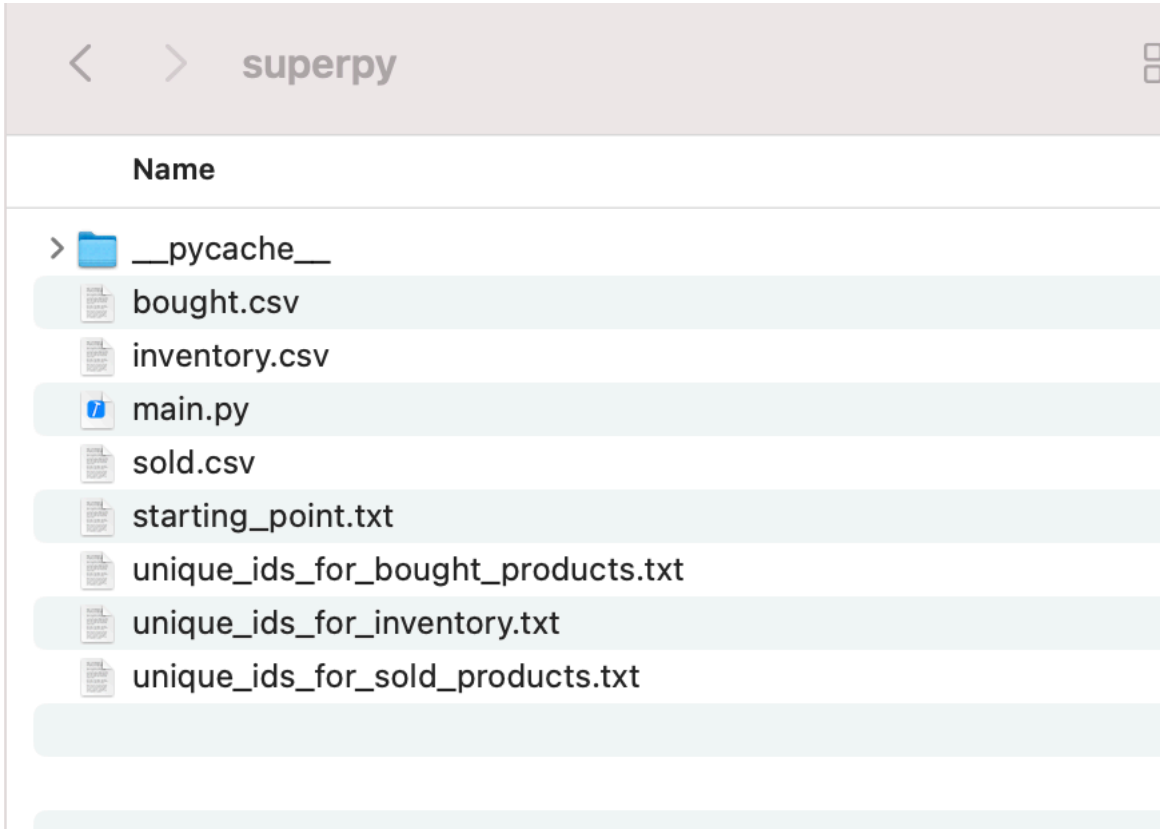
Manual SuperPy Klaas Jan Kostelijk

Okay, let's start! Running 'python3 main.py' in the directory of this assignment will generate all the files you'll need to work with my version of SuperPy.



Let's buy a couple of products!

```
klaas@Klaass-iMac superpy % python3 main.py buy --product_name orange --buying_price 0.1 --expiration_date 2021-07-06
klaas@Klaass-iMac superpy % python3 main.py buy --product_name orange --buying_price 0.1 --expiration_date 2021-07-06
klaas@Klaass-iMac superpy % python3 main.py buy --product_name apple --buying_price 0.3 --expiration_date 2021-07-06
klaas@Klaass-iMac superpy % python3 main.py buy --product_name apple --buying_price 0.3 --expiration_date 2021-07-06
klaas@Klaass-iMac superpy % python3 main.py buy --product_name apple --buying_price 0.3 --expiration_date 2021-07-06
klaas@Klaass-iMac superpy % python3 main.py buy --product_name light_bulb --buying_price 1.0 --expiration_date 2024-08-13
```

Running this command will make sure the purchases are tracked automatically by adding the purchased products to the file 'bought.csv':

bought

id	product_name	buying_date	buying_price	expiration_date
1	orange	2021-06-27	0.1	2021-07-06
2	orange	2021-06-27	0.1	2021-07-06
3	apple	2021-06-27	0.3	2021-07-06
4	apple	2021-06-27	0.3	2021-07-06
5	apple	2021-06-27	0.3	2021-07-06
6	light_bulb	2021-06-27	1.0	2024-08-13

Because a bought product is automatically part of the inventory, these products are automatically added to the file ‘inventory.csv’:

inventory

id	product_name	buying_price	expiration_date
1	orange	0.1	2021-07-06
2	orange	0.1	2021-07-06
3	apple	0.3	2021-07-06
4	apple	0.3	2021-07-06
5	apple	0.3	2021-07-06
6	light_bulb	1.0	2024-08-13

Getting the amount of items per product:

```
klaas@Klaass-iMac superpy % python3 main.py get_count_per_product
Number of products currently in our inventory:
product_name
apple         3
orange        2
light_bulb    1
dtype: int64
klaas@Klaass-iMac superpy %
```

Getting the different products currently present in the inventory:

```
klaas@Klaass-iMac superpy % python3 main.py get_products_currently_in_inventory
Overview of products currently in our inventory:
['orange' 'apple' 'light_bulb']
```

Selling a product requires to check whether or not it’s part of the inventory and whether or not this product isn’t expired. First, I’m going to try to sell chips (something we do not have):

```
klaas@Klaass-iMac superpy % python3 main.py sell --product_name chips --selling_price 2.0
Life's tough sometimes: product not part of the inventory or expired
```

Let’s sell an orange at a ridiculous price:

```
klaas@Klaass-iMac superpy % python3 main.py sell --product_name orange --selling_price 10.0
Product sold
```

inventory

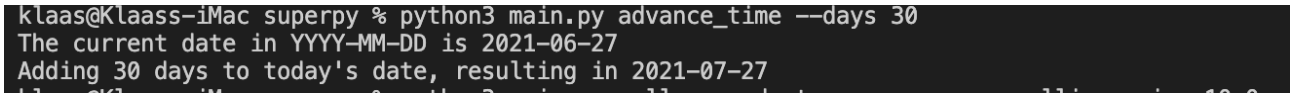
id	product_name	buying_price	expiration_date
2	orange	0.1	2021-07-06
3	apple	0.3	2021-07-06
4	apple	0.3	2021-07-06
5	apple	0.3	2021-07-06
6	light_bulb	1.0	2024-08-13

Now, this orange is no longer part of the inventory:

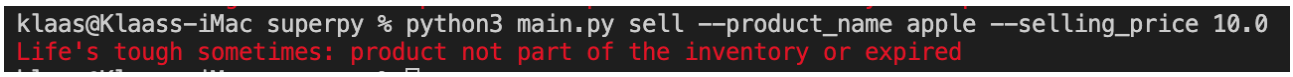
sold

id	product_name	selling_date	selling_price
1	orange	2021-06-27	10.0

This sale is now tracked in the file called ‘sold.csv’:

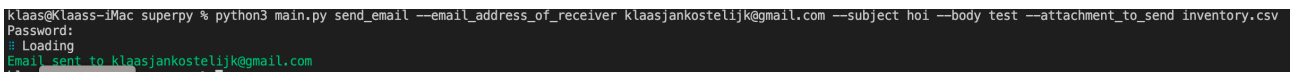


Let’s advance time by 30 days:



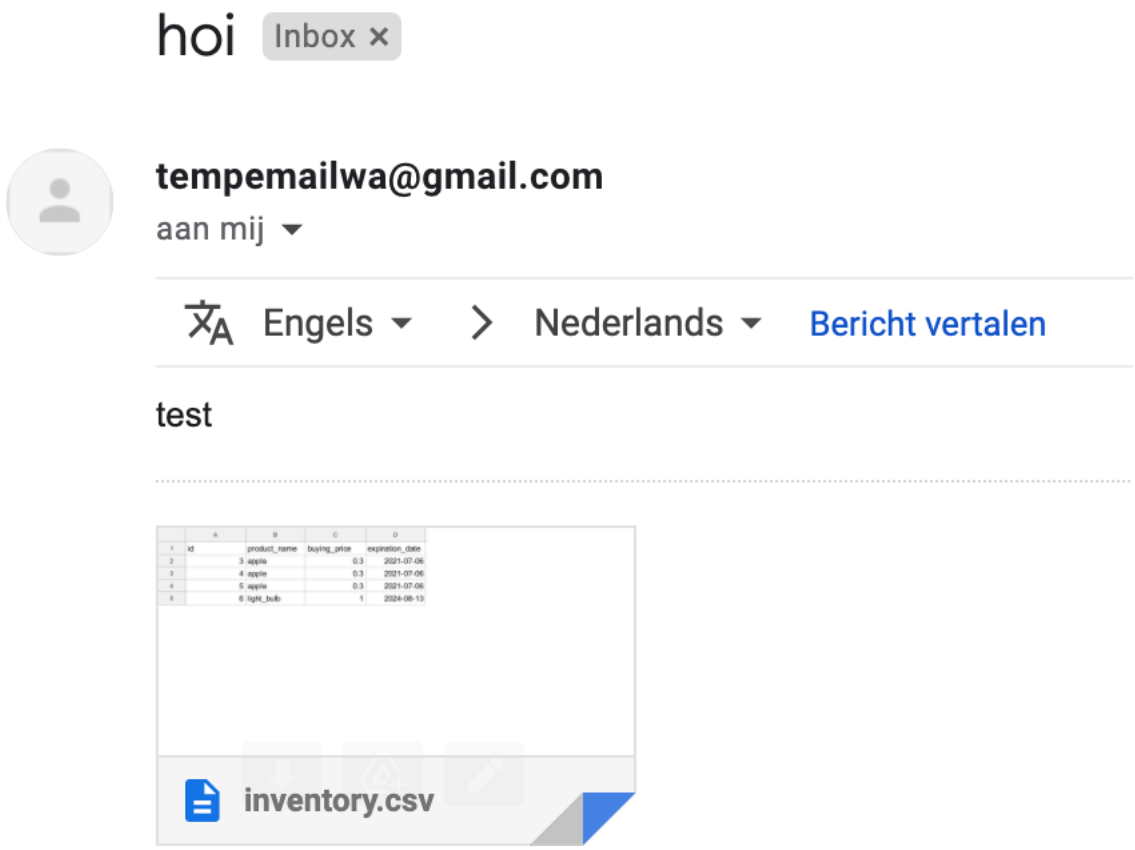
Now, let’s try to sell an apple of which we know it’s expired after advancing time:

You might have noticed the colours (red and green so far). This is the result of using the external module ‘Rich’. I’ve also added the external module called ‘Halo’ inside the function called ‘send_email’. The ability to send emails with my version of SuperPy and the ability to send the .csv files by email. I use Gmail for that. The function is protected by the password mentioned at the start of this document.



Let’s send an email with attachment!

Result:



The application can also keep track of revenue, expenses and profit. Because the idea behind these is the same, I'm only showing profit. This is the difference between revenue minus expenses, either on a specific date or over a specific period.

On a specific date:

```
klaas@Klaass-iMac superpy % python3 main.py report_profit_of_given_day --day 2021-06-27
Today's revenue: EUR 20.0
Expenses: EUR 2.1
Profit: EUR 17.9
```

```
klaas@Klaass-iMac superpy % python3 main.py report_profit_of_given_day --day 2021-06-26
Yesterday's revenue: EUR 0.0
Expenses: EUR 0.0
Profit: EUR 0.0
```

Over a given period:

```
klaas@Klaass-iMac superpy % python3 main.py report_profit_of_time_period --from_date 2021-06-01 --to_date 2021-08-01
Revenue over specified time period: EUR 20.0
Expenses over specified time period: EUR 2.1
Profit over specified time period: EUR 17.9
```

Because the date 2021-06-27 falls within the periode between 2021-06-01 to 2021-08-01, these numbers are the same.