Bakery cash registry

My solution to this problem is given in the given archive with following structure:

```
➜ Rubix tree
.
├── bakery.pdf
├── bakery_solution.pdf
├── bakery_solution.py
├── __init__.py
├── lookup_table.py
└── tests
    ├── __init__.py
    └── test_one.py

1 directory, 7 files
➜ Rubix
```

bakery.pdf	problem statement
bakery_solution.py	is the main file
lookup_table.py	contains product table as per problem statement (database proxy)
tests/test_one.py	unittets

Execution example is given in following snippet.

RUN IN PYTHON2

Only pytest needs to be installed

User is required to enter product code and product quantity separated by comma, without spaces.

```
➜ ~ python bakery.py
Hi, in our bakery you can purchase following products:

Croissant                       code = CF
Blueberry Muffin                code = MB11
Vegimite Scroll                 code = VS5

Please enter product code and product quantity.
When you are done with your orders press spacebar.
Enter product code and quanyity separated by comma: VS5,10
Enter product code and quanyity separated by comma: MB11,23
Enter product code and quanyity separated by comma: CF,12
Enter product code and quanyity separated by comma:

10 VS5 $17.980000
        2 x 5 $8.99
12 CF $22.940000
        1 x 3 $5.95
        1 x 9 $16.99
23 MB11 $76.200000
        2 x 8 $24.65
        1 x 5 $16.95
        1 x 2 $9.95
➜ ~
```

The code can take care the following input format errors.

Wrong product code.

```
➜  ~ python bakery.py
Hi, in our bakery you can purchase following products:

Croissant                       code = CF
Blueberry Muffin                code = MB11
Vegimite Scroll                 code = VS5

Please enter product code and product quantity.
When you are done with your orders press spacebar.
Enter product code and quanyity separated by comma: __CF xxx Z
Incorrect product code. Retype your order or press Enter to finish:
Enter product code and quanyity separated by comma:

➜  ~
```

Wrong product quantity.

```
➜  ~ python bakery.py
Hi, in our bakery you can purchase following products:

Croissant                       code = CF
Blueberry Muffin                code = MB11
Vegimite Scroll                 code = VS5

Please enter product code and product quantity.
When you are done with your orders press spacebar.
Enter product code and quanyity separated by comma: CF,----3.4rrr
Incorrect product quantity. Type integer or press Enter to finish:
Enter product code and quanyity separated by comma:

➜  ~
```

The code can sum up product code entered twice.

```
➜  ~ python bakery.py
Hi, in our bakery you can purchase following products:

Croissant                       code = CF
Blueberry Muffin                code = MB11
Vegimite Scroll                 code = VS5

Please enter product code and product quantity.
When you are done with your orders press spacebar.
Enter product code and quanyity separated by comma: CF,12
Enter product code and quanyity separated by comma: CF,12
Enter product code and quanyity separated by comma:

24 CF $43.930000
        1 x 5 $9.95
        2 x 9 $16.99
➜  ~
```

Unittest example.

Testing one particular order sequence - 10 VS5

```
➜  Rubix python -m pytest tests
============================= test session starts ==============================
platform linux2 -- Python 2.7.12, pytest-2.8.7, py-1.4.31, pluggy-0.3.1
rootdir: /home/igor/Dropbox/DATA/Job_ML_homeworks/Rubix/tests, inifile:
collected 2 items

tests/test_one.py ..

=========================== 2 passed in 0.02 seconds ===========================
➜  Rubix
```

Summary.

Unfortunately this code is not universal. It couldn't correctly solve order sequences from problem statement (12 CF and 14 MB11).
It works well when it is possible to use biggest batch in the price breakdown.

Also, given the batch constraint, for arbitrary order exact solution is not always possible. In such cases **bakery_solution.py** produces price breakdown that is within **[+-smallest_batch] from initial order**. It is possible to correctly calculate change in such cases.