Mock exam for p4p

You are tasked with writing a command line tool that simulates a vending machine. This machine should be able to stock an arbitrary number of products, each with its own name, price and available quantity. Each product should be assigned a number such that, like with a real vending machine, the customer can easily choose what to buy. Since the vending machine has to be able to hand out change, it should keep track of how much change it currently has.

Furthermore, there should be a nice interface printed to the console that contains information about product numbers, names, prices and stocks and that lets the user insert an arbitrary amount of money and make a selection from the list of products. When the user chooses to buy a product, the vending machine should perform all the necessary checks in order to ensure a valid purchase. Specifically, the purchase is considered invalid whenever the customer

- provides too little money,
- provides too much money and the vending machine does not have enough change to hand out,
- chooses an item that is out of stock or
- chooses a non-existing product id.

In all of these cases, the customer should be notified of their mistake. Whenever a product is successfully bought, the customer should be presented with their change and the product's stock as well as the machine's available change should be updated. The vending machine should run indefinitely, meaning that the program only ends when the user presses Ctrl+C, allowing customers to repeatedly make purchases. Please refer to the section *Example run* in order to get an idea of what the program could look like.

For your implementation, you must use an **object oriented** approach. This means that, apart from a *main* function that handles user input and output, all of the program's logic must be contained in classes. **Important:** Your main function must be inside of a file called *main.py* while any classes you write must be in (a) different file(s).

Allowed/forbidden tools and material

You are permitted to use any online/offline resources you like, **except**

- Generative AIs like ChatGPT, GitHub Copilot, etc.
- Communication with other people

Whenever you use someone else's code and said code has more than one line, make a note in the README file about where you found it (e.g. URL, book, script, etc.)

Grading

Everything above this line constitutes the program's **basic functionality** and has to be correctly implemented in order for you to pass the exam. It contributes 50/100 points to your total score. If there is a small mistake (e.g. a typo) that prevents (part of) the program from properly running, no points will be deducted.

Including the following **additional features** in your solution will give you the specified number of points (partial points are of course possible):

- Implementing at least one meaningful getter and setter [10/100 Pts.]
- Products (names, prices, stocks) and change are loaded from a JSON file called *products.json* when the program starts. When the program ends, the updated product stocks and the new change are saved back to the file. See the section *Example products.json* for an example file. You may implement the saving/loading inside the main.py file or inside of a class. [10/100 Pts.]
- Proper error handling (user input is checked and the program doesn't crash or end on illegal input/keyboard interrupt, the user is asked to re-enter illegal input instead of having to restart the program) [10/100 Pts.]
- Implementing at least three meaningful tests as seen in the exercises [10/100 Pts.]
- The code is documented and well organized [5/100 Pts.]
- The presence of an AUTHORS, LICENSE and README file with meaningful content (README file is **not** optional if you use other people's code, see above) [5/100 Pts.]
- *(Bonus)* Meaningful implementation of the __str__ method in at least one class [5 bonus Pts.]

Example run

This in an example run of the program, meant to give you an idea of what your solution could look like. It implements all the **basic functionality** required to pass the exam.

```
>>> python main.py

Vending machine products
------------------------

[1] Premium Cola: 1.5 Euro (10 available)
[2] Makava: 1.4 Euro (3 available)
[3] Wostok: 0.8 Euro (5 available)
[4] Club Mate: 1.0 Euro (1 available)


Please insert money.
> 2
What would you like to buy?
> 4

One Club Mate, there you go! Your change is 1.0 Euro.

Vending machine products
------------------------

[1] Premium Cola: 1.5 Euro (10 available)
[2] Makava: 1.4 Euro (3 available)
[3] Wostok: 0.8 Euro (5 available)
[4] Club Mate: 1.0 Euro (Out of stock)

```

```
27
28 Please insert money.
29 > 2
30 What would you like to buy?
31 > 4
32
33 The product Club Mate is out of stock
34
35 Vending machine products
36 ------------------------
37
38 [1] Premium Cola: 1.5 Euro (10 available)
39 [2] Makava: 1.4 Euro (3 available)
40 [3] Wostok: 0.8 Euro (5 available)
41 [4] Club Mate: 1.0 Euro (Out of stock)
42
43
44 Please insert money.
45 > 1
46 What would you like to buy?
47 > 1
48
49 The price of Premium Cola is 1.5 Euro, but you have only given 1.0 Euro
50
51 Vending machine products
52 ------------------------
53
54 [1] Premium Cola: 1.5 Euro (10 available)
55 [2] Makava: 1.4 Euro (3 available)
56 [3] Wostok: 0.8 Euro (5 available)
57 [4] Club Mate: 1.0 Euro (Out of stock)
58
59
60 Please insert money.
61 > 100
62 What would you like to buy?
63 > 1
64
65 Sorry, I don't have enough change left for the 100.0 Euro you gave me
66
67 Vending machine products
68 ------------------------
69
70 [1] Premium Cola: 1.5 Euro (10 available)
71 [2] Makava: 1.4 Euro (3 available)
72 [3] Wostok: 0.8 Euro (5 available)
73 [4] Club Mate: 1.0 Euro (Out of stock)
74
75
76 Please insert money.
77 > 1
78 What would you like to buy?
79 > 5
80
81 The product with id 5 does not exist
82
83 Vending machine products
84 ------------------------
85
86 [1] Premium Cola: 1.5 Euro (10 available)
87 [2] Makava: 1.4 Euro (3 available)
88 [3] Wostok: 0.8 Euro (5 available)
89 [4] Club Mate: 1.0 Euro (Out of stock)
90
91
92 Please insert money.
93 > (User presses Ctrl+C)
94 Bye!
```

Example products.json

This is an example products.json file:

```
{
    "change": 5.0,
    "products": [
        {"name": "Premium Cola", "price": 1.5, "stock": 10},
        {"name": "Makava", "price": 1.4, "stock": 3},
        {"name": "Wostok", "price": 0.8, "stock": 5},
        {"name": "Club Mate", "price": 1.0, "stock": 1}
    ]
}
```

Example project structure

This is an example for your folder structure, assuming that you implement everything in the *additional features* list.

```
.
├── AUTHORS
├── LICENSE
├── main.py
├── product.py
├── products.json
├── README
├── test_product.py
└── vending_machine.py
```

The only files that **have to** be present are

- `main.py` containing the main method for user input/output and, optionally, functions for loading/saving products.json
- `some_file_name.py` containing your class(es), in this example `product.py` and `vending_machine.py`
- `README` in case you use other people's code