**2.1 Brief & Synopsis**

In this second assessment, you are required to design, write, test and document software programs in Python to demonstrate your understanding and ability to apply the concepts presented at various stages of the term. Specifically, you are required to use Python 3 in this assessment.

**2.2 Instructions & Requirements**

In this assessment, you are required to develop an interactive program to be installed in a stamp-selling machine in Post Office. This machine sells postage stamps for various types of items (such as parcels and letters). Your program should provide two different functional suites, one for customers to purchase stamps; and the other for the system administrator, which allows different statistics to be performed based on the sales records.

Your software should sell stamps for the following six kinds of international post based on the price guide given in this assignment.

* Letter
* Economy Letter
* Express Letter
* Parcel
* Economy Parcel\_By Air
* Economy Parcel\_By Sea
* Standard Parcel
* Express Parcel

**Software inputs and output**

**Inputs** into your program will include:

* The price guides (as csv files) of the above mentioned six postage methods, as well as the country and zones mapping.
  + Economy Letters Price Guide ($).csv
  + Express Letter Price Guide ($).csv
  + Economy Parcel Price Guide\_by Sea ($).csv
  + Economy Parcel Price Guide\_by Air ($).csv
  + Standard Parcel Price Guide ($).csv
  + Express Parcel Price Guide ($).csv
  + Countries and Zones.csv

Each of these price guides will contain the price information for the postage method upon different item weights and destination zones. Note that these files should be loaded in a proper data structure when your software starts.

* The details of the items to be posted which are provided by users, including type (letter or parcel), weight, destination countries, postage methods, quantities etc. You will need to design a clear and interactive way to acquire inputs from users.
* The pre-stored sales history. You will be given a pre-stored sales history file “[sales\_history.](https://moodle.vle.monash.edu/pluginfile.php/6756432/mod_resource/content/1/sales_history.csv)csv”, which contains the history of stamp sales. Please refer to the section of Functional requirements regarding the format & description of this file.

**All the data files related to Assessment 2 could be downloaded from the bottom of this page.**

**Outputs** from your program include:

* Upon each customer completed his/her shopping, your program will need to:
  + Save the invoice along with the list of purchased stamps into a .txt file (refer to the section of Functional requirements).
  + Append the purchased items to the sales record “sales\_history.csv” (refer to the section of Functional requirements).
* Information and guidance to navigate and assist users during shopping.

**Functional Requirements**

Your program will need to provide two functional suites, one for stamp selling and one for the system administrator to view certain statistics on the selling records.

**Stamp-selling**

Your software program should provide the following functionalities as a stamp-selling machine.

* **[Fun 1]** Add an item to the shopping cart

Your program should allow the user to add items to his/her shopping cart. You will need to design an interactive way to acquire details of each item that he or she wants to purchase (such as a type of post, e.g., letter or parcel, its weight, destination country, a method of postage, quantity, etc.).

**[Challenge point]** Available methods and pricing information

You will need to provide proper guidance for customers during their shopping experience. For example, once a customer submits the detail of a parcel, e.g., weight, destination country, you should let know the available methods that they can choose (e.g., Standard, Express, etc.) and the associated pricing.

**[Challenge point]** Checking duplicate item entry

If the user add the same item multiple times (i.e., the same type, weight, destination country, and postage method), instead of adding a new item to the shopping cart,

* + Your program should update the quantity of the original item to be the sum of the quantities of these duplicate entries.

Note that each item in the shopping cart could contain multiple equivalent stamps, indicated by the item quantity.

* **[Fun 2]** Amend an item

Your program should allow the user to amend an item in the shopping cart, limited to its method of postage and quantity.

* **[Fun 3]** Remove an item from the shopping cart

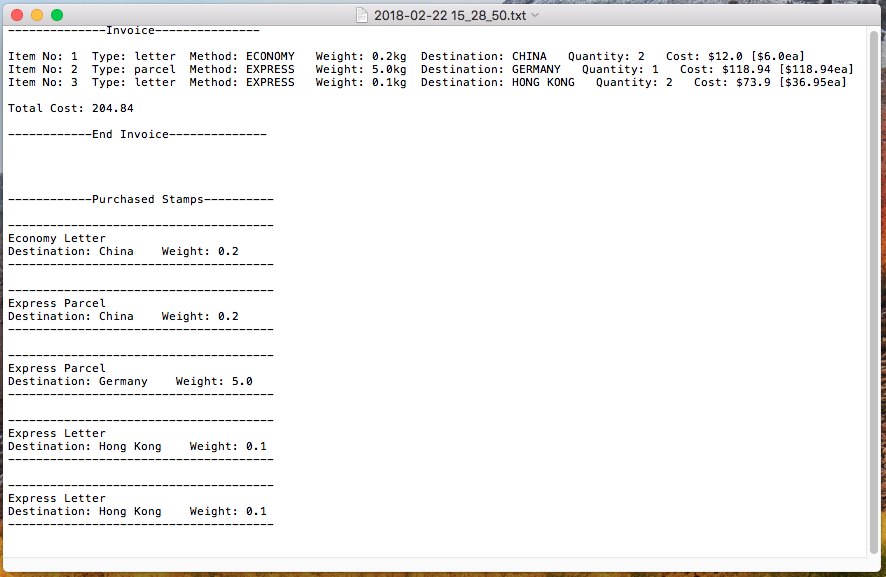
Your program should allow the user to delete an item in the shopping cart. It should ask the users to confirm the deletion before removing an item from the cart.

Upon the completion of adding, changing or removing an item, your program should update the shopping cart accordingly.

* **[Fun 4]**View shopping cart

Users should be able to view their shopping cart. Once requested, your program should print out the list of items & costs, as well as the total cost of the items in the shopping cart.

* **[Fun 5]** Checkout

Once a customer chooses to check out, the program should print out the invoice along with a list of purchased stamps to a .txt file.  This file should have a unique name, indicating the shopping date and time (see an example in the following figure). It is assumed that the list of stamps can be divided and used by the users in their post items. Then, staff in Post Office will check the stamps and arrange the delivery. This implies that if an item's quantity is larger than one, you should then print the stamps multiple times according to its quantity (refer to the following figure).

Feel free to modify the format of this file. However, you will need to make sure that the invoice and stamps are clear.

You will also need to **append** the record to the given “sales\_history.csv” file. Each record contains the information for each item sale as follows:

|  |  |
| --- | --- |
| **Field** | **Description** |
| sale\_id | this is a unique identifier for a sale record. You can choose to use an incremental key for the sale\_id, e.g., the id for the current sale = the id of the previous sale + 1.  Note that a sale only occurs when a customer successfully checks out. A sale (to a customer) could contain multiple items, but they are of the same sale\_id. |
| date\_time | date & time of the purchased |
| type | The type of the item, i.e., letter or parcel |
| weight | The weight of the item (in kg) |
| destination country | The destination country of where the item will be sent to |
| postage method | The method of delivery could be one of the following values: Economy Letter, Express Letter, Economy Parcel\_By Air, Economy Parcel\_By Sea, Standard Parcel, Express Parcel. |
| quantity | Number of stamps in purchased in this item. As previously mentioned, an item could contain multiple equivalent stamps. |
| cost | Total cost of the item, this would be equal to the unit price of the stamp times the item quantity |

Note that you should not change the name and format of this file.

Your software should support continuous shopping activities in Post Office. If you demand the next user to restart the program each time they do shopping, your program does not fulfil this requirement. However, you could assume that it serves only one customer at a time, i.e., the next customer will only start shopping after the previous customer checkouts or quits the program.

**System administration suite**

**[Fun 6]** Your software program should provide another functional suite for the system administrator, which allows the administrator to view the following analysis results based on the updated “sales\_history.csv” file:

* The gross sales amount of different years
* The customer flow during a day, e.g., you can compare the number of sales occur at different time (hours) of a day
* The popularity of postage method for letter and parcel, e.g., you could count the number of purchased stamps with different postage methods.
* The top 5 most popular destination countries including any postage.

Wherever suits, you should produce graphs or plots to present these results (You may want to consider the pandas and 'matplotlib' libraries in this part of the implementation.)

**Exception handling**

Your program should try to avoid or catch any kinds of exceptions and errors so that the execution won’t be interrupted and it won’t enter abnormal states.

When handling exceptions related to user inputs, your system should return meaningful error messages so that the users can correct their inputs.

Examples of these errors are **(but not limited to)**:

* User input errors, e.g., option entered are invalid, invalid country name, invalid weight, etc.
* If the shopping cart to be paid has zero cost, etc.

**Design Requirements**

The software could be implemented either as a procedural program or an object-oriented program.

However, you will need to design proper abstract data types wherever suits. This includes defining meaningful methods and instance variables in the classes, such that the classes can be reusable in similar projects.

Also, in this assessment, there is no hard requirement on what python data types to use, but you will consider the appropriate data types for attributes and arguments.

**Developing environment and settings**

To allow the assessors to complete marking of the assessments effectively, please make sure that:

* You don’t use an absolute path in your program. The input files should be read from the current directory ( .\ ), and output files are also written to the same directory. Although, you are welcome to create subfolders to separate input (price data), output and source (code) files.
* You will need to include a simple readme file to illustrate how to run your program. In the readme file, you will also need to state what functionalities you’ve implemented in your program, and what are outstanding.
* Submit electronic copies of ALL your files that are needed to run your programs.