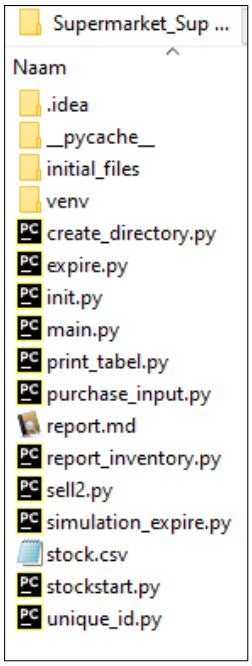
ASSIGNMENT SUPER MARKET—SUPERPY

by Erwin van der Haar (May 2021)



The assignment describes a supermarket SuperPy that manages stock items through command line input. Certain commands are required like the purchase of items, sales of items and expiration management.

In this comprehensive manual five main command line instructions are described in order to fulfil the required tasks. This manual does not explain the underlaying python code. However, an overview of the interacting modules is enclosed as attachment to this manual.

START UP

Once the files for this assignment have been placed in a directory the user is required to go to a command line prompt in order to interact with the python programms.

Apart from a list of required python programms , an initial inventory file ("stock.csv") is needed and serves as input for the already existing super market.

> python main.py

Main.py is the first module to start. If this is the first time main.py is called upon, an "initial files" directory is created and based on the "stock.csv" will be extracted. New files will be created from this inventory, placed in a newly created sub-directory "initial_files". The command results in an overview of the inventory of that moment, including all inventory items with unique id's per listed item.

In this inventory determination the program scans for expired items and lists them that way. The computer time is taken as the

```
Created initial_files folder in C:\Users\HP\academy\Supermarket_SuperPy

-------------------------------------------------------------------------------------------------------------------
id     | product_name | purchase_date | purchase_amount | purchase_price | expiration_date | cool_storage | sold_date  | sold_price | exit_status |
-------------------------------------------------------------------------------------------------------------------
174773 | kaas         | 2021-01-05    | 2.0             | 7.5            | 2021-01-07      | yes          | 0          | 0.0        | expired     |
176792 | worst        | 2021-04-05    | 3.0             | 1.5            | 2021-05-05      | yes          | 2021-02-03 | 18.95      | sold        |
141994 | eieren       | 2021-04-05    | 4.0             | 3.5            | 2021-07-05      | yes          | 0          | 0.0        | n           |

92362  | toastjes     | 2021-04-05    | 5.0             | 5.5            | 2021-05-05      | yes          | 2021-01-01 | 40.0       | sold        |
80145  | wijn         | 2021-02-05    | 6.0             | 6.5            | 2021-02-17      | yes          | 0          | 0.0        | expired     |
1284   | brood        | 2021-04-05    | 7.0             | 8.5            | 2021-05-25      | yes          | 0          | 0.0        | n           |

57479  | jam          | 2021-04-05    | 8.0             | 9.5            | 2021-05-05      | yes          | 2021-02-03 | 2.22       | sold        |
-------------------------------------------------------------------------------------------------------------------


       number of inventory items: 7

       total purchase costs:           42.5
       total of 3 sales:               61.17
       number of expired items today:  2
       valueloss due to expired items: 14.0
       today's profit on total stock:  4.67



Your input is required, based on information from the HELP file.
Further optional commands are report, purchase, sell or expire.

Type 'python main.py -h' for additional information.
```

date of "today". For completing these initial tasks the program "init.py" is executed. As part of this process expired items are examined and identified as such.

From the attached process overview the red arrows indicate the different command line executions to the associated modules. As the WINC task has been adjusted the module SELL.PY has been renamed to SELL2.PY as it is in it's second revision stage.

From this point onwards commands can be given to execute various events on the inventory. To absorb the users requests, the program uses command line arguments and sub-arguments to deliver the information of the event to the related underlaying modules, to be recognized by the corresponding file names. In the following paragraphs and following pages the optional executional commands are explained.

Via the command **python main.py –h** an overview is given of possible recognized commands.

```
C:\Users\HP\academy\Supermarket_SuperPy>python main.py -h
 The initial start-up of the Super Market Py has been executed.
usage: Supermarket SuperPy [-h] {report,purchase,sell,expire} ...

positional arguments:
  {report,purchase,sell,expire}
    report              report stock inventory
    purchase            start process of purchasing item
    sell                information on sold items
    expire              simulate expire dates and periods

optional arguments:
  -h, --help            show this help message and exit
```

```
C:\Users\HP\academy\Supermarket_SuperPy>python main.py purchase
 The initial start-up of the Super Market Py has been executed.
The purchase of a product includes many information fields. Your input per field is required.

Would you like the full input routine (if not a standard line is added) (y/n): n
the following files have been updated with the purchase information:

C:\Users\HP\academy\Supermarket_SuperPy\initial_files\purchase.csv
C:\Users\HP\academy\Supermarket_SuperPy\initial_files\inventory.csv
C:\Users\HP\academy\Supermarket_SuperPy\initial_files\expires_dates.csv

Purchase completed

-----------------------------------------------------------------------------------------------------------------------------------
 id     | product_name | purchase_date | purchase_amount | purchase_price | expiration_date | cool_storage | sold_date  | sold_price | exit_status |
-----------------------------------------------------------------------------------------------------------------------------------
 246810 | schaap       | 2021-04-15    | 12.0            | 150.0          | 2021-07-25      | n            | 0          | 0.0        | n           |
 174773 | kaas         | 2021-01-05    | 2.0             | 7.5            | 2021-01-07      | yes          | 0          | 0.0        | n           |
 176792 | worst        | 2021-04-05    | 3.0             | 1.5            | 2021-05-05      | yes          | 2021-02-03 | 18.95      | sold        |

 141994 | eieren       | 2021-04-05    | 4.0             | 3.5            | 2021-07-05      | yes          | 0          | 0.0        | n           |
 92362  | toastjes     | 2021-04-05    | 5.0             | 5.5            | 2021-05-05      | yes          | 2021-01-01 | 40.0       | sold        |
 80145  | wijn         | 2021-02-05    | 6.0             | 6.5            | 2021-02-17      | yes          | 0          | 0.0        | n           |

 1284   | brood        | 2021-04-05    | 7.0             | 8.5            | 2021-05-25      | yes          | 0          | 0.0        | n           |
 57479  | jam          | 2021-04-05    | 8.0             | 9.5            | 2021-05-05      | yes          | 2021-02-03 | 2.22       | sold        |
-----------------------------------------------------------------------------------------------------------------------------------

C:\Users\HP\academy\Supermarket_SuperPy>
```

> python main.py purchase

For items that have been bought and need to be added to the supermarket inventory, the program facilitates this requirement.

Once the program has started the user is asked to follow a lengthy input sequence or to add a pre-defined item to the inventory. Although this predefined item is always the same product with associated attributes, the unique-id makes the item(s) unique in the inventory. The lengthy input sequence asks for details concerning product attributes.

After the input choice and actions the purchase file is displayed, including previously purchased items mentioned in the inventory. Purchased items are marked with a “none” exit status as they are neither sold nor expired.

> python main.py report

At any time the user can ask to present the complete inventory of the super market. It will display all items known to the inventory, including historical items and their attributes. In addition to showing the items with their respective information the program shows the supermarket investments (purchase), sales values and value loss due to expired items. It finally calculates the profit (or loss) given the momentary inventory values.

```
C:\Users\HP\academy\Supermarket_SuperPy>python main.py report
 The initial start-up of the Super Market Py has been executed.

-----------------------------------------------------------------------------------------------------------------------------------
 id     | product_name | purchase_date | purchase_amount | purchase_price | expiration_date | cool_storage | sold_date  | sold_price | exit_status |
-----------------------------------------------------------------------------------------------------------------------------------
 246810 | schaap       | 2021-04-15    | 12.0            | 150.0          | 2021-07-25      | n            | 0          | 0.0        | n           |
 174773 | kaas         | 2021-01-05    | 2.0             | 7.5            | 2021-01-07      | yes          | 0          | 0.0        | expired     |
 176792 | worst        | 2021-04-05    | 3.0             | 1.5            | 2021-05-05      | yes          | 2021-02-03 | 18.95      | sold        |

 141994 | eieren       | 2021-04-05    | 4.0             | 3.5            | 2021-07-05      | yes          | 0          | 0.0        | n           |
 92362  | toastjes     | 2021-04-05    | 5.0             | 5.5            | 2021-05-05      | yes          | 2021-01-01 | 40.0       | sold        |
 80145  | wijn         | 2021-02-05    | 6.0             | 6.5            | 2021-02-17      | yes          | 0          | 0.0        | expired     |

 1284   | brood        | 2021-04-05    | 7.0             | 8.5            | 2021-05-25      | yes          | 0          | 0.0        | n           |
 57479  | jam          | 2021-04-05    | 8.0             | 9.5            | 2021-05-05      | yes          | 2021-02-03 | 2.22       | sold        |
-----------------------------------------------------------------------------------------------------------------------------------


        number of inventory items: 8

        total purchase costs:           192.5
        total of 3 sales:               61.17
        number of expired items today:  2
        valueloss due to expired items: 14.0
        today's profit on total stock:  -145.33
```

> python main.py sell

```
C:\Users\HP\academy\Supermarket_SuperPy>python main.py sell -h
 The initial start-up of the Super Market Py has been executed.
usage: Supermarket SuperPy sell [-h] -id ID -soldprice SOLDPRICE [-solddate SOLDDATE]

optional arguments:
  -h, --help            show this help message and exit
  -id ID, --id ID       identification number of sold item
  -soldprice SOLDPRICE, --soldprice SOLDPRICE
                        price for sold item
  -solddate SOLDDATE, --solddate SOLDDATE
                        date of sold item. Default = today
```

Through an “argparse/subparse” sequence the user is able to sell specific inventory items, indicating the required ID number and sales price. An optional sold date is possible (default = today) which completes the process. The user is presented with a sales overview with the momentary inventory information from the past.

```
C:\Users\HP\academy\Supermarket_SuperPy>python main.py sell -id 141994 -soldprice 12.90
 The initial start-up of the Super Market Py has been executed.

 Action: sell item

 The indicated item-ID:          141994
 The indicated sold date:        2021-05-05
 The total selling price:        12.90

Transaction completed; item 141994 and sold action completed

Total sales:                     74.07

 Sales overview:

-------------------------------------------------------------------------------------------------------------
 id     | product_name | purchase_date | purchase_amount | purchase_price | sold_date  | sold_price | exit_status |
-------------------------------------------------------------------------------------------------------------
 141994 | eieren       | 2021-04-05    | 4.0             | 3.5            | 2021-05-05 | 12.90      | sold        |
 176792 | worst        | 2021-04-05    | 3.0             | 1.5            | 2021-02-03 | 18.95      | sold        |
 92362  | toastjes     | 2021-04-05    | 5.0             | 5.5            | 2021-01-01 | 40.0       | sold        |

 57479  | jam          | 2021-04-05    | 8.0             | 9.5            | 2021-02-03 | 2.22       | sold        |
-------------------------------------------------------------------------------------------------------------
```

A build-in check involves the consideration if the inventory item has already been sold, has been expired or the ID is not existing in the inventory database. In these cases the program terminates and indicates an impossible sale.

> python main.py expire

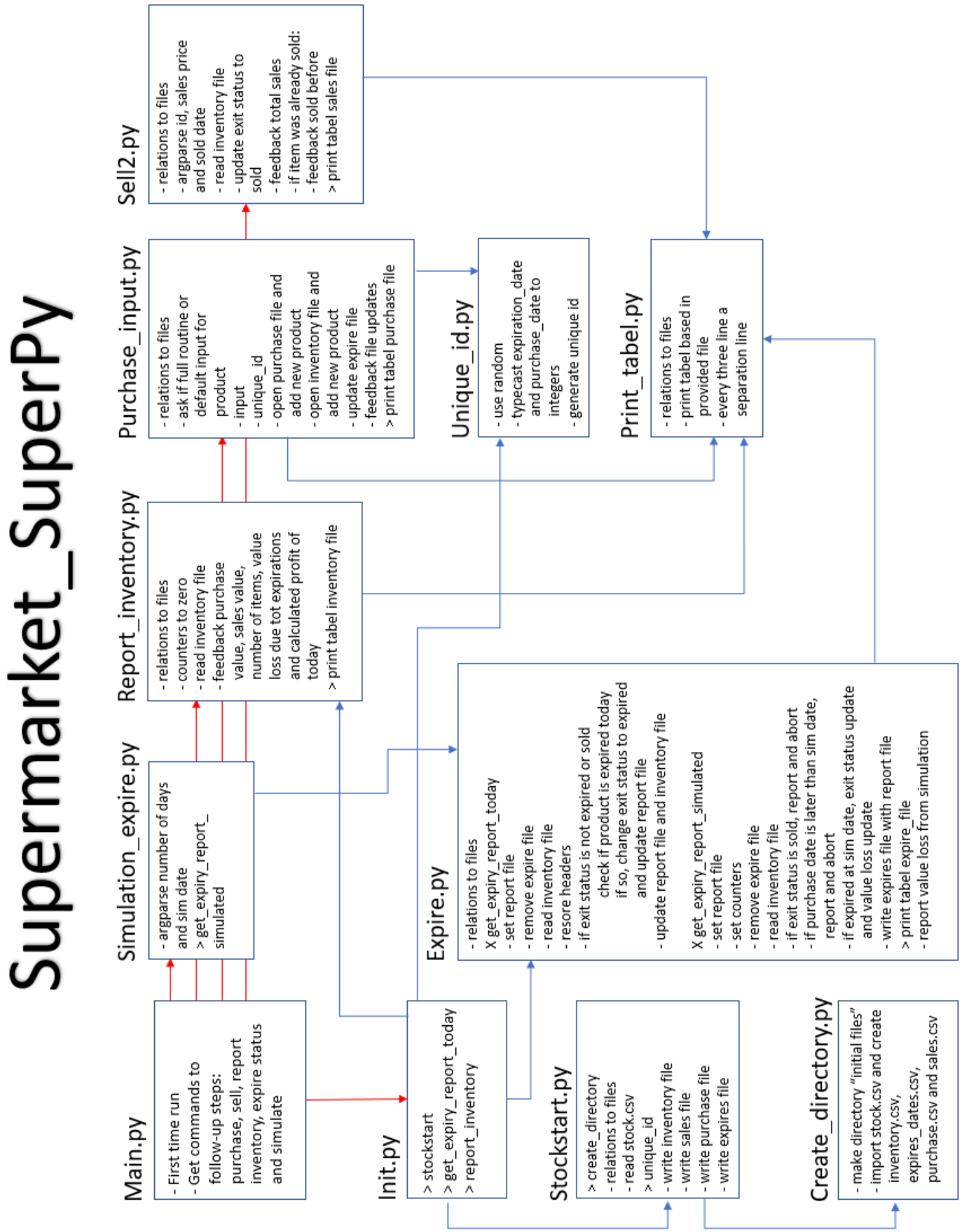
```
C:\Users\HP\academy\Supermarket_SuperPy>python main.py expire -h
 The initial start-up of the Super Market Py has been executed.
usage: Supermarket SuperPy expire [-h] [-simdate SIMDATE] -numdays NUMDAYS

optional arguments:
  -h, --help            show this help message and exit
  -simdate SIMDATE, --simdate SIMDATE
                        start date of simulation. Default = today
  -numdays NUMDAYS, --numdays NUMDAYS
                        number of simulated days
```

The program can simulate a date-shift and produce a list of simulated expired items. Through an “argparse/subparse” procedure the startdate of the check can be given by the user and the period (number of days) for which the expiration determination will be simulated.

The program presents the user with an overview of expired stock items at the time of the give simulation date. In addition the

```
C:\Users\HP\academy\Supermarket_SuperPy>python main.py expire -numdays 30
 The initial start-up of the Super Market Py has been executed.

 The simulated date for this run is 2021-06-04

 product 246810 / schaap is still       'fresh'         given the simulation date       (Exp_Date: 2021-07-25)
 product 174773 / kaas has been         expired,        given the simulation date
 product 176792 / worst has been        sold            before the simulation date
 product 141994 / eieren has been       sold            before the simulation date
 product 92362 / toastjes has been      sold            before the simulation date
 product 80145 / wijn has been  expired,        given the simulation date
 product 1284 / brood has been  expired,        given the simulation date
 product 57479 / jam has been   sold            before the simulation date

-----------------------------------------------------------------------------------------------
 id     | product_name | purchase_amount | purchase_price | expiration_date | exit_status         |
-----------------------------------------------------------------------------------------------
 174773 | kaas         | 2.0             | 7.5            | 2021-01-07      | expired (simulated) |
 80145  | wijn         | 6.0             | 6.5            | 2021-02-17      | expired (simulated) |
 1284   | brood        | 7.0             | 8.5            | 2021-05-25      | expired (simulated) |
-----------------------------------------------------------------------------------------------

 Total value loss due to expiration for this simulation: 22.5
```

value loss due to expiration is depicted.


Supermarket_SuperPy
Main.py
- First time run
- Get commands to follow-up steps: purchase, sell, report inventory, expire status and simulate
Simulation_expire.py
- argparse number of days and sim date
> get_expiry_report_ simulated
Report_inventory.py
- relations to files
- counters to zero
- read inventory file
- feedback purchase value, sales value, number of items, value loss due tot expirations and calculated profit of today
> print tabel inventory file
Purchase_input.py
- relations to files
- ask if full routine or default input for product
- input
- unique_id
- open purchase file and add new product
- open inventory file and add new product
- update expire file
- feedback file updates
> print tabel purchase file
Sell2.py
- relations to files
- argparse id, sales price and sold date
- read inventory file
- update exit status to sold
- feedback total sales
- if item was already sold:
- feedback sold before
> print tabel sales file
Unique_id.py
- use random
- typecast expiration_date and purchase_date to integers
- generate unique id
Print_tabel.py
- relations to files
- print tabel based in provided file
- every three line a separation line
Expire.py
- relations to files
X get_expiry_report_today
- set report file
- remove expire file
- read inventory file
- resore headers
- if exit status is not expired or sold
check if product is expired today
if so, change exit status to expired
and update report file
- update report file and inventory file
X get_expiry_report_simulated
- set report file
- set counters
- remove expire file
- read inventory file
- if exit status is sold, report and abort
- if purchase date is later than sim date, report and abort
- if expired at sim date, exit status update and value loss update
- write expires file with report file
> print tabel expire_file
- report value loss from simulation
Init.py
> stockstart
> get_expiry_report_today
> report_inventory
Stockstart.py
> create_directory
- relations to files
- read stock.csv
> unique_id
- write inventory file
- write sales file
- write purchase file
- write expires file
Create_directory.py
- make directory "initial files"
- import stock.csv and create inventory.csv, expires_dates.csv, purchase.csv and sales.csv