SuperPy User Guide

Important:
Make sure that:
-you’ve set the working directory in your CLI to “superpy’. All examples in this manual assume that the working directory is ‘superpy’.
-you have read/write access on the the volume where you have installed SuperPy.

Chapter 1. General information

About the SuperPy interface

The messages that SuperPy generates will be displayed in one of the 4 colours:

Blue: all requested information will be displayed in blue.
Yellow: user input is required.
Green: a requested task has completed or is confirmed
Red: in case user input is incorrect or items are expired

Batches versus Inventory

Since fresh products are mostly bought on a daily basis for a price that changes daily, product inventory can exist out of several batches that have different buying prices and expiration dates. This means that the profit made when selling a product could also vary, depending on which part of the inventory of a certain product was sold. In order to be able to handle this information, SuperPy stores inventory as batches that each contain their own buying price, purchase date, amount and expiration date. Let’s take an example:

On Jun. 30, 10 kg of apples were bought for 1.34 per kilo, expiring on Jul. 31
On Jul. 02, 5 kg of apples were bought for 1.17 per kilo, expiring on Aug. 04

SuperPy stores this information as 2 separate batches with their own ID.

As soon as we ask SuperPy to show us the batch list, both batches will show up. And when we ask SuperPy to show us the Inventory list, SuperPy will give us the total stock position of the apples, being 10 + 5 is 15 kg.

Now, as soon as we start selling the apples, SuperPy will use the FIFO (FIRST IN, FIRST OUT) system. This means sold product will be deducted from the stock that came in first, based on the batch ID. So when we’re selling 1 kg of apples, that 1 kg will be deducted from the first batch (containing 9 kg of apples after that) and a buying price of 1.34 per kilo will be registered together with the sold price.
After that, when for example 12 kg of apples are sold in a second transaction, SuperPy takes the first 9 kilos from batch 1, does the sales registration and then

takes the remaining 3 kilos from the second batch and registers that sale accordingly. While splitting up the transaction, all parts of the sale transaction stay coupled by their unique transaction ID.

When calculating profit, the bought price for the first 10 kilo's is deducted from the sale of the first 10 kilo's while the bought price for the remaining 3 kilos is deducted from the sale of the last 3 kilos.

Default time perception
While SuperPy is able to book transactions in retrospect or in the future and can generate information about certain periods of time, by default SuperPy works with today's date for transaction registration if no additional command is given regarding date. If an inventory or batch report is requested, SuperPy uses today 's state to create the inventory or batch list. And the default for revenue report is from start until now.

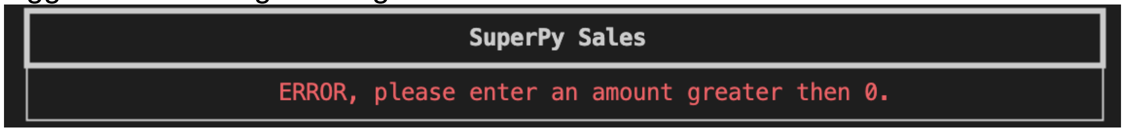
Chapter 2. Operation

2.1. Buy

Registering newly bought inventory items is done with the command:
python super.py buy
This command needs to be appended by adding the product's name, amount, buying price and expiration date, separated by a space.
For example, if 10 kg of apples is bought at a price of 1.39 per kg and with expiration date 08 august 2022, the command would look like this:
python super.py buy apples 10 1.39 2022-08-10

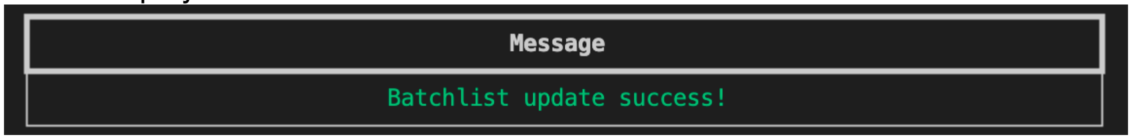
Note: just like the expiration date all dates have to be entered in ISO format (YYYY-MM-DD)

Please also note that **the SuperPy buy function does not accept negative product amounts** or an amount of 0. Any attempt to enter such amount will trigger the following message :



```
SuperPy Sales
ERROR, please enter an amount greater then 0.
```

After completing the registration of newly bought product, the following message will be displayed:



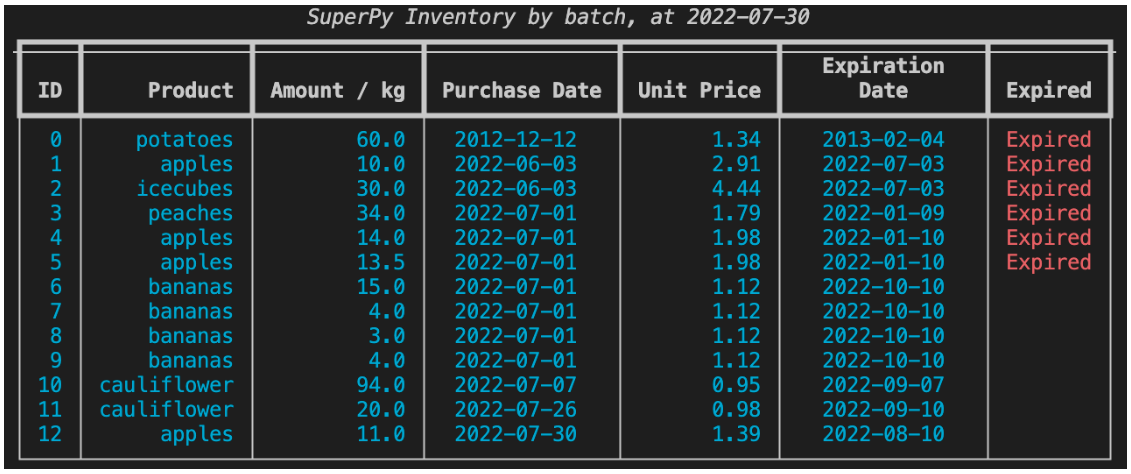
```
Message
Batchlist update success!
```

In order to book a purchase in retrospect or on a date in the future, an optional

command can be added to the buy command:
-y : book the transaction on yesterday's date
-t : book the transaction on tomorrow's date
-d (YYYY-MM-DD): book the transaction on any chosen date
An example: *python super.py buy apples 10 1.39 2022-08-10 -y

2.2. Batch

When the batch command is given, SuperPy shows a list of all batches that contain product; batches with the amount of 0 are not shown.
The Batch function is called with the command:
python super.py batch. An example of such Inventory by batch list is shown hereunder:



```
                SuperPy Inventory by batch, at 2022-07-30
┌────┬──────────────┬─────────────┬───────────────┬────────────┬────────────────┬─────────┐
│    │              │             │               │            │   Expiration   │         │
│ ID │      Product │ Amount / kg │ Purchase Date │ Unit Price │      Date      │ Expired │
├────┼──────────────┼─────────────┼───────────────┼────────────┼────────────────┼─────────┤
│  0 │     potatoes │        60.0 │  2012-12-12   │       1.34 │   2013-02-04   │ Expired │
│  1 │       apples │        10.0 │  2022-06-03   │       2.91 │   2022-07-03   │ Expired │
│  2 │     icecubes │        30.0 │  2022-06-03   │       4.44 │   2022-07-03   │ Expired │
│  3 │      peaches │        34.0 │  2022-07-01   │       1.79 │   2022-01-09   │ Expired │
│  4 │       apples │        14.0 │  2022-07-01   │       1.98 │   2022-01-10   │ Expired │
│  5 │       apples │        13.5 │  2022-07-01   │       1.98 │   2022-01-10   │ Expired │
│  6 │      bananas │        15.0 │  2022-07-01   │       1.12 │   2022-10-10   │         │
│  7 │      bananas │         4.0 │  2022-07-01   │       1.12 │   2022-10-10   │         │
│  8 │      bananas │         3.0 │  2022-07-01   │       1.12 │   2022-10-10   │         │
│  9 │      bananas │         4.0 │  2022-07-01   │       1.12 │   2022-10-10   │         │
│ 10 │  cauliflower │        94.0 │  2022-07-07   │       0.95 │   2022-09-07   │         │
│ 11 │  cauliflower │        20.0 │  2022-07-26   │       0.98 │   2022-09-10   │         │
│ 12 │       apples │        11.0 │  2022-07-30   │       1.39 │   2022-08-10   │         │
└────┴──────────────┴─────────────┴───────────────┴────────────┴────────────────┴─────────┘
```

On the top you'll find the name of the displayed table, together with the requested date, in this case 2022-07-30.
The table gives a row by row overview of each product batch: their ID, amount, purchase date, unit price, expiration date and a warning when a batch has expired.

In order to show the state of the batch list on a different date than today, an optional command can be added to the batch command:
-y: show the batch list state on yesterday's date
-d (YYYY-MM-DD): show the batch list state on any chosen date
An example: python super.py batch -y

2.3. Edit

In case a mistake was made while registering a new product batch, batches can be adjusted with the **edit** function.

For example, let's assume we have the following batch that was displayed after calling the batch function.

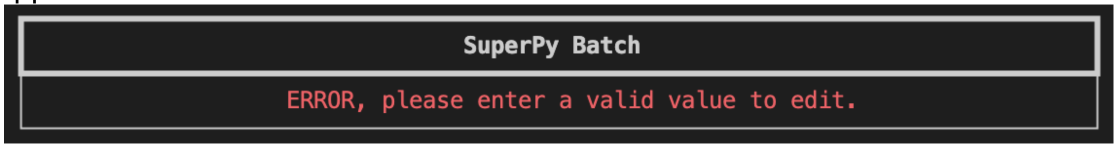
```
 5 |     apples |       13.5 |  2022-07-01    |       1.98 |   2022-01-10
```

The edit function allows the following variables to be edited:
-**product** (name of the product)
-**amount**
-**price** (buying price)
-**purch_date** (purchase date in the ISO format YYYY-MM-DD)
-**expired** (expiration date in the ISO format YYYY-MM-DD)

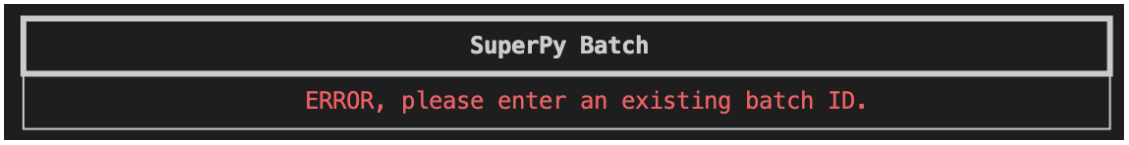
If we would like to change the amount from 13.5 to 15, first we need to look up the ID of the batch that needs to be edited. In this case it's 5.
Second, we enter the variable that we would like to edit, which is amount.
Last, we enter the new amount, which is 15. The command will look like this:
python super.py edit 5 amount 15

WARNING: Batch information as well as sales information is stored in CSV files. Do not attempt to manually edit these CSV files,
for example with a text editor, because this could cause corrupted data to be stored and will affect SuperPy's functioning.

If a non-existing parameter or ID is entered, one of the following messages appears:

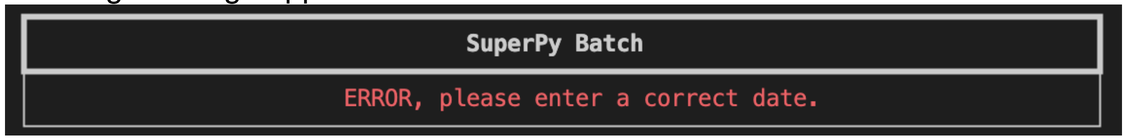


```
SuperPy Batch
ERROR, please enter a valid value to edit.
```



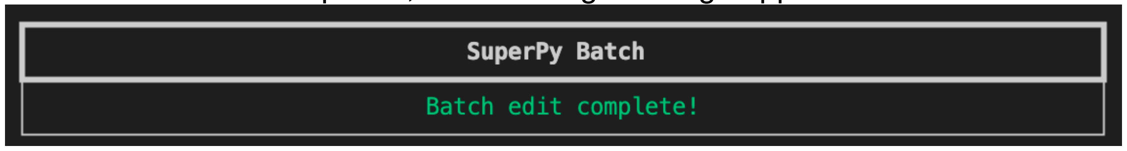
```
SuperPy Batch
ERROR, please enter an existing batch ID.
```

If the purchase date or expired date are not entered in the correct format, the following message appears:



```
SuperPy Batch
ERROR, please enter a correct date.
```

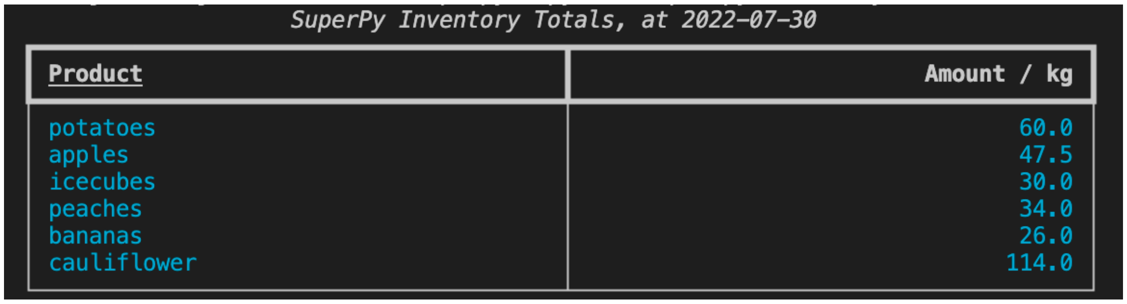
If a batch edit was completed, the following message appears:



```
SuperPy Batch
Batch edit complete!
```

2.4. Inventory

The inventory command shows a consolidated list of all products that are in stock: the Inventory Totals list.
Command: **python super.py inventory**



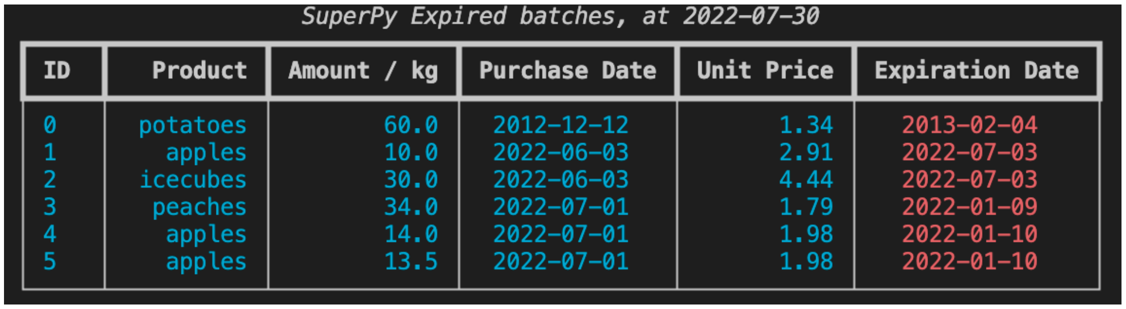
```
            SuperPy Inventory Totals, at 2022-07-30
┌──────────────────────────────────────┬──────────────────────────────────────┐
│ Product                              │                          Amount / kg │
├──────────────────────────────────────┼──────────────────────────────────────┤
│ potatoes                             │                                 60.0 │
│ apples                               │                                 47.5 │
│ icecubes                             │                                 30.0 │
│ peaches                              │                                 34.0 │
│ bananas                              │                                 26.0 │
│ cauliflower                          │                                114.0 │
└──────────────────────────────────────┴──────────────────────────────────────┘
```

On the top you'll find the name of the displayed table, together with the requested date, in this case 2022-07-30.
On the left all the products are displayed that have actual stock, on the right the corresponding amounts are displayed.

In order to show the state of the inventory totals list on another given date, an optional command can be added to the batch command:
-y: show the inventory state on yesterday's date
-d (YYYY-MM-DD): show the inventory state on any chosen date.

2.5. Expired

"Expired" generates a list with all the batches that have expired on the requested date. We can call this function with:
python super.py expired



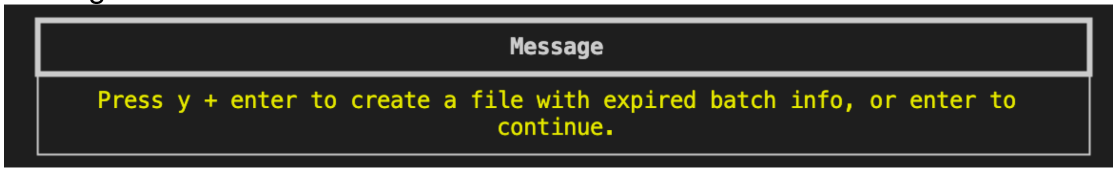
```
                 SuperPy Expired batches, at 2022-07-30
┌────┬──────────┬─────────────┬───────────────┬────────────┬─────────────────┐
│ ID │  Product │ Amount / kg │ Purchase Date │ Unit Price │ Expiration Date │
├────┼──────────┼─────────────┼───────────────┼────────────┼─────────────────┤
│ 0  │ potatoes │        60.0 │  2012-12-12   │       1.34 │   2013-02-04    │
│ 1  │   apples │        10.0 │  2022-06-03   │       2.91 │   2022-07-03    │
│ 2  │ icecubes │        30.0 │  2022-06-03   │       4.44 │   2022-07-03    │
│ 3  │  peaches │        34.0 │  2022-07-01   │       1.79 │   2022-01-09    │
│ 4  │   apples │        14.0 │  2022-07-01   │       1.98 │   2022-01-10    │
│ 5  │   apples │        13.5 │  2022-07-01   │       1.98 │   2022-01-10    │
└────┴──────────┴─────────────┴───────────────┴────────────┴─────────────────┘
```

On the top you'll find the name of the displayed table, together with the requested date, in this case 2022-07-30.

In order to show the state of the expired batches list on another given date, an optional command can be added to the inventory command:
-y: show the inventory state on yesterday's date
-t: show the inventory state on tomorrow's date

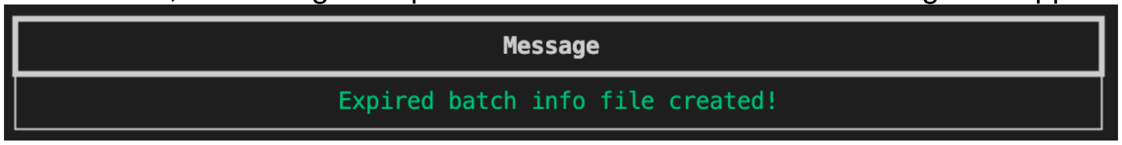
-ad: Advance or back date with a set number of days (-99+99)
-d (YYYY-MM-DD): show the inventory state on any chosen date.

Besides the fact that it might be handy to see all expired batches in one dedicated list, this function comes with 2 extra options:
After displaying the expired batches table, SuperPy generates the following message:



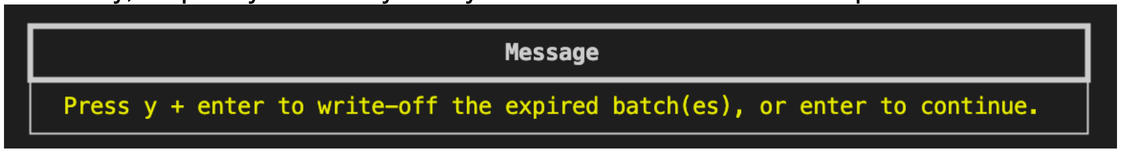
```
Message
Press y + enter to create a file with expired batch info, or enter to
continue.
```

When 'y' + 'enter' is pressed (pressing just enter will pass this option) SuperPy creates a csv.file in the superpy directory with the name 'expired batch-YYYY-MM-DD.csv', containing all expired batches. A confirmation message will appear:



```
Message
Expired batch info file created!
```

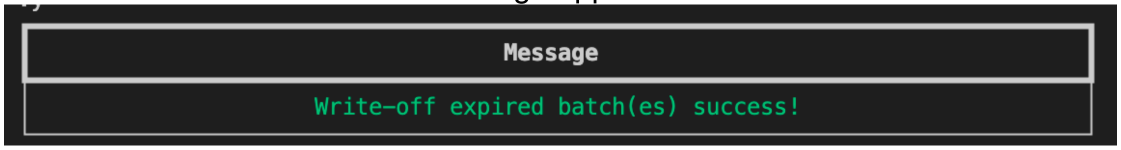
Secondly, SuperPy will ask you if you would like to write off expired batches:



```
Message
Press y + enter to write-off the expired batch(es), or enter to continue.
```

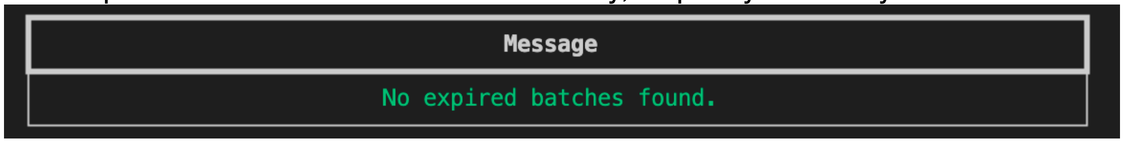
Since most of the times a store is not allowed to sell expired products, these products should be removed from the inventory and the costs should be booked as a loss.
This is exactly what this function does: expired batch amounts will be set to zero and their cost price will be booked in sales, like the products were sold for 0.00.
Again, pressing 'y' + 'enter' will cause this function to be executed, while just pressing enter will cause the option to be passed.
Note: Write-off transactions always follow today's date and not the expiration date.

After execution a confirmation message appears:



```
Message
Write-off expired batch(es) success!
```

If no expired batches were found in inventory, SuperPy will notify with:



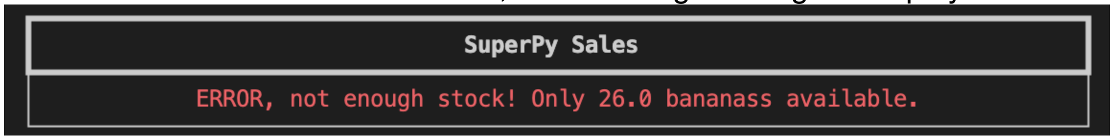
```
Message
No expired batches found.
```

2.6. Sell

The sell function is used to book any sales into the system.
This can be done in a pretty straight-forward way.
Simply type: **python super.py sell** and add the product, amount and selling price separated by a space.

So, if we would like to register the sales of 3 kg of bananas for 2.99 per kilo, the command would look like: **python super.py sell bananas 3 2.99**
As mentioned earlier in chapter 1, SuperPy will deduct sales from inventory based on the lowest batch ID first, continuing with the next one until the total sold amount is deducted from inventory.

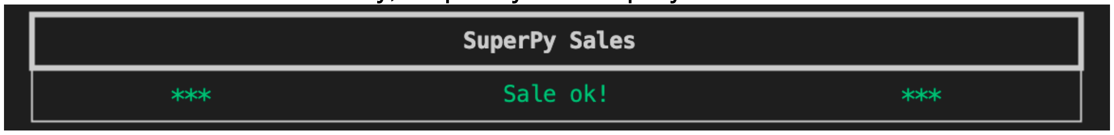
After receiving the sell command, SuperPy will check first if there will be enough stock to meet the sold amount. If not, the following message is displayed:



```
SuperPy Sales
ERROR, not enough stock! Only 26.0 bananass available.
```

Also, when a user tries to register the sale of some product that's not in stock, this will trigger the same error message.

If there is sufficient inventory, SuperPy will display:



```
                         SuperPy Sales
      ***                  Sale ok!                   ***
```

and deduct the amounts from batches accordingly and for each inventory batch mutation it will book a sales transaction line.

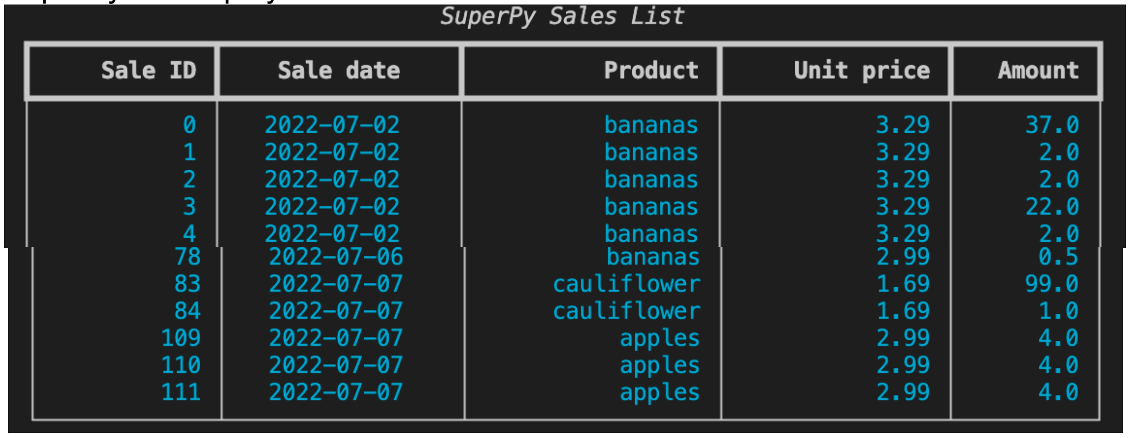
In order to book a sale in retrospect or on a date in the future, an optional command command be added to the sell command:
-y: book the transaction on yesterday's date
-t: book the transaction on tomorrow's date
-d (YYYY-MM-DD): book the transaction on any chosen date
An example: *python super.py sell apples 10 2.99 -y

2.7. Return

Sometimes fresh products may be returned because of a quality issue. Since products with a quality issue will not be restocked but thrown away, the Return function helps to make the administrative booking in order to adjust revenue and profit accordingly, without placing the item back in stock.
The return function can by executed by typing:

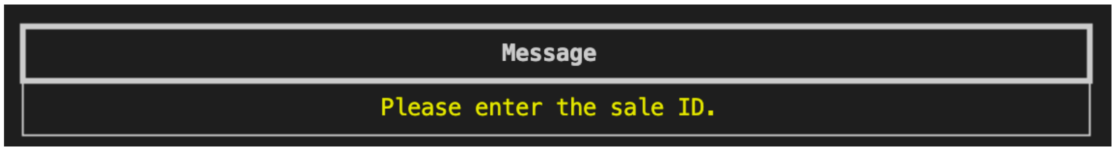
python super.py return

SuperPy will display a list of sales that will look like this:

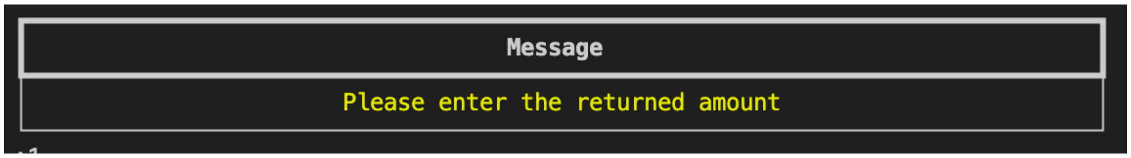


```
                           SuperPy Sales List
┏━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━┓
┃ Sale ID ┃ Sale date  ┃       Product ┃ Unit price ┃ Amount ┃
┡━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━┩
│       0 │ 2022-07-02 │       bananas │       3.29 │   37.0 │
│       1 │ 2022-07-02 │       bananas │       3.29 │    2.0 │
│       2 │ 2022-07-02 │       bananas │       3.29 │    2.0 │
│       3 │ 2022-07-02 │       bananas │       3.29 │   22.0 │
│       4 │ 2022-07-02 │       bananas │       3.29 │    2.0 │
│      78 │ 2022-07-06 │       bananas │       2.99 │    0.5 │
│      83 │ 2022-07-07 │   cauliflower │       1.69 │   99.0 │
│      84 │ 2022-07-07 │   cauliflower │       1.69 │    1.0 │
│     109 │ 2022-07-07 │        apples │       2.99 │    4.0 │
│     110 │ 2022-07-07 │        apples │       2.99 │    4.0 │
│     111 │ 2022-07-07 │        apples │       2.99 │    4.0 │
└─────────┴────────────┴───────────────┴────────────┴────────┘
```

In the list you can find the sale ID that corresponds with the return that you would like to book.
Directly after displaying the list of sales, SuperPy will ask you for the sale ID and the returned amount:

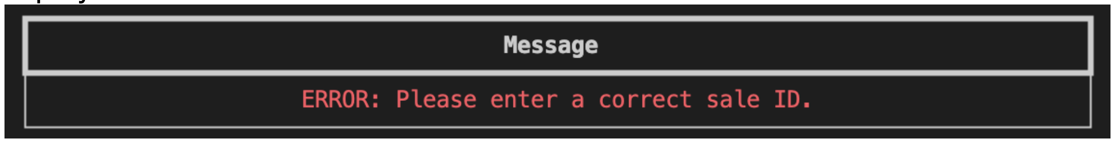


```
Message
Please enter the sale ID.
```

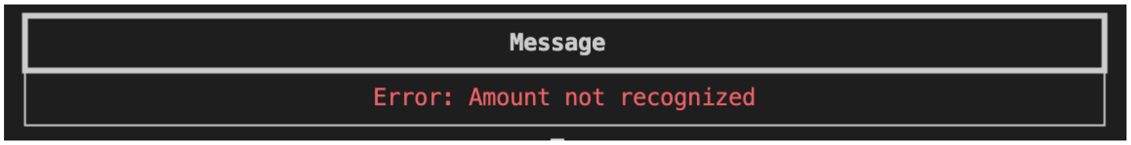


```
Message
Please enter the returned amount
```

When the entered Sale ID or amount is not recognised, an error message will be displayed:

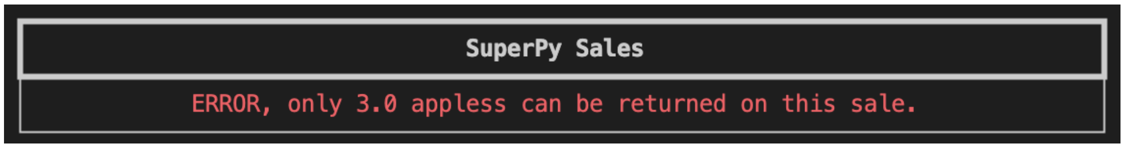


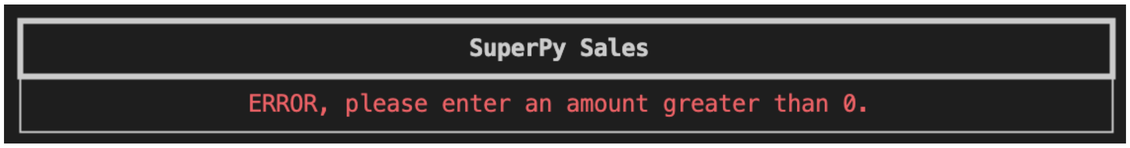
```
Message
ERROR: Please enter a correct sale ID.
```



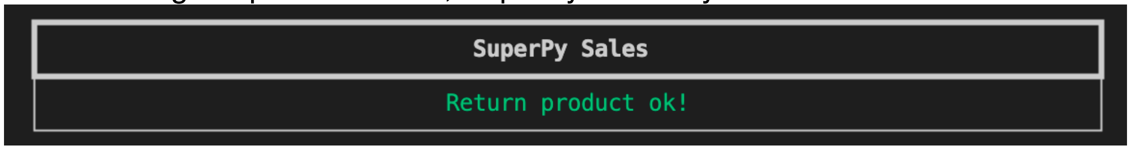
```
Message
Error: Amount not recognized
```

SuperPy also checks that the returned amount on a transaction is never bigger than the amount sold or smaller than 1.
If this situation occurs, one of the following messages is displayed:


```
SuperPy Sales
ERROR, only 3.0 appless can be returned on this sale.
```


```
SuperPy Sales
ERROR, please enter an amount greater than 0.
```

After booking the product return, SuperPy will notify with:

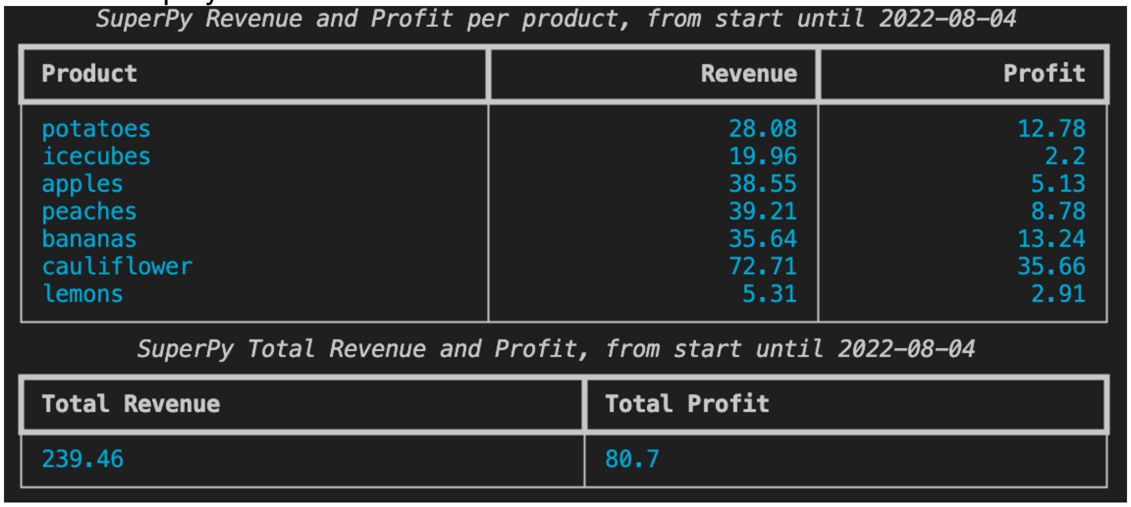

```
SuperPy Sales
Return product ok!
```

2.7 Revenue

The revenue command lets SuperPy build a revenue and profit report, which contains revenue and profit per product and the total revenue and profit numbers.

Command: python super.py revenue

This will display:


```
      SuperPy Revenue and Profit per product, from start until 2022-08-04
 ┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━┓
 ┃ Product                          ┃               Revenue ┃             Profit ┃
 ┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━┩
 │ potatoes                         │                 28.08 │              12.78 │
 │ icecubes                         │                 19.96 │                2.2 │
 │ apples                           │                 38.55 │               5.13 │
 │ peaches                          │                 39.21 │               8.78 │
 │ bananas                          │                 35.64 │              13.24 │
 │ cauliflower                      │                 72.71 │              35.66 │
 │ lemons                           │                  5.31 │               2.91 │
 └──────────────────────────────────┴───────────────────────┴────────────────────┘
         SuperPy Total Revenue and Profit, from start until 2022-08-04
 ┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
 ┃ Total Revenue                          ┃ Total Profit                        ┃
 ┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
 │ 239.46                                 │ 80.7                                │
 └────────────────────────────────────────┴─────────────────────────────────────┘
```

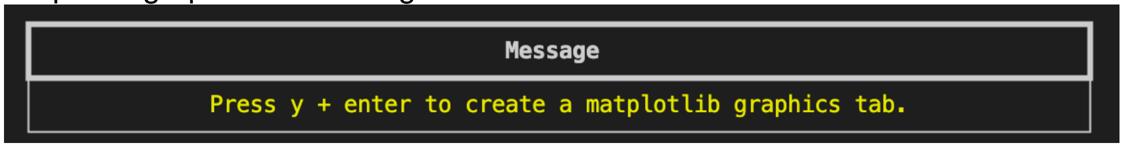
In the upper right corner, the start and end date of the requested period are displayed.

SuperPy is able to create a revenue and profit report for each requested period of time. In order to make it easier for the user to request reports over different periods of time,
some presets were created:

-td (—today): shows revenue and profit generated today
-y (—yesterday): shows revenue and profit generated yesterday
-w (—thisweek): shows revenue and profit generated this week until now
(starting on Monday)
-m (—thismonth): shows revenue and profit generated this month until now
-th (—thisyear): shows revenue and profit generated this year until now
-r (—range): shows revenue and profit generated in a set range (YYYY-MM-DD YYYY-MM-DD)
-year (—year): shows revenues and profit generated in a specific year (YYYY)
-f (—from): shows revenues and profit generated from set date (YYYY-MM-DD) until now

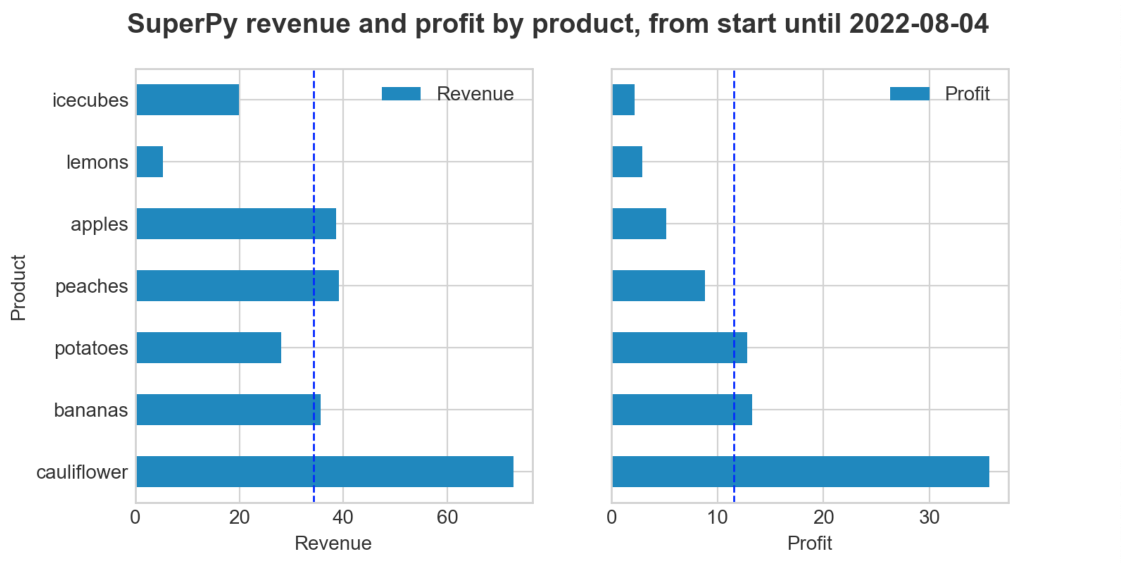
Note: with **-f** / **—from** it's also possible to enter just a year (start date is set to YYYY-01-01) or year-month (start date is set to YYYY-MM-01).
Example: **-f 2022-dec** would set a start date of 2022-12-01 and **-f 2022** would set a start date of 2022-01-01.

If any revenues were generated in the requested period, SuperPy will ask if a Matplotlib graphic should be generated:

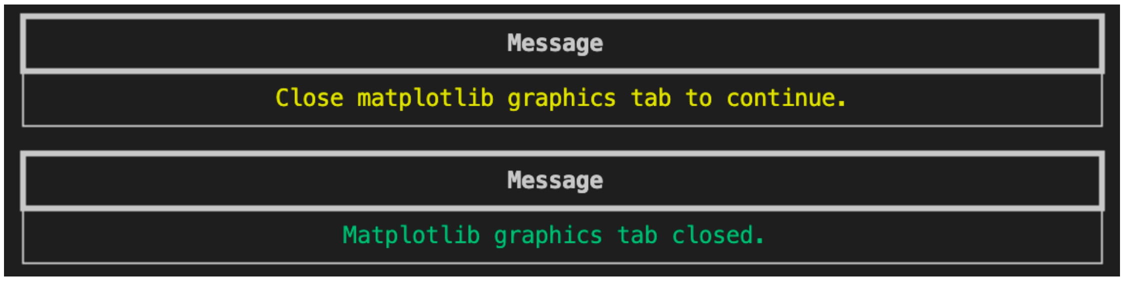


```
Message
Press y + enter to create a matplotlib graphics tab.
```

When 'y' + 'enter' is pressed (pressing just enter will pass this option) a Matplotlib file is generated, showing revenue and profit per product.



NOTE: While this file is opened, SuperPy will pause and wait for the user to close the tab. After closing SuperPy will continue.



```
                    Message
      Close matplotlib graphics tab to continue.

                    Message
          Matplotlib graphics tab closed.
```