# Technische handleiding SuperPy

Zie ook de Gebruikershandleiding SuperPy

Inhoud

[Technische handleiding SuperPy 1](#_Toc147331733)

[Technische opmerkingen 2](#_Toc147331734)

[Beginsituatie 2](#_Toc147331735)

[Datumformaat 2](#_Toc147331736)

[Gegevensbestanden in csv-formaat 2](#_Toc147331737)

[Rich 2](#_Toc147331738)

[Taal 2](#_Toc147331739)

[Bevindingen 3](#_Toc147331740)

[Functies 4](#_Toc147331741)

In geel: nieuw of aangepast in deze versie.

Onderstaand, in alfabetische volgorde, een aantal [technische opmerkingen](#_Technische_opmerkingen).  
En aan het eind een opsomming van bekende [bevindingen](#_Mogelijke_verbeteringen_1) die nog verwerkt moeten worden.

## Technische opmerkingen

### Beginsituatie

* Alle csv-bestande bevatten 1 rij met kolomnamen, van elkaar gescheiden door een komma.
* Bestand SuperPy.csv bevat onder de kolomrij een tweede rij met de in SuperPy te hanteren “dag van vandaag”.
* De beginsituatie voor het eigen concreet uitgewerkt voorbeeld is in te stellen via de Command line interface (CLI). Type in **python super.py initiate own**

Zie de Gebruikershandleiding voor dit voorbeeld.

### Datumformaat

* In de csv-bestanden en in de CLI worden datums getoond in ISO-formaat yyyy-dd-mm.

### Gegevensbestanden in csv-formaat

* Het bestand Bought.csv en Sold.csv bevat in de eerste kolom resp. het gegeven bought\_id en sold\_id, een integer die begint bij 1 en door het programma steeds met 1 wordt opgehoogd.
* Om mogelijk te maken dat de SuperPy-gegevens eenvoudig met andere programma’s of interfaces gedeeld kunnen worden zijn die altijd opgeslagen in csv-bestanden. Na elk commando door de gebruiker in command line interface ingevoerd of na elke bewerking door SuperPy zelf worden de cs-bestanden daarom bijgewerkt op nieuw opgebouwd.

### Rich

* Aan SuperPy toegevoegd en ingezet op belangrijke boodschappen aan de gebruiker te benadrukken dor die in rood en vet te presenteren.
* Dit wordt bijvoorbeeld gebruikt wanneer getracht wordt een niet-bestaande beginsituatie te initialiseren:  
  .

En ook als de opgevraagde *revenu* of *profit* negatief is.

### Taal

* De beide handleidingen zijn in het Nederlands.
* De producten, tabelnamen, CLI-commando’s en python-code (de toelichting in de code, de boodschappen aan de gebruiker) zijn in het Engels.

# Bevindingen

Nog te doen / mogelijke verbeteringen.

* Import van modules via \* haalt veel meer op dan nodig is.
* Opmaak van de output van de csv-bestanden in functie def show\_csv(csv\_name:str): verbeteren.
* Voor bijna alle functies 1 of meerdere test-functies maken, incl. assert en gebruik pytest.

# Functies

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Onderdeel** | **Functienaam** | **Beschrijving** | **Input** | **Output** | **Onderdeel** | **Test** |
| my\_csv\_handler.py | read\_mytoday()->datetime: | read my\_today from Today.csv | n.a. | return my\_today\_datetime | test\_csv\_handler.py |  |
| my\_csv\_handler.py | modify\_mytoday(shift\_in\_days:int)->datetime: | writing a adjusted my\_today in SuperPy.csv | shift\_in\_days:int | my\_todate changed | test\_csv\_handler.py |  |
| my\_csv\_handler.py | next\_id(csv\_name:str)->int: | counting lines in {csv\_name} | csv\_name:str | return next\_id | test\_csv\_handler.py | def test1\_next\_id(): |
| my\_csv\_handler.py | show\_csv(csv\_name:str): | showing table content {csv\_name} to user. | csv\_name:str | table content showed | test\_csv\_handler.py |  |
| my\_csv\_handler.py | append\_csv(csv\_name:str, new\_row:list): | row {new\_row} will be appended to {csv\_name}. | csv\_name:str, new\_row:list | row added | test\_csv\_handler.py | def test1\_appends\_new\_line(): |
| my\_csv\_handler.py | make\_csv\_inventory\_and\_expired(): | making a new Inventory.csv and Expired.csv | n.a. | tables Inventory.csv and expired.csv new filled | test\_csv\_handler.py |  |
|  |  |  |  |  |  |  |
| my\_initiate.py | initiate(): | all csv-files are reset to startvalue | startsituation:str | print('Finished: SuperPy has been initiated.') | test\_initiate.py |  |
|  |  |  |  |  |  |  |
| my\_functions.py | bought\_id\_sold(find\_bought\_id:int)->bool: | Checks if bought\_id {find\_bought\_id} is sold. | find\_bought\_id:int | return message | test\_my\_functions.py |  |
| my\_functions.py | bought\_id\_expired(find\_bought\_id:int)->bool: | Checks if bought\_id = {find\_bought\_id} is expired. | find\_bought\_id:int | return message | test\_my\_functions.py |  |
| my\_functions.py | product\_id\_in\_csv\_inventory(find\_product\_name:str): | Searches the product\_id of {product\_name} in Inventory.csv. | find\_product\_name:str | return found\_product\_id | test\_my\_functions.py |  |
| my\_functions.py | find\_bought\_id(find\_product\_name:str): | Searches bought\_id for {find\_product\_name} in Bought.csv | find\_product\_name:str | return found\_product\_id | test\_my\_functions.py |  |
| my\_functions.py | total\_sold\_price(date:datetime): | Calculates the total\_sold\_price up to and including the specified date | date:datetime | return total\_sold\_price | test\_my\_functions.py |  |
| my\_functions.py | total\_bought\_price(date:datetime): | Calculates the total\_bought\_price up to and including the specified date | date:datetime | return total\_bought\_price | test\_my\_functions.py |  |
| my\_functions.py | total\_expired\_price(date:datetime): | Calculates the total\_expired\_price up to and including the specified date | date:datetime | return total\_expired\_price | test\_my\_functions.py |  |
| my\_functions.py | revenue(date:datetime): | In SuperPy 'revenue' is defined as: the total price of all (and only) SOLD products | date:datetime | return revenue | test\_my\_functions.py |  |
| my\_functions.py | profit(date:datetime): | In SuperPy 'profit' is defined as: the total price of (all sold products - bought products - expired products). | date:datetime | return profit | test\_my\_functions.py |  |
| my\_functions.py | this\_date\_total\_sold\_price(date:datetime): | Calculates total\_sold\_price on the specified date | date:datetime | return this\_date\_total\_bought\_price |  |  |
| my\_functions.py | this\_date\_total\_bought\_price(date:datetime): | Calculates the total\_bought\_price on the specified date | date:datetime | return this\_date\_total\_bought\_price |  |  |
| my\_functions.py | this\_date\_total\_expired\_price(date:datetime): | Calculates the total\_expired\_price on the specified date | date:datetime | return this\_date\_total\_expired\_price |  |  |
| my\_functions.py | this\_date\_revenue(date:datetime): | In SuperPy 'revenue' is defined as: the total price of all (and only) SOLD products | date:datetime | return this\_date\_revenue | test\_my\_functions.py |  |
| my\_functions.py | this\_date\_profit(date:datetime): | In SuperPy 'profit' is defined as: the total price of (all sold products - bought products - expired products). | date:datetime | return this\_date\_profit | test\_my\_functions.py |  |

.