

# Assignment 3

Due Friday, June 25th, 11:59 pm

## Reading

Read the following sections in **Introduction to Computing using Python: An Application Development Focus, Second Edition** by Ljubomir Perković:

1. 4.2 - Formatted output
2. 4.1 - Slicing
3. 4.1 - String functions
4. 4.3 - file I/O
5. debugging (case study starting on page 471)
6. 4.4 - Exceptions

## Logistics

Please document any collaboration relevant to this assignment, in the header comment of the source file, as described in the first assignment.

Complete the problems described below and upload your version of the solutions to the submission folder named 'Homework 3', in D2L.

## Assignment

Implement the functions below in the file **csc401-homework3.py** which can be found on [the D2L site](#) in the Week 3 folder. You should save the template file I provided and then modify that file by adding the bodies for the functions. When you do, make sure to remove the placeholder pass statements that are currently there.

A note about the built-in tests: these should not be your only testing. I will test your code with other inputs besides those shown in the test and your code should still work if I have used valid values.

1. Implement a function named **print\_columns(data)** that takes 1 parameter:
  - a. data - a list contains strings, each string representing a data record

Each record represents a name, a price and a total number of items purchased as a single comma separated string.

The exercise is to:

- split the record into its 3 parts
- convert the data where necessary (price should be a float and number of items an integer).
- Trim the name field to just the first 4 characters
- Calculate the total for each line ( cost multiplied by items)
- print** out the records formatted as below:

Name	Cost	Items	Total
Laa	208.10	10	2081.00
Arno	381.00	9	3428.99
Sion	327.01	1	327.01
Shay	429.50	2	859.00
Rene	535.29	4	2141.16

Name : left justified in 20 places

Cost : right justified in 6 places, 2 decimal places

Items : right justified in 6 places

Total: right justified in 8 places, 2 decimal places

Some fiddling with the headers, spaces between the captions may be necessary. Names should line up under “Name”, costs should line up under “Cost”, totals should line up under “Total”.

Examples:

```
>>> print_columns(sample_data)
Name          Cost      Items  Total
Laa           208.10     10  2081.00
Arno          381.00      9  3428.99
Sion          327.01      1   327.01
Shay          429.50      2   859.00
Rene          535.29      4  2141.16
```

Notes

- The test data **sample\_data** is in **csc401-homework3.py** already.
- The captions are all left justified
- Write a function **scan\_for(file\_name, word)** that takes 2 string parameters:  
**file\_name** - a string representing a file name  
**word** – a word to scan for in the file

The function reads the file and **prints** the number of occurrences of the specified word that exist in the file, in the format displayed below in the example output.

The function must ignore any of these punctuation marks: . ! ?  
That is, 'cat' and 'cat?' are 2 instances of 'cat'. You are permitted to remove the punctuation from the text

The following is sample output using a file available with the homework template:

```
>>> scan_for('shoes.txt', 'I')
The word I appears 5 times in shoes.txt
>>> scan_for('shoes.txt', 'shoes')
The word shoes appears 3 times in shoes.txt
>>> scan_for('shoes.txt', 'tree')
The word tree appears 0 times in shoes.txt
```

4. Write a function **calc\_tax(in\_file, out\_file)** that takes two string parameters:
- **in\_file** – name of a file to read
  - **out\_file** – name of a file to write results to

The input file contains lines with the following format:

**id      amount      tax\_rate**

**id** is a record identifier that can be ignored

**amount** is a dollar amount we will calculate a tax on

**tax\_rate** is the rate at which we will tax the **amount**

The goal of the function is to calculate the tax for each line in the input file and write the result to the output file.

The tax is simply  $\text{amount} * \text{tax\_rate}$

See **two\_files\_demo.py** for an example of how to open 2 files using the **with** command. If you choose to not use the **with** command, remember to close both files.

For testing purposes, the homework template file has a function named **show\_file**. You can use it in your testing but don't change it as my tests need it.

Example:

```
>>> calc_tax('data.txt', 'tax1.txt')
>>> show_file('tax1.txt')
10.00
 9.94
119.46
 7.44
23.81
<BLANKLINE>
```

Also note that “<BLANKLINE>” should NOT be in your output literally. <BLANKLINE> indicates that there is a blank line after the last number.

## Grading

Points for these problems will be allocated as follows:

print_columns	33.3 points
scan_for	33.3 points
calc_tax	33.3 points