You have 0.72 XETRA(s).
```

Sample Run 6

```
Please enter your purchase history:
HIRE:0.3;IRE:-1.2;TSE:14;HIRE:-0.3;HIRE:0.3;IRE:2;IRE:-1.1;IRE:-1.1;TSE:-8
9
0.3 HIRE bought
You don't have IRE
14.0 TSE bought
0.3 HIRE sold
0.3 HIRE bought
2.0 IRE bought
1.1 IRE sold
Not enough IRE
Not enough TSE
Please enter the currency type: XETRA
Please enter prices:
HIRE_XETRA:0.02;EPA_SUC:0.25;IRE_XETRA:0.03;TSE_XETRA:0.012
You have 0.20 XETRA(s).
```

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?

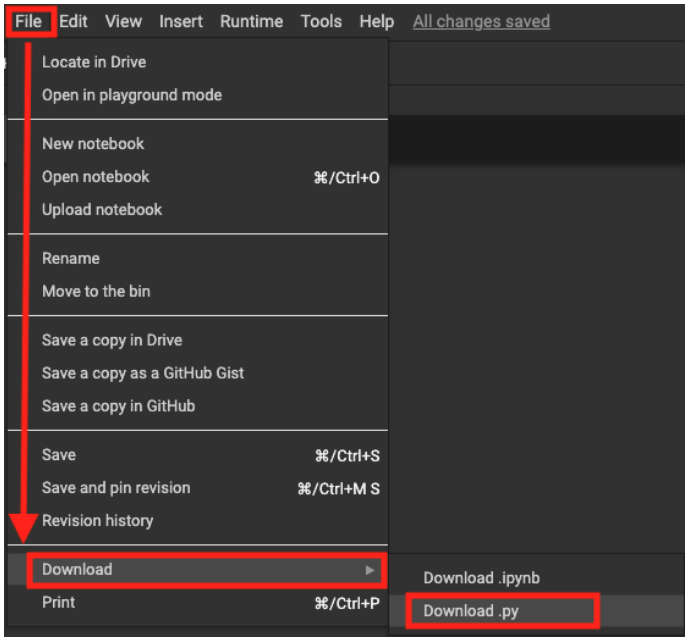
You should prepare (or at least test) your program using Python 3.x.x. We will use Python 3.x.x while testing your take-home exam.

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*" -> "*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 take-home exam program.
- Submit your work **through SUCourse+ only**! 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 minutes 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.
- ***You must use Google Colab while developing your solutions. Otherwise, you might be asked to have an interview with the instructors or we might not help when you object.***

Good luck!
Ethem Tunal Hamzaoğlu
& IF100 Instructors