"Shopping Cart" Project

> Prerequisites:
>
> - Python Language Overview
> - "Groceries" Exercise
> - "List Comprehensions" Exercise
> - The `datetime` Module
> - Raising and Handling Errors

Learning Objectives

1. Create a tool to facilitate and streamline a real-world business process.
2. Practice processing and validating user inputs in Python.
3. Reinforce introductory Python programming language concepts such as datatypes, functions, variables, and loops.
4. Practice incorporating version control into your development process.

Business Prompt

Your local corner grocery store has hired you as a technology consultant to help modernize their checkout system.

Currently, when managing inventory, store employees affix a price tag sticker on each grocery item in stock. And when a customer visits the checkout counter with their selected items, a checkout clerk uses a calculator to add product prices, calculate tax, and calculate the total amount due.

Instead, the store owner describes a desired checkout process which involves the checkout clerk scanning each product's barcode to automatically lookup prices, perform tax and total calculations, and print a customer receipt. To facilitate this process, the store owner has authorized the purchase of a few inexpensive barcode scanners, as well as checkout computers capable of running Python applications.

The store owner says it would be "acceptable but not preferable" to manage the inventory of products via the application's source code, that it would be "better" to manage the inventory of products via a local CSV file stored on the checkout computer, and that it would be "ideal" to be able to manage the inventory of products via a centralized Google Sheet spreadsheet document.

The store owner also says it would be "nice to have" a feature which prompts the checkout clerk or the customer to input the customer's email address in order to send them a receipt via email.

Instructions

Iteratively develop a Python application which satisfies the store owner's objectives, as described in more detail by the "Basic Requirements" below.

Before attempting to implement the basic requirements, take some time to configure your project repository according to the "Setup" instructions below. After doing so, you'll have a remote repo on GitHub.com and a local copy on your computer within which to develop.

When developing, as you reach key milestones, use the command-line or GitHub Desktop software to intermittently "commit", or save new versions of, your code. And remember to push / sync / upload your work back up to your remote project repository on GitHub.com at least once before you're done.

If you are able to implement the basic requirements with relative ease, or if you are interested in a challenge, consider addressing one or more of the "Further Exploration Challenges". Otherwise, if you need help breaking the problem up into more manageable pieces, consult the "Guided Checkpoints". And if you would like a narrated walkthrough, consult the "Guided Screencast".

Setup

Repo Setup

Use the GitHub.com online interface to create a new remote project repository called something like "shopping-cart". When prompted by the GitHub.com online interface, let's get in the habit of adding a "README.md" file and a Python-flavored ".gitignore" file (and also optionally a "LICENSE") during the repo creation process. After this process is complete, you should be able to view the repo on GitHub.com at an address like `https://github.com/YOUR_USERNAME/shopping-cart`.

After creating the remote repo, use GitHub Desktop software or the command-line to download or "clone" it onto your computer. Choose a familiar download location like the Desktop.

After cloning the repo, navigate there from the command-line:

```
cd ~/Desktop/shopping-cart
```

Use your text editor or the command-line to create a file in that repo called "shopping_cart.py", and then place the following contents inside:

```
# shopping_cart.py

products = [
    {"id":1, "name": "Chocolate Sandwich Cookies", "department": "snacks",
"aisle": "cookies cakes", "price": 3.50},
    {"id":2, "name": "All-Seasons Salt", "department": "pantry", "aisle":
"spices seasonings", "price": 4.99},
    {"id":3, "name": "Robust Golden Unsweetened Oolong Tea", "department":
"beverages", "aisle": "tea", "price": 2.49},
    {"id":4, "name": "Smart Ones Classic Favorites Mini Rigatoni With Vodka
Cream Sauce", "department": "frozen", "aisle": "frozen meals", "price":
6.99},
    {"id":5, "name": "Green Chile Anytime Sauce", "department": "pantry",
"aisle": "marinades meat preparation", "price": 7.99},
    {"id":6, "name": "Dry Nose Oil", "department": "personal care", "aisle":
"cold flu allergy", "price": 21.99},
    {"id":7, "name": "Pure Coconut Water With Orange", "department":
"beverages", "aisle": "juice nectars", "price": 3.50},
    {"id":8, "name": "Cut Russet Potatoes Steam N' Mash", "department":
"frozen", "aisle": "frozen produce", "price": 4.25},
    {"id":9, "name": "Light Strawberry Blueberry Yogurt", "department":
"dairy eggs", "aisle": "yogurt", "price": 6.50},
    {"id":10, "name": "Sparkling Orange Juice & Prickly Pear Beverage",
"department": "beverages", "aisle": "water seltzer sparkling water", "price":
2.99},
    {"id":11, "name": "Peach Mango Juice", "department": "beverages",
"aisle": "refrigerated", "price": 1.99},
    {"id":12, "name": "Chocolate Fudge Layer Cake", "department": "frozen",
"aisle": "frozen dessert", "price": 18.50},
    {"id":13, "name": "Saline Nasal Mist", "department": "personal care",
"aisle": "cold flu allergy", "price": 16.00},
    {"id":14, "name": "Fresh Scent Dishwasher Cleaner", "department":
"household", "aisle": "dish detergents", "price": 4.99},
    {"id":15, "name": "Overnight Diapers Size 6", "department": "babies",
"aisle": "diapers wipes", "price": 25.50},
    {"id":16, "name": "Mint Chocolate Flavored Syrup", "department":
"snacks", "aisle": "ice cream toppings", "price": 4.50},
    {"id":17, "name": "Rendered Duck Fat", "department": "meat seafood",
"aisle": "poultry counter", "price": 9.99},
    {"id":18, "name": "Pizza for One Suprema Frozen Pizza", "department":
"frozen", "aisle": "frozen pizza", "price": 12.50},
    {"id":19, "name": "Gluten Free Quinoa Three Cheese & Mushroom Blend",
"department": "dry goods pasta", "aisle": "grains rice dried goods", "price":
3.99},
    {"id":20, "name": "Pomegranate Cranberry & Aloe Vera Enrich Drink",
"department": "beverages", "aisle": "juice nectars", "price": 4.25}
] # based on data from Instacart: https://www.instacart.com/datasets/grocery-
shopping-2017

def to_usd(my_price):
    """
    Converts a numeric value to usd-formatted string, for printing and
display purposes.
    Source: https://github.com/prof-rossetti/intro-to-
python/blob/master/notes/python/datatypes/numbers.md#formatting-as-currency
    Param: my_price (int or float) like 4000.444444
    Example: to_usd(4000.444444)
    Returns: $4,000.44
    """
    return f"${my_price:,.2f}" #> $12,000.71

# TODO: write some Python code here to produce the desired output
```

```
print(products)
```

Make sure to save Python files like this whenever you're done editing them. After setting up a virtual environment, we will be ready to run this file.

Environment Setup

Create and activate a new Anaconda virtual environment:

```
conda create -n shopping-env python=3.7 # (first time only)
conda activate shopping-env
```

From within the virtual environment, demonstrate your ability to run the Python script from the command-line:

```
python shopping_cart.py
```

If you see the provided "products" data structure, you're ready to move on to project development. This would be a great time to make any desired modifications to your project's "README.md" file (like adding instructions for how to setup and run the app like you've just done), and then make your first commit, with a message like "Setup the repo".

Data Setup

The provided code includes a variable called `products` which facilitates management of the products inventory from within the application's source code.

> FURTHER EXPLORATION: If you'd like to manage the products inventory via a CSV file instead, download the provided "products.csv" file and place it into your project directory in a directory called "data", and use The `csv` Module or The `pandas` Package for CSV file management.

> FURTHER EXPLORATION: If you'd like to manage the products inventory via Google Sheet document instead, see the Integrating with a Google Sheets Datastore challenge for more info.

Basic Requirements

Write a program that asks the user to input one or more product identifiers, then looks up the prices for each, then prints an itemized customer receipt including the total amount owed.

The program should use one of the provided datastores (see "Data Setup") to represent the store owner's inventory of products and prices.

The program should prompt the checkout clerk to input the identifier of each shopping cart item, one at a time.

When the clerk inputs a product identifier, the program should validate it, displaying a helpful message like "Hey, are you sure that product identifier is correct? Please try again!" if there are no products matching the given identifier.

At any time the clerk should be able to indicate there are no more shopping cart items by inputting the word `DONE` or otherwise indicating they are done with the process.

After the clerk indicates there are no more items, the program should print a custom receipt on the screen. The receipt should include the following components:

- A grocery store name of your choice
- A grocery store phone number and/or website URL and/or address of choice
- The date and time of the beginning of the checkout process, formatted in a human-friendly way (e.g. `2020-02-07 03:54 PM`)
- The name and price of each shopping cart item, price being formatted as US dollars and cents (e.g. `$3.50`, etc.)
- The total cost of all shopping cart items (i.e. the "subtotal"), formatted as US dollars and cents (e.g. `$19.47`), calculated as the sum of their prices
- The amount of tax owed (e.g. `$1.70`), calculated by multiplying the total cost by a New York City sales tax rate of 8.75% (for the purposes of this project, groceries are not exempt from sales tax)
- The total amount owed, formatted as US dollars and cents (e.g. `$21.17`), calculated by adding together the amount of tax owed plus the total cost of all shopping cart items
- A friendly message thanking the customer and/or encouraging the customer to shop again

The program should be able to process multiple shopping cart items of the same kind, but need not display any groupings or aggregations of those items (although it may optionally do so).

Example Output

```
(shopping-env)  --->> python shopping_cart.py
Please input a product identifier: 1
Please input a product identifier: 2
Please input a product identifier: 3
Please input a product identifier: 2
Please input a product identifier: 1
Please input a product identifier: DONE
#> ---------------------------------
#> GREEN FOODS GROCERY
#> WWW.GREEN-FOODS-GROCERY.COM
#> ---------------------------------
#> CHECKOUT AT: 2020-02-07 03:54 PM
#> ---------------------------------
#> SELECTED PRODUCTS:
#>  ... Chocolate Sandwich Cookies ($3.50)
#>  ... All-Seasons Salt ($4.99)
#>  ... Robust Golden Unsweetened Oolong Tea ($2.49)
#>  ... All-Seasons Salt ($4.99)
#>  ... Chocolate Sandwich Cookies ($3.50)
#> ---------------------------------
#> SUBTOTAL: $19.47
#> TAX: $1.70
#> TOTAL: $21.17
#> ---------------------------------
#> THANKS, SEE YOU AGAIN SOON!
#> ---------------------------------
```

Guided Checkpoints

Feel free but not obligated to follow these guided "checkpoints", which provide one example strategy for breaking-up the requirements into smaller, more manageable pieces.

Guided Screencast

For a more in-depth guided exercise walkthrough, feel free but not obligated to follow the screencast, but keep in mind a few caveats:

1. Some of the links reference a previous course repository, but you should be able to find related documents in this course repository as well
2. If there is any discrepancy between requirements referenced in the video and requirements stated in this document, defer to the requirements stated in this document

Further Exploration Challenges

If you are able to implement the basic requirements with relative ease, consider addressing one or more of these "Further Exploration Challenges" to enrich and expand your learning experience.

Evaluation

Project submissions will be evaluated according to the requirements set forth above, as summarized by the rubric below:

Category	Requirement	Weight
Info Inputs	Captures / scans product identifiers	10%
Info Inputs	Handles invalid inputs	10%
Info Inputs	Handles the "DONE" signal	10%
Info Outputs (Receipt)	Displays store info	10%
Info Outputs (Receipt)	Displays checkout date and time	10%

Category	Requirement	Weight
Info Outputs (Receipt)	Displays names and prices of all scanned products	15%
Info Outputs (Receipt)	Displays tax and totals	15%
Dev Process	Submitted via Git repository which reflects an incremental revision history	20%

If experiencing execution error(s) while evaluating the application's required functionality, evaluators are advised to reduce the project's grade by between 4% and 25%, depending on the circumstances and severity of the error(s).

In recognition of deliverables which exhibit functionality above and beyond the basic requirements, evaluators are encouraged to award between 4% and 15% "engagement points" to be applied as extra credit.