[Sample Inputs 3](#_Toc133613939)

[Input.csv 3](#_Toc133613940)

[Output 5](#_Toc133613941)

[Screen print 5](#_Toc133613942)

[Invoice folder 8](#_Toc133613943)

[Detail in Invoice 9](#_Toc133613944)

[Audited file 11](#_Toc133613945)

[List of programs 11](#_Toc133613946)

[CostVerPro.py 11](#_Toc133613947)

[calculate\_delivery\_fee(sub\_total) 11](#_Toc133613948)

[calculate\_total\_cost(sub\_total, discount1, discount2, delivery\_fee) 11](#_Toc133613949)

[cost\_verification(order\_number, df, outputFile, index) 12](#_Toc133613950)

[mallsDollar.py 12](#_Toc133613951)

[calculate\_mall\_dollars(total:float) 12](#_Toc133613952)

[determineNextOrderNumber.py 12](#_Toc133613953)

[get\_next\_order\_number(last\_order\_number) 12](#_Toc133613954)

[extract\_data.py 13](#_Toc133613955)

[convert\_order\_csv\_to\_json() 13](#_Toc133613956)

[get\_next\_order\_number(order\_number) 13](#_Toc133613957)

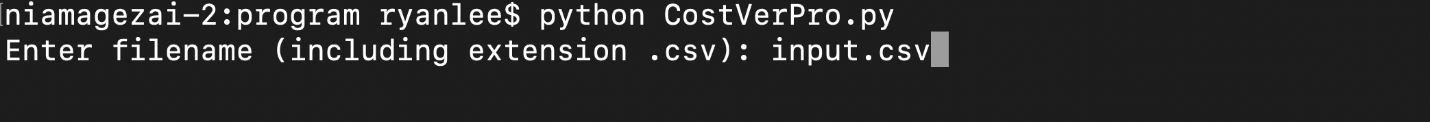
[checkDigit.py 13](#_Toc133613958)

[calculate\_check\_digit 13](#_Toc133613959)

[nameAdress.py 14](#_Toc133613960)

[convertorder.py 14](#_Toc133613961)

Sample Inputs



## Input.csv

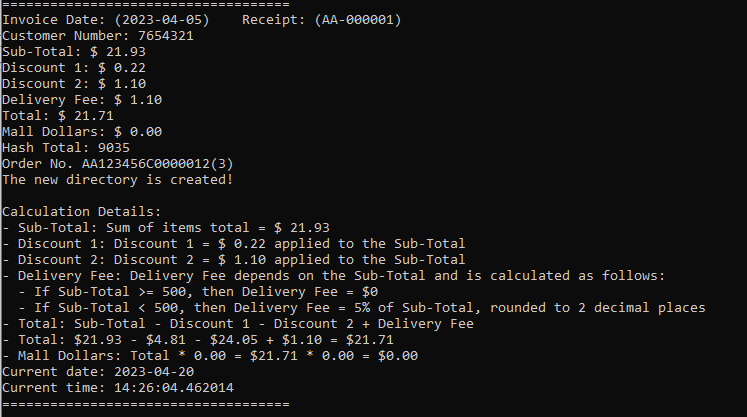
一張含有 資料表 的圖片

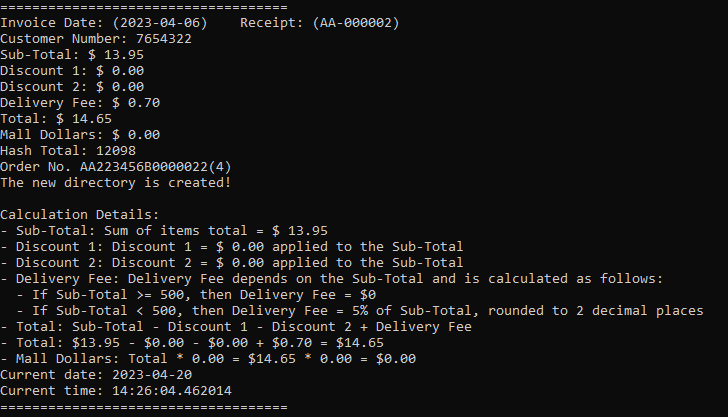
自動產生的描述

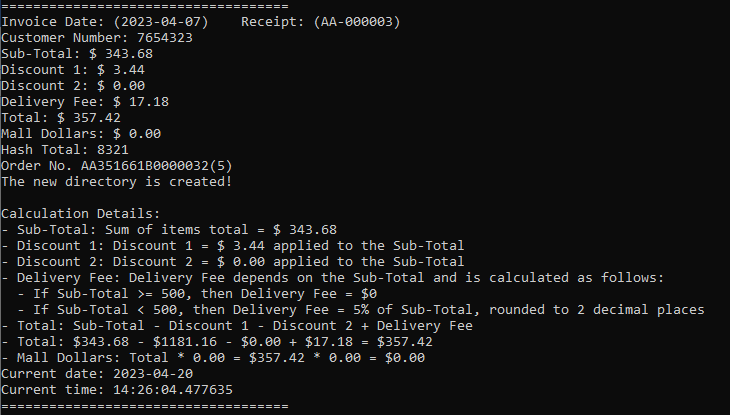
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| staff\_number | item\_number | item\_name | quantity | cost | discounts\_VIP | discounts\_VIPDay95 | discounts\_VIPDay123 | customer\_number | order |
| 123456 | 4018 | Pineapple | 3 | 1.99 | TRUE | 0.01 | 0.05 | 7654321 | 1 |
| 123456 | 5017 | Watermelon | 4 | 3.99 | TRUE | 0.01 | 0.05 | 7654321 | 1 |
| 223456 | 6014 | Strawberry | 3 | 1.99 | FALSE | 0 | 0 | 7654322 | 2 |
| 223456 | 6084 | Blueberry | 2 | 3.99 | FALSE | 0 | 0 | 7654322 | 2 |
| 351661 | 1121 | MetaSpeed Shampoo | 8 | 27 | TRUE | 0.01 | 0 | 7654323 | 3 |
| 351661 | 7200 | Papaya | 32 | 3.99 | TRUE | 0.01 | 0 | 7654323 | 3 |
| 456321 | 7400 | Peach | 43 | 1.99 | FALSE | 0 | 0 | 7654324 | 4 |
| 456321 | 7600 | Pear | 12 | 3.99 | FALSE | 0 | 0 | 7654324 | 4 |
| 534569 | 7800 | Grape | 3 | 1.99 | FALSE | 0 | 0 | 7654325 | 5 |
| 534569 | 7900 | Cherry | 2 | 3.99 | FALSE | 0 | 0 | 7654325 | 5 |
| 633456 | 8000 | Raspberry | 6 | 1.99 | FALSE | 0 | 0 | 7654326 | 6 |
| 633456 | 8100 | Blackberry | 2 | 3.99 | FALSE | 0 | 0 | 7654326 | 6 |
| 734567 | 4018 | KF-95 Anti-Cpvid Mask | 6 | 280 | FALSE | 0 | 0 | 7654327 | 7 |
| 734567 | 8300 | Lime | 2 | 3.99 | FALSE | 0 | 0 | 7654327 | 7 |
| 878987 | 8400 | Orange | 30 | 1.99 | FALSE | 0 | 0 | 7654328 | 8 |
| 878987 | 8500 | Banana | 1 | 3.99 | FALSE | 0 | 0 | 7654328 | 8 |
| 999999 | 8200 | Lemon | 3 | 1.99 | TRUE | 0 | 0.04 | 7654329 | 9 |
| 999999 | 8300 | Lime | 2 | 3.99 | TRUE | 0 | 0.04 | 7654329 | 9 |
| 999999 | 8600 | Apple | 3 | 1.99 | TRUE | 0 | 0.04 | 7654329 | 9 |
| 999999 | 8700 | Pineapple | 2 | 3.99 | TRUE | 0 | 0.04 | 7654329 | 9 |
| 999999 | 6084 | Blueberry | 200 | 3.99 | TRUE | 0 | 0.04 | 7654329 | 9 |
| 999999 | 8000 | Raspberry | 3 | 1.99 | TRUE | 0 | 0.04 | 7654329 | 9 |
| 999999 | 1121 | MetaSpeed Shampoo | 2 | 27 | TRUE | 0 | 0.04 | 7654329 | 9 |
| 999999 | 2134 | BTM Baby Powder | 5 | 198 | TRUE | 0 | 0.04 | 7654329 | 9 |
| 999999 | 3019 | Covid-19 Test kit | 8 | 52.9 | TRUE | 0 | 0.04 | 7654329 | 9 |

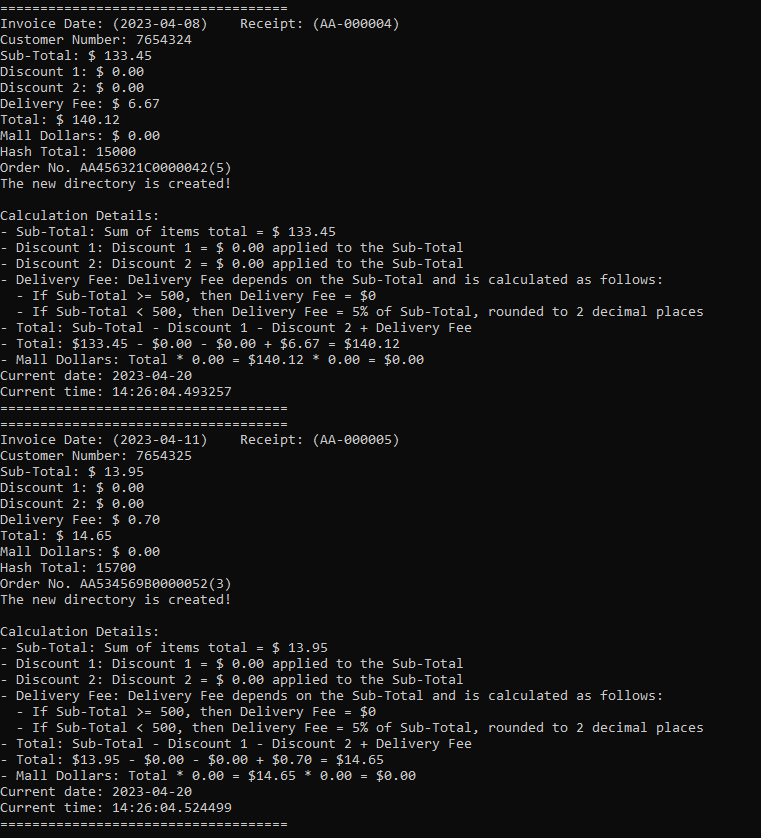
# Output

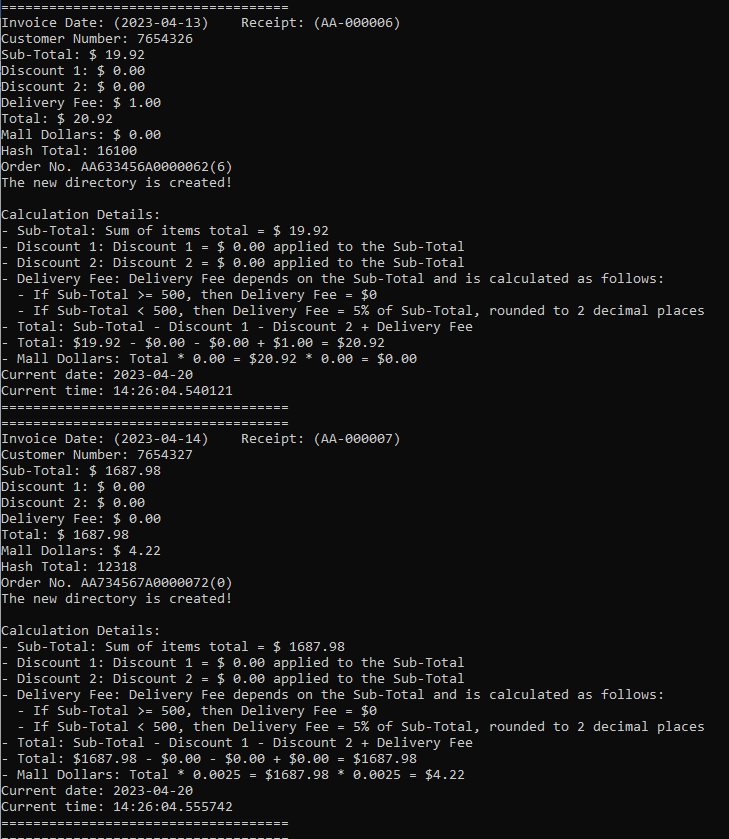
Screen print

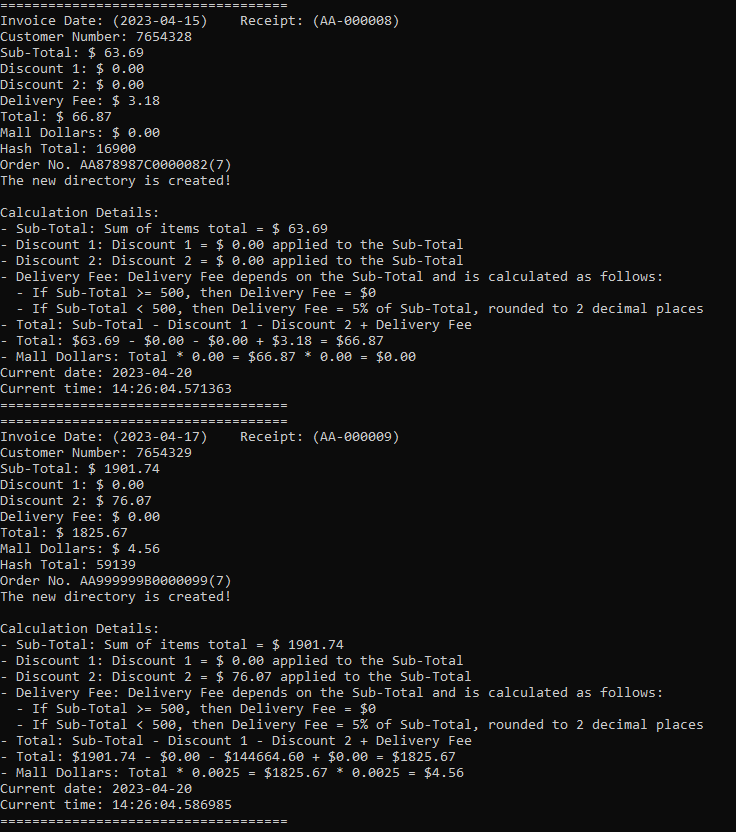


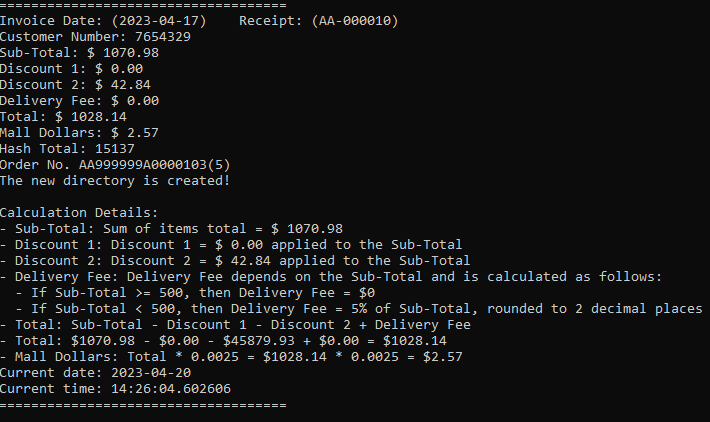




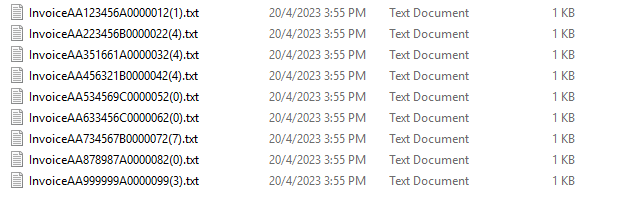




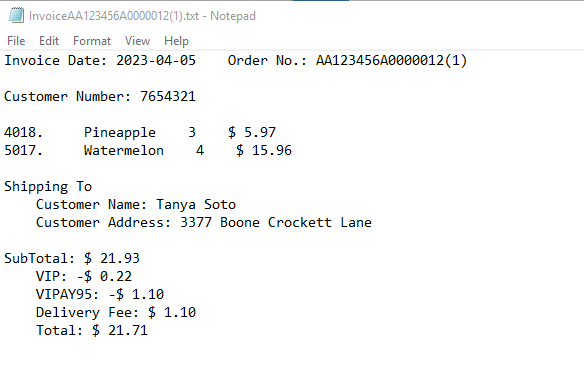


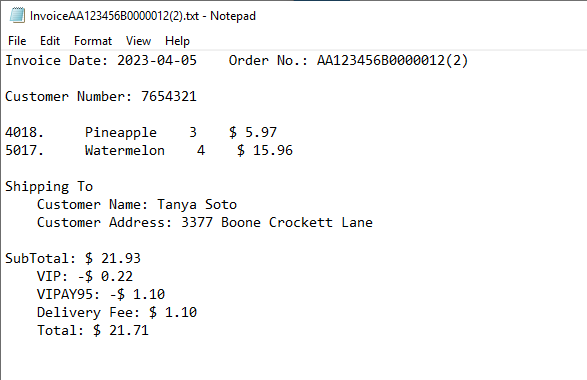


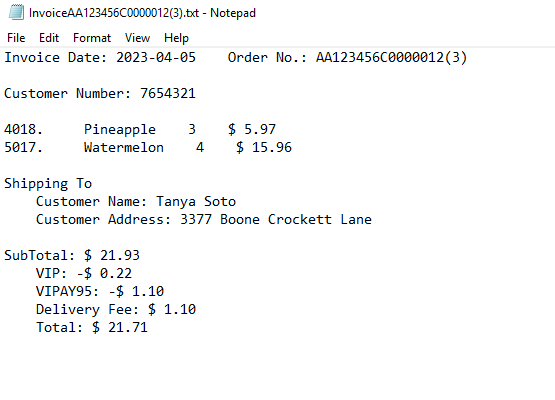
Invoice folder



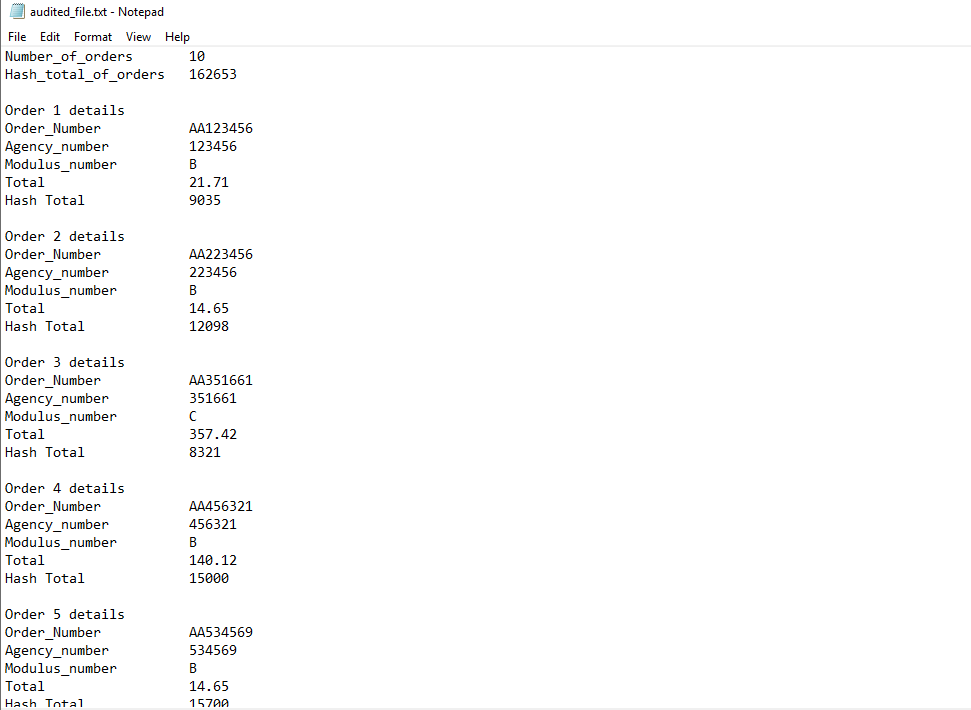
### Detail in Invoice







## Audited file



# List of programs

## CostVerPro.py

calculate\_delivery\_fee(sub\_total)

This function takes the subtotal of an order and returns the delivery fee. If the subtotal is equal to or greater than $500, then the delivery fee is 0. Otherwise, the delivery fee is calculated as 5% of the subtotal and rounded to 2 decimal places.

### calculate\_total\_cost(sub\_total, discount1, discount2, delivery\_fee)

This function takes the subtotal, two discount rates, and the delivery fee of an order, and returns the total cost. The total cost is calculated by subtracting the discounts and adding the delivery fee to the subtotal, rounded to 2 decimal places.

### cost\_verification(order\_number, df, outputFile, index)

This is the main function for cost verification. It takes the order number, a DataFrame containing the order details, an output file, and an index as inputs. It first filters the DataFrame to include only the rows with the matching order number and calculates the subtotal, discounts, and delivery fee using the previous two functions. It then calculates the total cost and the mall dollars earned, using another function calculate\_mall\_dollars(), and generates an order number using a random module. It also calculates the check digit and writes the order details and invoice to the output file. Finally, it prints the receipt and creates a new directory to store the invoice.

## mallsDollar.py

### calculate\_mall\_dollars(total:float)

This function takes a single argument total, which is a float representing the total amount spent by a customer at a mall. The function returns a float value representing the amount of "mall dollars" earned by the customer based on their total spending.

The calculation of mall dollars is based on the following criteria:

If the total amount spent is between 500 and 999 (inclusive), the customer earns mall dollars equal to 0.2% (0.002) of their total spending.

If the total amount spent is 1000 or greater, the customer earns mall dollars equal to 0.25% (0.0025) of their total spending.

## determineNextOrderNumber.py

### get\_next\_order\_number(last\_order\_number)

This function takes the last order number as an argument and returns the next order number. It first separates the prefix and number parts of the order number using the '-' character. The number part is converted to an integer for numerical operations. The function then checks whether the current order number is at the maximum value, which is 'Z-999999'. If so, it returns the first order number of the next prefix, which is 'AA-000001'. If the last character of the prefix is 'Z' and the number part is at the maximum value, it increments the prefix and resets the number part to 1. If the number part is at the maximum value but the prefix is not 'Z', it increments the prefix and resets the number part to 1. Otherwise, it increments the number part.

The increment\_prefix() function is a helper function that increments the prefix based on the rules mentioned above.

The returnRandomMoudule() function is not used in this program, and it simply returns a random module from the list ['A', 'B', 'C'].

## extract\_data.py

### convert\_order\_csv\_to\_json()

A function that converts a CSV file to JSON format.

### get\_next\_order\_number(order\_number)

A function from an external module called determineNextOrderNumber that generates the next order number based on the previous one.

## checkDigit.py

calculate\_check\_digit, takes in three arguments: staff\_number (a string of 6 digits), order\_number (a string of 6 digits), and modulus (a string that is either "A", "B", or "C"). The function calculates a check digit that is appended to the end of the order\_number such that the sum of the products of each digit in staff\_number and the corresponding digit in order\_number (i.e., multiplying the first digit in staff\_number with the first digit in order\_number, and so on) is divisible by the integer value associated with modulus. The check digit is returned as an integer value.

The second function, extract\_order\_info, takes in one argument: order\_string (a string that contains information about an order). The function extracts various pieces of information from the order\_string, including the first 1 or 2 alphabets (which determine the format of the order number), the staff\_number (a string of 6 digits), the modulus (a string that is either "A", "B", or "C"), the order\_number (a string of 6 digits), and the number of items (an integer value). These pieces of information are returned as separate variables.

## nameAdress.py

name\_address that takes a parameter customer\_number and returns a tuple with three values. The function tries to open a file named "customer.json" and reads its contents into a dictionary using the json.load() method. It then checks if the customer\_number parameter exists in the dictionary, and if it does, it returns a tuple containing the corresponding customer name, customer address, and invoice date from the dictionary. If the customer\_number parameter is not found in the dictionary, the function returns a tuple containing the strings "Customer not found" and "Customer not found" for the customer name and address, respectively.

## convertorder.py

convert\_order\_csv\_to\_json() prompts the user to input a CSV file to convert it from CSV to json file, it will not accept any files that is not .csv. Once it gets the CSV file, it sorts the DataFrame and group the orders in the format of the CSV. It then create a list for the json data and loop through each order group and convert the json data into a string, and finally writing it in the input.json file.