HW 3: Functions and Testing

Due October 6th, 6pm, codePost.io
Please submit all files created for this assignment to codePost. You can submit as many times as you like up until the deadline. ***If you are planning to use one or more late days, please notify us by sending a private Piazza message to "Instructors".***

Familiarize yourself with how to approach a programming assignment and review the grading rubric on this assignment's page in Canvas before getting started.

Code style, documentation, and other requirements
The formatting guidelines introduced in Labs 1-3 are ***requirements*** for all assignments: every file must have a file comment, your programs (excluding test files) must start with `main()`, use constants instead of magic numbers and strings, and document all functions (excluding `main()` and test functions).

The focus of this homework is functions and testing. We are expecting to see your solutions split into multiple functions where each function has one job. If you have repeated code, move it into a function. We are also looking for thorough testing of all your functions. You should have a test file for each solution file and you should write tests to verify your code works under a range of conditions. For example, if a function requires arguments to be in a particular range, write unit tests to confirm your code handles input outside that range appropriately. One rule of thumb is that you should have at least one assert for every branch in your code. Another rule of thumb is that you should have a test function for every function in your program with the exception of `main()`. You cannot test functions that prompt for input from the user—this is one reason why you should only prompt for input in `main()`.

We will be checking that you are using GitHub regularly (at least 4 commits per homework) from now on. GitHub usage counts toward your professional skills score.

There is no written component for this assignment.

Programming Component (81 pts)

Problem 1: sizefinder.py and test_sizefinder.py

You are tasked with writing an interactive size finder for a T-shirt company. The program should prompt the user for their chest measurement in inches then return the user's size according to the company's three size charts: kids, women, and men.

Input prompt: `"Chest measurement in inches: "` You can assume the user will enter a number (rather than a string or boolean).

Next, your program should find the user's size on each of the three size charts. The size charts are as follows:

Kids		**Womens**		**Mens**	
Size	Chest (inches)	Size	Chest (inches)	Size	Chest (inches)
S	26 to < 28	S	30 to < 32	S	34 to < 37
M	28 to < 30	M	32 to < 34	M	37 to < 40
L	30 to < 32	L	34 to < 36	L	40 to < 43
XL	32 to < 34	XL	36 to < 38	XL	43 to < 47
XXL	34 to < 36	XXL	38 to < 40	XXL	47 to < 50
		XXXL	40 to < 42	XXXL	50 to < 53

Inform the user of the sizes that are available as follows (see examples 1 and 2):

```
Your size choices:
Kids size: <size>
Womens size: <size>
Mens size: <size>
```

...where **`<size>`** is the name of the matching size on the appropriate chart (S, M, L etc.) or "not available" if the chart does not have a matching size for the user. If there are no matching sizes for the user (i.e. all three charts are "not available"), print `"Sorry, we don't carry your size"` instead of the message above. See example 3 below.

Example input / output # 1 (user input is shown in blue)

```
Chest measurement in inches: 32.5
Your size choices:
Kids size: XL
Womens size: M
Mens size: not available
```

Example input / output # 2 (user input is shown in blue)

```
Chest measurement in inches: 36
Your size choices:
Kids size: not available
```

```
Womens size: XL
Mens size: S
```

Example input / output # 3 (user input is shown in blue)

```
Chest measurement in inches: 22
Sorry, we don't carry your size
```

Tip: It should be straightforward to write a program that produces the correct output and passes all codePost tests. However, for full credit, your program must make good use of functions and avoid repeating calculations or comparisons unnecessarily. For example, you should not need to write a separate conditional block for each size chart—consider how you can write one or more functions to handle the logic that is common to all size charts.

Don't forget to test your functions in test_sizefinder.py!

Problem 2: expenses.py and test_expenses.py

For this problem, you will write a module that contains a set of functions for calculating business trip driving expenses. The problem has been broken down into three parts: implementing functions, writing tests to verify your functions, and implementing the main function to allow a user to calculate their expenses.

Part 1 - Function implementation:

Implement the four functions described by the docstrings below. You should include the docstrings in your program. You may write additional "helper" functions as needed—just make sure the four functions exist and work as described.

```
'''
      Function -- calculate_mileage
          Calculates miles driven using the start and end odometer values.
      Parameters:
          start -- The odometer reading at the start of the trip. Expecting a
              number greater than 0.
          end -- The odometer reading at the end of the trip. Expecting a
              number greater than 0 and greater than the start value.
      Returns:
          The miles driven, a number. If either parameter is invalid (e.g.
          either parameter is negative or end is less than start), returns 0.
   '''
```

```
'''
```

```
    Function -- get_reimbursement_amount
        Calculates the amount in dollars that the employee should be
        reimbursed based on their mileage and the standard rate per mile.
        The standard rate for 2020 is 57.5 cents per mile.
    Parameters:
        mileage -- The number of miles driven.
    Returns:
        The amount the employee should be reimbursed in dollars, a float
        rounded to 2 decimal places.
  '''
```

```
'''
    Function -- get_actual_mileage_rate
        Calculates the actual trip cost per mile in dollars based on the
        car's MPG and the fuel price.
    Parameters:
        mpg -- The car's miles per gallon (MPG), an integer greater than 0.
        fuel_price -- The fuel cost in dollars per gallon, a non-negative
        float.
    Returns:
        The actual cost per mile in dollars, a float rounded to 4 decimal
        places. If supplied arguments are invalid, returns 0.0
  '''
```

```
'''
    Function -- get_actual_trip_cost
        Calculates the cost of a trip in dollars based on the miles driven,
        the MPG of the car, and the fuel price per gallon.
    Parameters:
        start -- The odometer reading at the start of the trip. Expecting a
            number greater than 0.
        end -- The odometer reading at the end of the trip. Expecting a
            number greater than 0 and greater than the start value.
        mpg -- The car's miles per gallon (MPG), an integer greater than 0.
        fuel_price -- The fuel price per gallon, a non-negative float.
    Returns:
        The cost of the drive in dollars, a float rounded to 2 decimal
        places. If any of the supplied arguments are invalid, returns 0.0
  '''
```

Note: Copying text from PDF documents can produce syntax errors due to the PDF character encoding. If this happens, try deleting and re-typing the copied quote marks.

Part 2 - Unit testing:

In test_expenses.py, write unit tests for the four required functions and any additional helper functions you wrote in part 1.

Part 3 - user interface:

In your `main()` function, print the following usage instructions (you will need multiple print statements or you can use the newline character, "\n"):

```
"MILEAGE REIMBURSEMENT CALCULATOR
Options:
1 - Calculate reimbursement amount from odometer readings
2 - Calculate reimbursement amount from miles traveled
3 - Calculate the actual cost of your trip"
```

Next, prompt the user to enter their choice:

```
"Enter your choice (1, 2, or 3): "
```

You can assume the user will enter an integer. If they enter an integer other than 1, 2, or 3, print `"Not a valid choice"`.

Depending on the user's choice, your program will prompt for different information and provide different output. You can assume the user will enter the appropriate data type at each prompt.

If the user chooses 1:
Ask the user to enter their start and end odometer readings, using the following prompts:

```
"Enter your starting odometer reading: "
"Enter your ending odometer reading: "
```

Use the functions you wrote for part 1 to get the amount they will be reimbursed for their trip. Print the result in the following format:

"You will be reimbursed $**<amount in dollars to 2 decimal places>**"

If the user chooses 2:
Ask the user to enter the number of miles they drove using the following prompt:

```
"Enter the number of miles traveled: "
```

Use the functions you wrote for part 1 to get the amount they will be reimbursed for their trip. Print the result in the following format:

"You will be reimbursed $**<amount in dollars to 2 decimal places>**"

If the user chooses 3:

Ask the user to enter their start and end odometer readings, their car's MPG, and the price they paid for fuel (dollars per gallon), using the following prompts:

```
"Enter your starting odometer reading: "
"Enter your ending odometer reading: "
"Enter your car's MPG: "
"Enter the fuel price per gallon: "
```

Use the functions you wrote for part 1 to calculate the actual trip cost in dollars:

```
"Your trip cost $<amount in dollars to 2 decimal places>"
```

Example input / output # 1 (user input is shown in blue)

```
MILEAGE REIMBURSEMENT CALCULATOR
Options:
1 - Calculate reimbursement amount from odometer readings
2 - Calculate reimbursement amount from miles traveled
3 - Calculate the actual cost of your trip
Enter your choice (1, 2, or 3): 1
Enter your starting odometer reading: 1000
Enter your ending odometer reading: 1010
You will be reimbursed $5.75
```

Example input / output # 2 (user input is shown in blue)

```
MILEAGE REIMBURSEMENT CALCULATOR
Options:
1 - Calculate reimbursement amount from odometer readings
2 - Calculate reimbursement amount from miles traveled
3 - Calculate the actual cost of your trip
Enter your choice (1, 2, or 3): 2
Enter the number of miles traveled: 10
You will be reimbursed $5.75
```

Example input / output # 3 (user input is shown in blue)

```
MILEAGE REIMBURSEMENT CALCULATOR
Options:
1 - Calculate reimbursement amount from odometer readings
2 - Calculate reimbursement amount from miles traveled
3 - Calculate the actual cost of your trip
Enter your choice (1, 2, or 3): 3
Enter your starting odometer reading: 1000
Enter your ending odometer reading: 1010
Enter your car's MPG: 36
Enter the fuel price per gallon: 3.09
Your trip cost $0.86
```