SuperPy

Track and report inventory

The superpy tool makes it easy to create and track an inventory, as well as stay on top of financial and business matters through: bestselling product reports, inventory alerts, profit tracking, etc.

Index

Tracking inventory..... 1
- *Adding products to inventory..... 1*
- *Selling a product from inventory..... 2*
- *Removing a product from inventory..... 3*

Reporting inventory data..... 4
- *Plotting report data..... 4*
- *Printing report to terminal..... 6*
- *Saving report to csv file..... 7*

Adjusting Time..... 8

Tracking inventory

When you first run superpy a “root folder” will be created in the current directory, along with a “root_date.txt” file. The “root_date” file contains the current inventory date, and is initialized to the current date on it's first run. Your inventory will be initialized as an empty dict at first. Once you add an item to your stock, a “root_inventory.csv”, “root_purchases.csv” and a “root_expiry_dates.csv” file will be added to your “root_folder”. And henceforth, whenever there's a root file available for a property (date, inventory, sales etc), superpy will initialize to that property. If there are any items in your stock, superpy will automatically run an “inventory health scan” to check for any low-stock or expired items in your inventory.

```
% python main.py -h
Created "root_files" folder at
/Users/nnekatielman/Documents/Winc_Academy/BackEnd/superPy/root_files
Created "root_date.txt" file at
/Users/nnekatielman/Documents/Winc_Academy/BackEnd/superPy/root_files/root_date
.txt
Current date is: 2021-01-29
usage: superpy [-h] {buy,sell,discard,plot,print,save,date} ...

Track and report inventory
```

Adding products to inventory:

Populating the inventory is done by making calls to the 'buy' command. Generally this command will store the products and generate purchase records for the transaction. Calling the buy command returns a notification of success or failure and a string with the purchase ID of the transaction. The buy command has 3 required arguments:

- --product: The product or name of the product you wish to add (type=str)
- --cost: The cost of a single unit of the product (type=float)
- --exp-date: The date on which the product expires (type=str)

```
% python main.py buy --product 'apples' --cost 0.3 --exp-date '2021-03-16'
```

```
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring
Success: 1 apples added to inventory. Transaction ID: #SUP210129PURCH01
```

Additionally the buy command also accepts the following optional arguments:

- --quantity: The amount of items you wish to add (default=1, type=int)
- --purch-date: The date on which the product was purchased (default=current inventory date, type=str)

Example adding an item to inventory, that was bought in the past:

```
% python main.py buy --product 'toilet papers' --quantity 9999 --purch-date
'2020-04-01' --cost 1.75 --exp-date '3005-10-10'
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring
Success: 9999 toilet papers added to inventory. Transaction ID:
#SUP200401PURCH01
```

Example adding an item to inventory that will be bought in the future (pre-order)

```
% python main.py buy --product 'iphone 13' --quantity 50 --purch-date '2021-09-
24' --cost 858.00 --exp-date '2022-09-24'
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring
Success: 50 iphone 13 added to inventory. Transaction ID: #SUP210924PURCH01
```

Selling a product from inventory:
The 'sell' command functions similar to the buy command. Calling the sell command returns a notification of success or failure along with a string of the sales ID of the transaction. Conversely, this command will remove the product(s) from your inventory and generate sales records for the transaction. The sell command has 3 required arguments:

- --product: The product or name of the product you wish to sell (type=str)
- --price: The price of a single unit of the product (type=float)
- --purch-ID: The purchase-ID you received when adding the product to inventory (type=str)

```
% python main.py sell --product 'iphone 13' --price 975.50 --purch-ID
'#SUP210924PURCH01'
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring
Success: sold 1 iphone 13 from inventory. Transaction ID: #SUP210129SALE01
```

The sell command accepts the following optional arguments:

- --quantity: The amount of items you wish to sell (default=1, type=int)
- --sell-date: The date on which the product was or will be sold (default=current inventory date, type=str)

Example adding a sales transaction that took place in the past:

```
% python main.py sell --product 'apples' --price 3 --quantity 2 --purch-ID
'#SUP191201PURCH01' --sell-date '2020-05-23'
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring
Success: sold 2 apples from inventory. Transaction ID: #SUP200523SALE01
```

Example adding a sales transaction that will take place in the future (by pre-sale)

```
% python main.py sell --product 'valentine box' --price 11.50 --purch-ID
'#SUP210129PURCH02' --sell-date '2021-02-14'
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring
Success: sold 1 valentine box from inventory. Transaction ID: #SUP210214SALE01
```

Generally superpy will notify you if you try to sell a product quantity higher than what you have in stock, or if you try selling an item that is no longer in stock.

```
% python main.py sell --product 'apples' --price 3 --quantity 8 --purch-ID
'#SUP191201PURCH01' --sell-date '2020-05-23'
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring
Failure: "apples" is no longer in stock
```

Removing a product from inventory:
To remove an item from your inventory without recording it as a sale (because of a product defect, product expired, product was bought for free by the staff bijv:“personeelseten”, etc.) use the 'discard' command. The discard command takes 1 required argument and accepts 1 optional argument:

- --purch-ID: The purchase-ID you received when adding the product to inventory (type=str)
- --quantity: The amount of items you wish to discard (default=1, type=int)

```
% python main.py discard --purch-ID #SUP210125PURCH01
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring
Success: discarded 1 mango from inventory
```

```
% python main.py discard --purch-ID #SUP210125PURCH01 --quantity 3
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring
Success: discarded 3 mango from inventory
```

Reporting inventory data

Superpy is able to produce 9 report-types based on your inventory data:

1. Products report: This report returns a simple list of all available products in inventory
2. Inventory report: Returns the current inventory items and quantities in stock
3. Expiry report: Reports if any products have expired or are nearing expiry
4. Purchase report: Provides an overview of all purchases and expenses of a given date
5. Sales report: Provides an overview of all sales and revenues of a given date
6. Profit report: Calculates your profit for any given date(s)
7. Low-stock report: Lists al products that are running low on stock
8. Bestselling days report: Finds the date(s) with your highest quantity of sales
9. Bestselling products report: Finds the product(s) with the highest quantities sold

All Superpy's reporting commands (plot, print, save) support retrieving data from one or more of the aforementioned report-types. To retrieve more specific report data, all reporting commands support optional “report-type specific” arguments. Each argument has its own short description below, but in more detail they are:

- --stock-qty: This argument is only used for the “Low-stock report” report-type. When creating a low stock report, superpy determines that an item is low on stock, if the item's quantity is lower than 10 (the default minimum stock quantity). The –stock-qty argument allows you to change this default integer.
- --exp-days: Only used with the “Expiry report” report-type. An item is determined to be “Expired” if the current inventory date has passed the item's expiration date. An item is determined to be “Near Expiry” if it's expiration date is within 7 days of the current inventory date. The –exp-days argument allows you to change the default number of days between current inventory date and an item's expiration date.
- --date: Can be used for “Purchase report”, “Sales report”, “Profit report”, “Bestselling days report” and “Bestselling products report”. The –date argument is used to override the default report date (default is all dates from the first inventory transaction date to the current inventory date). To get a report of a specific day, use the date format 'yyyy-mm-dd'. For a specific month report the format 'yyyy-mm' should be employed. And to retrieve report data of a specific year, use the format 'yyyy'

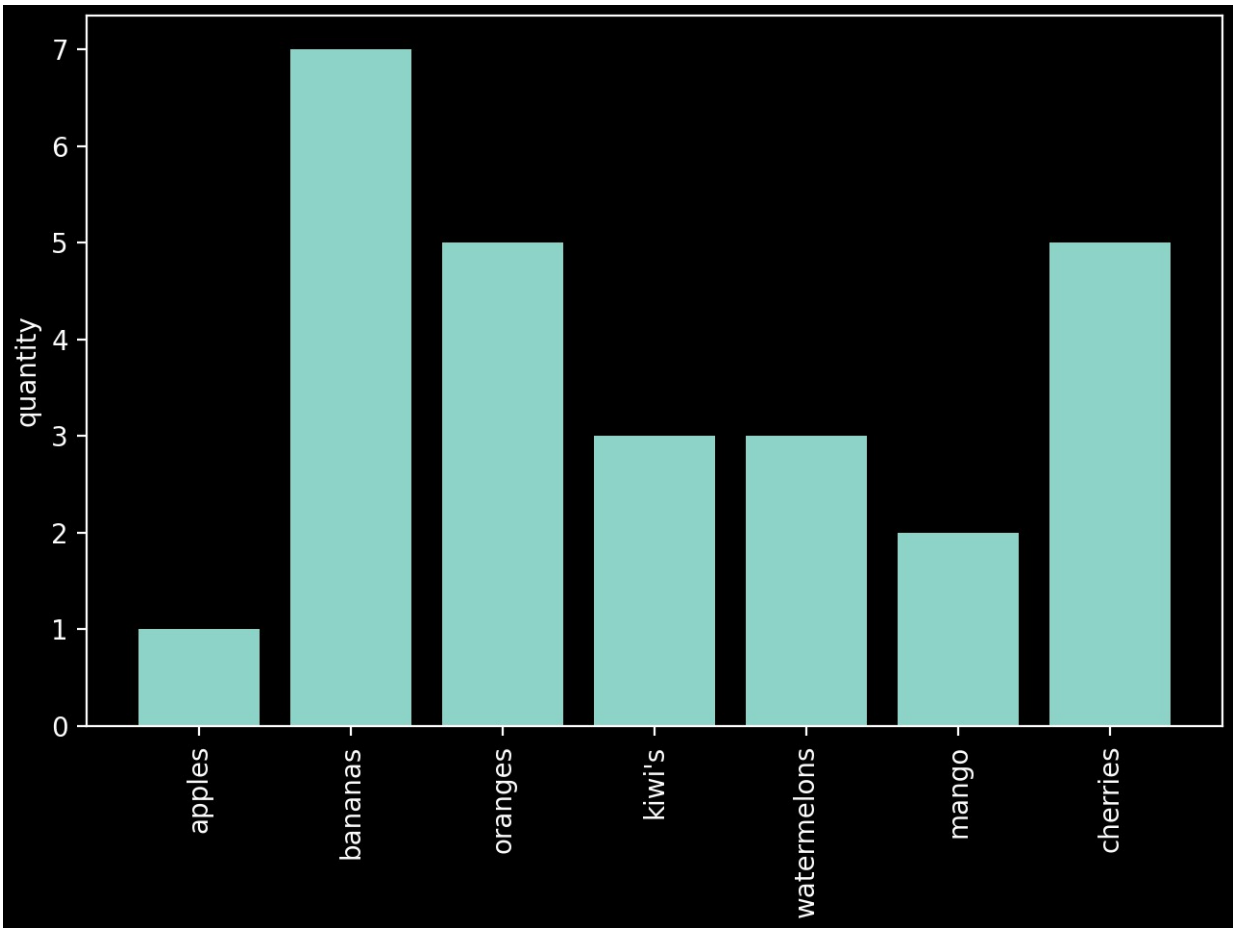
Plotting report data:

With the “plot” command superpy tries to visualize some report data in bar and/ or line charts. The plot command opens up a new window, containing the report type as window title, and a graphical representation of the requested report's data. To plot a report, use the following required arguments:

- --report: The report-type you wish to plot (type=str)
  - *Currently plot only supports “inventory” or “profit” report-types

```
% python main.py plot --report inventory
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
```

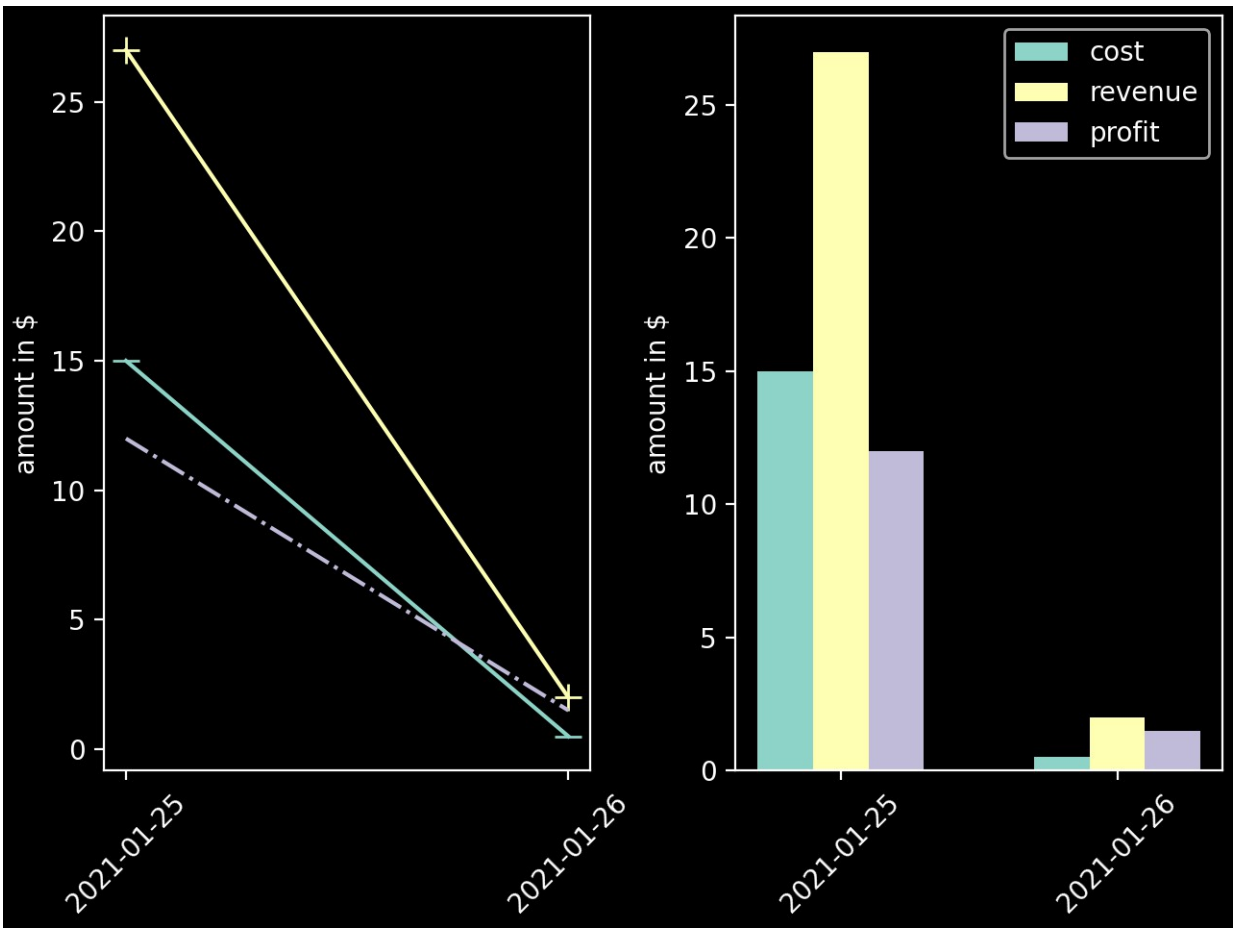
```
Warning: some items have expired or are close to expiring
Plotting inventory report...
```



Plotting a profit report also gives you the following optional argument:

- --date: The profit date(s) you wish to plot (default='-', type=str)
  - *Currently the plot command only supports “yyyy” or “yyyy-mm”

```
% python main.py plot --report profit --date 2021
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring
Plotting profit report...
```



Printing report to terminal:
The “print” command logs a report's data to your terminal. All report-types are supported. Print, if successful, returns a table representation of the report data. If for any reason, the report could not be created, print will return a notification of failure. The print command requires the ensuing argument:

- --report: The report-type you wish to print (type=str)

```
% python main.py print --report expiry
Current date is: 2021-01-29
Running "Inventory Health" scan...
Warning: some items are low on stock
Warning: some items have expired or are close to expiring


EXPIRY REPORT
+----------+----------+-----------------------+
|  Product | Quantity |      Days till expiry |
+----------+----------+-----------------------+
| cherries |      5.0 | EXPIRED on 2012-03-15 |
+----------+----------+-----------------------+
```

Optional print arguments include:

- --date: The report date(s) you wish to print (default='-', type=str)
- --stock-qty: Minimum stock quantity. If a product's quantity is less than this minimum, product is considered to be low on stock (default=10, type=int)

- --exp-days: Number of days from current inventory date till product expiry (default=7, type=int)

```
% python main.py print --report low-stock --stock-qty 5
Current date is: 2021-01-29
Running "Inventory Health" scan...
No items found that expire within 7 days
Warning: some items are low on stock


LOW STOCK REPORT
+--------------+--------------------+
| Product Name | Quantity In Stock |
+--------------+--------------------+
|       apples |                1.0 |
|      oranges |                5.0 |
|       kiwi's |                3.0 |
|  watermelons |                3.0 |
|        mango |                2.0 |
+--------------+--------------------+
```

```
% python main.py print --report purchases --date 2021-01-29
Current date is: 2021-01-29
Running "Inventory Health" scan...
No items found with quantities lower than 10
No items found that expire within 7 days


PURCHASES REPORT
+------------+-------------+------+-----------+-------+------------+--------------------+
|       Date |     Product |  Qty | Unit Cost | Total |   Exp Date |    Transaction ID |
+------------+-------------+------+-----------+-------+------------+--------------------+
| 2021-01-29 |      apples |  1.0 |       0.3 |   0.3 | 2021-03-16 | #SUP210129PURCH01 |
| 2021-01-29 |      apples | 20.0 |       0.5 |  10.0 | 2021-03-15 | #SUP210129PURCH02 |
| 2021-01-29 |     oranges | 20.0 |       0.6 |  12.0 | 2021-02-17 | #SUP210129PURCH03 |
| 2021-01-29 | watermelons | 20.0 |       3.6 |  72.0 | 2021-02-17 | #SUP210129PURCH04 |
| 2021-01-29 |      kiwi's | 20.0 |       1.4 |  28.0 | 2021-06-26 | #SUP210129PURCH05 |
| 2021-01-29 |     bananas | 20.0 |       0.2 |   4.0 | 2021-06-26 | #SUP210129PURCH06 |
+------------+-------------+------+-----------+-------+------------+--------------------+
|                                                                  Total Cost: $126.3 |
+-------------------------------------------------------------------------------------+
```

Saving report to csv file:
superpy supports the ability to save reports to a csv file. When using the save command the program creates a reports folder in the current directory and generates a report title for the csv file in the format “report type-date argument.csv” (purchases2021-01-29.csv). Please note: saving an identical report-type with identical date argument as a previously saved report will cause the program to overwrite the previous report file. All report-types are supported.

- --report: The report-type you wish to save (type=str)

```
% python main.py save --report sales
Current date is: 2021-01-29
Running "Inventory Health" scan...
No items found with quantities lower than 10
No items found that expire within 7 days
Created "reports" folder at
/Users/nnekatielman/Documents/Winc_Academy/BackEnd/superPy/reports
```

```
Success: "sales-2021-01-29.csv" report was saved to:
/Users/nnekatielman/Documents/Winc_Academy/BackEnd/superPy/reports
```

The csv output:

```
Date,Product,Qty,Unit Price,Total,Transaction ID
2021-01-25,mango,6.0,4.5,27.0,#SUP210125SALE01
2021-01-26,apples,1.0,2.0,2.0,#SUP210126SALE01
2020-05-23,apples,2.0,3.0,6.0,#SUP200523SALE01
2020-05-23,apples,2.0,3.0,6.0,#SUP200523SALE02
2020-05-23,apples,2.0,3.0,6.0,#SUP200523SALE03
2021-01-29,mango,2.0,5.0,10.0,#SUP210129SALE01
Total Revenue: $57.0
```

Optional arguments for the save command are:

- --date: The report date(s) you wish to save (default='-', type=str)
- --stock-qty: Minimum stock quantity. If a product's quantity is less than this minimum, product is considered to be low on stock (default=10, type=int)
- --exp-days: Number of days from current inventory date till product expiry (default=7, type=int)

```
% python main.py save --report sales --date 2021
Current date is: 2021-01-29
Running "Inventory Health" scan...
No items found with quantities lower than 10
No items found that expire within 7 days
Success: "sales-2021.csv" report was saved to:
/Users/nnekatielman/Documents/Winc_Academy/BackEnd/superPy/reports
```

The csv output:

```
Date,Product,Qty,Unit Price,Total,Transaction ID
2021-01-25,mango,6.0,4.5,27.0,#SUP210125SALE01
2021-01-26,apples,1.0,2.0,2.0,#SUP210126SALE01
2021-01-29,mango,2.0,5.0,10.0,#SUP210129SALE01
Total Revenue: $39.0
```

Adjusting Time

If there is a “root_date.txt” file present in the roots folder, superpy initializes the current inventory date to the date contained in that file. While it is possible for you to manually change the inventory date by rewriting the date in “root_date.txt”, the 'date' command can also be used to manipulate the inventory date using one of the following required arguments:

- --advance: Amount of days to advance inventory date by (type=int)
- --reverse: Amount of days to reverse inventory date by (type=int)
- --reset: Resets the inventory date to match the current date (action=“store_true”)
    - It is advisable to reset time at the beginning of any session, to ensure the inventory date is set to the current date before completing any other transactions (buying, selling, saving reports, etc.) as a majority of the commands and report-types are date sensitive

```
% python main.py date --advance 5
```

```
Current date is: 2021-01-29
Running "Inventory Health" scan...
No items found with quantities lower than 10
No items found that expire within 7 days
Success: advanced date by 5 days
Current day is: 2021-02-03
```