Shopping Lists

The aim of this exercise is to use your knowledge of lists, functions and strings to produce a program that can read a shopping list and then tell you what items you have left to buy.

A shopping list looks like this:

```
Mints x
Apples x
Mangoes x
Shampoo /
Steam-Voucher /
```

The `x`s (crosses) represent that we have not bought the item, the `/`s (ticks) represent we have already got the item. In this exercise, we call these ticks and crosses *marks.*

In python, this is represented as a string:
`"Mints x\nApples x\nMangoes x\nShampoo /\nSteam-Voucher /"`
Where `\n` represents a new line.

After you have implemented all of the functions in the first exercise, you will have a program that can print:

```
You need to get:
Mints
Apples
Mangoes
```

When given the shopping list above

Helpful Functions

To help you with this task, I have given you some handy functions:

`split_string(s, character)` which can separates a string `s` into smaller strings by using `character` as the character to separate on.
For example: `split_string('hello there. How's it going', '. ')` will return:
`['hello there', 'How's it going']`

`get_shopping_list()` which you can use in your REPL to manually enter a shopping list. I will show you how this works in class.

`shopping_list_program()` Which runs the full program you have created. It uses **`get_shopping_list`** to read a shopping list, and then prints the items you have yet to buy.

Tests

To know if your code is correct, I have written some tests. When you run the python script, you should see the tests running before the REPL starts. For example, you might see something like this:

```
testing split_line_into_item_and_mark:
ERROR: split_line_into_item_and_mark('oranges /') returned ['', ''],
when it should return ['oranges', '/']
ERROR: split_line_into_item_and_mark('apples x') returned ['', ''],
when it should return ['apples', 'x']
DONE
…
```

Which shows that the implementation of `split_line_into_item_and_mark` is incorrect and why. These should be very helpful when trying to understand why your code might not be working. When a function has all of its test passing, you should see no more error messages, something like this:

```
testing split_line_into_item_and_mark:
DONE
```

Main Exercise

I have written out some functions for you to implement. You may do them in any order you wish, but you may find some of the first functions will be helpful in the later ones. Some of these functions may only need one line of code, some of them might need quite a few.

Below are the description of each function you need to implement

`split_line_into_item_and_mark(line)`
This function takes a line of the shopping list (e.g. `'apples /'`) and returns an array which is made up of `[<item-name>, <mark>]`, for the previous example: `['apples', '/']`. Hint: Use the `split_sting()` function here.

`mark_to_bool(mark)`
This function takes a mark (i.e `'/'` or `'x'`) and returns `True` if the mark is a `'/'` and `False` if the mark is a `'x'`. This will be useful in implementing `filter_out_ticked_items`

`filter_out_ticked_items(items, marks)`
This function takes two arguments: a list of items and a corresponding list of marks, and returns a list of items which have not been bought. The `marks[i]` is the mark on the shopping list for `items[i]`. HINT: use **`mark_to_bool`** to get the boolean value for the mark.

`get_list_of_items_to_get(shopping_list)`
This is the function where you put all of your work together. I recommend finishing all of the previous functions before trying this. `Shopping_list` is a string that represents the shopping list. This function returns a list of the items in the shopping list that have not already been bought.
HINT: think about how you might use the previous functions in this one, try not to copy code from other functions, but use them instead
ANOTHER HINT: How might you split the `shopping_list` string into the lines of the shopping list (use `split_string`)

When all of the tests pass, try out your code by running **`shopping_list_program()`** in the REPL.

Extension Exercise - Dictionaries

This extension excercise makes the shopping list program have even more functionality by using dictionaries instead of lists.

`turn_shopping_list_to_dict(items, marks)`
This functions takes the same input as **`filter_out_ticked_items.`** It returns a dictionary which given an item, returns a boolean value (True or False) representing if we have bought the item yet.
HINT: Use **`mark_to_bool`**

`buy_item(shopping_dict, item)`
This function returns `shopping_dict,` but with `item` is now set to True. <>

`make_shopping_list_string(shopping_dict)`
This function re-builds the shopping list string from **`shopping_dict.`**

`get_unbought_items(shopping_dict)`
This function has the same use as **`get_list_of_items_to_get,`** but takes the dictionary version of the shopping list.