Paper 2 Practice: Bookstore Inventory

1a. A small bookstore keeps its inventory in a text file (`books.txt`). Each line contains a book title and its price in dollars, separated by a comma. Example line: `"The Great Gatsby,12.99"`

The manager wants a program that:

- Reads the book data from the file into two parallel arrays:
  - `tbl_title` (book titles) and `tbl_price` (prices as `float`).
- Displays a table showing:
  - A header row and a dividing line
  - Each book title, its original price, and the price after applying a 10% discount (shown as whole dollars – round down to nearest integer)
  - The overall average price (original) of all books, shown with two decimal places
- Allows the user to enter a book title and a new price.
- If the title is found, its price is updated.
- If not found, an error message is shown.
- After any update, the program writes the entire updated inventory back to the same file (overwriting the old data).
- Contains at least one subprogram
- The program must work correctly even if the number of books in the file changes.

Figure 1: Example of display table

```
Book Title                Original Price   Discounted ($)
-------------------------------------------------------------
The Great Gatsby               12.99            11
To Kill a Mockingbird          14.50            13
1984                            9.99             8
Pride and Prejudice            11.25            10
-------------------------------------------------------------
Overall average price:         12.18
```

Hint: the `split()` string method converts a string to a list of string. By default, it will split on whitespace, but if a parameter is provided, it splits on that string, removing the string. For example, we can split on a comma:

```
fruit_string = "apples,oranges,bananas"
print(fruit_string.split(','))
```

```
['apples', 'oranges', 'bananas']
```

Paper 2 Practice: Bookstore Inventory

Hint: do not waste time on formatting the columns until you have completed the main functionality.

To format columns, you may use f-strings. To create an f-string, the quotation mark is preceded by the letter `f`, variables are enclosed in curly braces, `{` and `}`, and the number of spaces a variable takes is placed after the variable and a colon, (:).

```
students = [ 'Alice', 'Allen' ]
scores = [ 9.81, 9.9 ]

print('_' * 22)

for i in range(len(students)):
    print(f"| {students[i]:10} | {scores[i]: 5} |")

print('-' * 22)
```

```
______________________
| Alice      |  9.81 |
| Allen      |   9.9 |
----------------------
```

By default, strings are aligned left, while numbers are aligned right. If you're interested, you can look up how to change this.