Project #7 – Inventory Data Fixup

1 Problem Overview

You are again doing consulting doing work for Angie's Autos, a small car dealership. An earlier failed attempt to hack together some computer-based parts inventory data has left some messy files that Angie wants you to clean up. In preparation for that, you'll need to break apart the unwieldy strings and clean up some errors that exist in the data.

2 User Interface

2.1 Input

Since we don't yet know about files, we'll just paste in sample strings representing inventory data:

```
      Enter inventory string:
```

This is a good sample to get you started (*hint: this is not all the test cases you will need*):

```
*PART*DRPNO432DriverDoorPanel *2*Shop 1*$1,012.84*1,499.99*Chevy*Camaro*2011-2013*
```

2.2 Output

The program breaks up the data into "fields" (separate kinds or columns of data), fixes up data as shown below and displays the field data. The sample above would yield this output:

```
Type:       Part
Part#:      DRPN0432
Desc:       Driver Door Panel
Qty:        2
Loc:        Shop1
Cost:       1012.84
Price:      1499.99
Make:       Chevy
Model:      Camaro
Year Start: 2011
Year End:   2013
```

3 Calculations

Here are the rules you will need, to transform the flawed data into the desired form:

3.1 Field Separator

In general, fields are demarcated by asterisks, though in a few cases we'll want a further breakdown. The first field represents the type of data; at present, Angie has only Part data.

3.2 Field Specifics

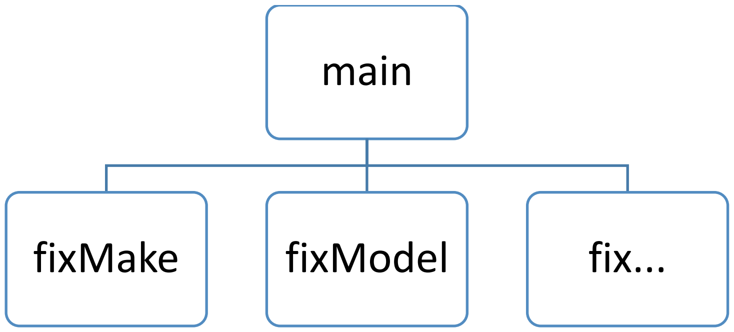
Here are further details on each field:

Field	**Description**	**Issues to Fix**
Type	Type of data this string represents. Currently, these are all "PART"	The data is in all caps; Angie wants this "proper" cased, with only the first letter capitalized
Part#	Angie's part number for the item, consisting of four alphabetic character followed by four numeric characters	Two fields are crammed into one. The Part Number is the first eight characters. There was bad data entry on these, with some O's where there should be zeroes; fix those
Description	Description of the item	Second field that's crammed; everything after the initial eight characters is Description. Some of this data has leading or trailing spaces that need to be removed. Also, no spaces were used, which needs to be fixed; each capital letter needs to be preceded by a space, e.g., 'HelloWorld' becomes 'Hello World'
Quantity*	Quantity on hand	Turn this into a number in case we need to do math on it later
Location	Where the item is located	Any spaces in this data need to be deleted, so 'Shelf 1' becomes 'Shelf1'
Cost*	Angie's cost for the item	Some of these have leading dollar signs; these need to be deleted. Some have embedded commas; we need those gone as well. Turn this into a number.
Price*	Angie's selling price	Same as Cost
Make	Make of the car the part is for	(no known issues at this point)
Model	Model of the car the part is for	(no known issues at this point)
Year Start*	Starting model year the part is for	Two fields are crammed into one; the data has a four-digit starting year, a hyphen, then a four-digit ending year; these need to be separated out and turned into numbers for computation purposes
Year End*	Ending model year the part is for	Second field that's crammed; read Year Start issues

* Turn all numeric-only strings into numbers of the appropriate type, e.g., Year Start should be an integer after processing.

4 Functions

4.1 Call Hierarchy



Each field should have its own “fix” function. Adding helper functions is allowed.

5 Code Specifications

- At the bottom of the program you should have a call to main.
- Include header comments at the top of each file. Include your name, the date, and a brief description of what the program does.
- Include comments for each section saying what’s going on in the lines of code below.
- Use comments elsewhere as you think they help guide the reader. Don’t overdo, though! Not every line needs a comment; think about describing a block of related code.
- Use blank lines to separate sections and provide visual “breathing room.”
- Use descriptive variable and function names.

6 Hints

- There are some fields that require no “fixing,” at least not at this point. For consistency’s sake and because we could learn new requirements for these fields later, create a fix function for each field anyway.
- Write pseudocode so you can get clear on what needs to be done to each field. Pseudocode can often turn into useful code comments; you can start your internal documentation that way.
- There are cases where you’ll need to make several transformations on one field. Think about how to do this one step at a time. You can continue to modify the data as you go along.
- There are a bunch of string functions and methods you can choose from. My solution used these (though yours might find other clever ways): .isupper, .lower, .replace, .split, .strip, .upper, concatenation (using +), conversion of strings to integers and floats, and slicing.

7 Testing

- Develop an appropriate number of test cases.
- Document your testing and results in comments at the bottom of the program as shown below.

8 Summary

At the bottom of your program, add comments that answer these questions:

- How did you approach this assignment? Where did you get stuck, and how did you get unstuck?
- How did you test your program? What doesn't work as you'd like, perhaps things that you'd like to fix as you learn more?
- What did you learn from this assignment? What will you do differently on the next project?

9 Extra Credit

Create a PyUnit test script called TestInvFixup.py. In it, create an automated test case for each of the fix-up functions. Each test case might require multiple assert statements as you'll want to make sure the function works properly in multiple scenarios. Use the materials in the Canvas Resource module (at the top), including the Introduction to Software Testing document, and the two linked videos that demonstrate how to construct automated tests.

10 Grading Matrix

Area	**Pct**
Setup/looping	10
Fix type	5
Fix part number	15
Fix description	15
Fix quantity	5
Fix location	5
Fix cost/price	10
Fix and separate start/end years	10
Fix functions for non-problematic fields	5
Test cases	10
Comments, variable names, white space	5
Summary report	5
Extra Credit	5
Total	**105**