

# Assignment 4

Due Monday, June 28th, 5:30 pm

## Reading

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

- 5.2 - Counter loops
- 5.2 - Accumulators loops
- 5.4 - While loops
- 12.3 - List comprehensions

## 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 'Assignment 4', in D2L.

## Assignment

Implement the functions below in the file **csc401-homework4.py** which can be found on [the D2L site](#) in the Session 4 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.

## Problem descriptions

1. Implement a function named **convert\_2\_km()**, which has no parameters. It converts user input from miles into kilometers. The formula for this conversion is:

1 mile == 1.60934 kilometers.

### Steps

- a. In a loop, prompt the user for a value in miles
- b. If the user enters "STOP", exit the function but DO NOT use the exit() function
- c. if the user enters a valid number, show the conversion
- d. if the user enters an invalid value, show an error message and prompt again

### Example of usage

```
>>> convert_2_km()
Miles (or STOP): 12
Kilometers: 19.31208
Miles (or STOP): cats
Bad input: cats
Miles (or STOP): 3
Kilometers: 4.82802
Miles (or STOP): stop
Bad input: stop
Miles (or STOP): STOP
```

Please notice that this function has no self test.

2. Implement a function **meal\_price(items)** that calculates and **returns** the price for a meal. Its only parameter is a list containing meal item names. The goal of the function is to compute the price of the meal. The only items the function should recognize are:

```
apple : 0.59
burger : 2.50
drink : 0.99
fries : 1.29
```

If the list contains anything else, print "Unknown item: \_\_\_\_" and include the item name that is unknown. An unknown item should not stop the process of adding up the price.

Review the test file for example usage.

3. Implement a function **print\_change(row)** that calculates and **prints** the difference between adjacent values in a list. The differences are printed separated by commas. For example, if row is [1,2,9], the output should be "1,7" (not including the quote marks) because 2 compared to 1 is 1 greater and 9 compared to 2 is 7 greater. Negative results are expected. If the input was [1,9,2], the expected result is "8,-7".

The length of the list should not matter. You do not need to test for a list that is too short.

See the test file for more examples.

4. Implement a function **sum\_nums(a\_list)** that sums the numbers from the provided list and **returns** the result. Note that all the values in the list may not be numeric and therefore need to be skipped. For example, `sum_nums([5,'11',27,3,'cat'])` should return 35. The values '11' and 'cat' are skipped.

Use a list comprehension to solve this problem.

See the test file for more examples.

## Grading

Points for these problems will be allocated as follows: each problem is worth 25 points.