You’ve recently been hired as a cashier at the local sewing hobby shop, ***Thread Shed***. Some of your daily responsibilities involve tallying the number of sales during the day, calculating the total amount of money made, and keeping track of the names of the customers.

Unfortunately, the ***Thread Shed*** has an extremely out-dating register system and stores all of the transaction information in one huge unwieldy string called daily\_sales.

All day, for each transaction, the name of the customer, amount spent, types of thread purchased, and the date of sale is all recorded in this same string. Your task is to use your Python skills to iterate through this string and clean up each transaction and store all the information in easier to access lists.

If you get stuck during this project or would like to see an experienced developer work through it, click “**Get Help**“ to see a **project walkthrough video**.

Tasks

22/22Complete

Mark the tasks as complete by checking them off

**Break up `daily\_sales` in easy to understand lists `customers`, `sales`, and `threads\_sold`.**

1.

First, take a minute to inspect the string daily\_sales in the code editor.

How is each transaction stored? How is each piece of data within the transaction stored?

Start thinking about how we can split up this string into its individual pieces of data.

2.

It looks like each transaction is separated from the next transaction by a ,, and then each piece of data within a transaction is separated by the artifact ;,;.

If we want to split up daily\_sales into a list of individual transactions, we are going to want to split by ,, but first, we need to replace the artifact ;,; to something *without* a comma, so we don’t split any transactions themselves.

Replace all the instances of ;,; in daily\_sales with some other character and save the result to daily\_sales\_replaced.

3.

Now we can split the string into a list of each individual transaction.

Split daily\_sales\_replaced around commas and save it to a new list daily\_transactions.

4.

Print daily\_transactions.

How does it look?

5.

Our next step is to split each individual transaction into a list of its data points.

First, define an empty list daily\_transactions\_split

6.

Now, iterate through daily\_transactions (remember, this is a list of strings currently), and for each transaction, split the string around whatever character you replaced the ;,; artifacts with in **Step 2**.

Append each of these split strings (which are lists now) to our new list daily\_transactions\_split.

7.

Print daily\_transactions\_split.

How’s it looking?

8.

It looks like each data item has inconsistent whitespace around it. First, define an empty list transactions\_clean.

Now, Iterate through daily\_transactions\_split and for each transaction iterate through the different data points and strip off any whitespace.

Add each of these cleaned up transactions to the new list transactions\_clean.

9.

Print transactions\_clean.

If you performed the last step correctly, you shouldn’t see any unnecessary whitespace.

10.

Create three empty lists. customers, sales, and thread\_sold. We are going to collect the individual data points for each transaction in these lists.

11.

Now, iterate through transactions\_clean and for each transaction:

1. Append the customers name to customers.
2. Append the amount of the sale to sales.
3. Append the threads sold to thread\_sold.

12.

Print customers, sales, and thread\_sold to make sure each list is what you are expected.

**Determine the total value of the days sales.**

13.

Now we want to know how much *Thread Shed* made in a day.

First, define a variable called total\_sales and set it equal to 0.

14.

Now, consider the list sales. It is a list of *strings* that we want to sum. In order for us to sum these values, we will have to remove the $, and set them equal to floats.

Iterate through sales and for each item, strip off the $, set it equal to a float, and add it to total\_sales

15.

Print total sales.

How much did we make today?

**How much thread of any specific color was sold?**

16.

Finally, we want to determine how many of each color thread we sold today. Let’s start with a single color, and then we’ll generalize it.

First, print out thread\_sold and inspect it.

17.

We see that thread\_sold is a list of strings, some are single colors and some are multiple colors separated by the & character.

The end product we want here is a list that contains an item for each color thread sold, so no strings that have multiple colors in them.

First, define an empty list thread\_sold\_split.

18.

Next, iterate through thread\_sold. For each item, check if it is a single color or multiple colors. If it is a single color, append that color to thread\_sold\_split.

If it is multiple colors, first split the string around the & character and then add each color indivudally to thread\_sold\_split.

19.

Great, now we have a list thread\_sold\_split that contains an entry with the color of every thread sold today.

Define a function called color\_count that takes one argument, color. The function should iterate through thread\_sold\_split and count the number of times the item is equal to argument, color. Then, it should return this count.

20.

Test your new function by running color\_count('white').

Your function should return 28.

21.

Define a list called colors that stores all of the colored threads that *Thread Shed* offers:

py colors = ['red','yellow','green','white','black','blue','purple']

22.

Now, using the list colors, the string method .format(), and the function color\_count, iterate through thread\_sold\_split and print a sentence that says how many threads of each color were sold today.