CS 1410: Caloric Balance

How many calories should you eat each day? It's a commonly-asked question, and the answer is that it's different for each person depending on their physical characteristics, activity, and whether they want to lose weight, gain weight, or maintain their current weight.

Calories are units of energy that our bodies consume and expend each day. If we consume the same number of calories that we expend, then our body weight does not change. If we consume fewer calories than we expend (a caloric deficit), then our body weight decreases, and if we consume more calories than we expend (a caloric surplus), then our body weight increases. We consume calories by eating food, and we expend calories by moving around and exercising (and even while resting or sleeping).

By leveraging a little bit of math, we can write a program that helps us to determine the number of calories that we should consume in order to lose weight, gain weight, or maintain our current weight, depending on our goals.

Calculations

You can calculate a person's daily caloric balance to determine if the person ends the day with a caloric deficit, a caloric surplus, or a stable balance. The calculation is simple: calories consumed minus calories expended. But how are each of these calculated?

- The number of calories a person consumes in a day is calculated by adding the number of calories of each food item that the person eats during the day. It's that simple.
- The number of calories a person expends in a day is calculated by adding two values: 1) the number of calories that the person's body expends while at rest (called the Basal Metabolic Rate, or BMR), and 2) the number of calories that the person expends while moving around or exercising during the day (which depends on the level of physical activity or how rigorous the exercise is).

  There are simple formulas to calculate both the BMR and the number of calories expended through physical activity:

  - **BMR**: there are two formulas to calculate this: one for men and one for women. In either case, the person's weight, height, and age are needed. Here are the formulas:

    ```
    Men:   BMR = 66 + (12.7 * height in inches) + (6.23 * weight in pounds) - (6.8 * age in years)
    Women: BMR = 655 + (4.7 * height in inches) + (4.35 * weight in pounds) - (4.7 * age in years)
    ```

  - **Physical activity**: each type of physical activity expends a different number of calories. Refer to this table for a list of physical activities and the number of calories that each activity expends per minute, per pound. So, in order to calculate the total number of calories expended for an activity, you'll need to know the person's weight and the duration of the activity in minutes.

Assignment

Your assignment is to create a Python program that uses a class to calculate and track a person's caloric balance throughout a day.

Your program should start by asking the user for a person's physical characteristics that are needed for the calculations. This includes the person's gender, weight, height, and age. Your program should use this information to calculate the person's BMR, subtract it from the caloric balance, and then print the updated caloric balance. The caloric balance should start at 0, so after subtracting the BMR, the caloric balance will be a negative number. Then, your program should display a menu that provides the user with three options:

1. **Record Food Consumption**: this option will ask the user for the number of calories of the food item that was consumed. Your program should add this number to the caloric balance and then print the updated caloric balance.
2. **Record Physical Activity**: this option will ask the user to select an activity from a list of choices (you must include at least four different activities for the user to choose from), as well as the number of minutes that the activity was performed. Your program should use this information to calculate the

number of calories expended, subtract this number from the caloric balance, and then print the updated caloric balance.

3. **Quit**: this option should terminate your program.

Sample

Program execution:

```
Hi! This program will calculate your caloric balance for the day!
Before we can start, I need some information about you. Be honest! :)

What is your gender (f or m)? f
What is your age? 23
What is your height in inches? 65
What is your weight in pounds? 130

Thanks! Now, throughout the day, tell me each time you eat or move.
Your caloric balance is starting at -1417.9 (you need to eat something)
What would you like to do?
[f] Record Food Consumption
[a] Record Physical Activity
[q] Quit
Enter an option: f
Okay! How many calories did you just eat? 400
Sweet! Your caloric balance is now -1017.9000000000001

What would you like to do?
[f] Record Food Consumption
[a] Record Physical Activity
[q] Quit
Enter an option: a
Choose an activity to record
[j] Jump rope
[r] Running
[s] Sitting
[w] Walking
Enter an option: j
For how many minutes did you perform this activity? 30
Awesome! Your caloric balance is now -1306.5

What would you like to do?
[f] Record Food Consumption
[a] Record Physical Activity
[q] Quit
Enter an option: e
Sorry, that option is invalid.

What would you like to do?
[f] Record Food Consumption
[a] Record Physical Activity
[q] Quit
Enter an option: a
Choose an activity to record
[j] Jump rope
[r] Running
[s] Sitting
[w] Walking
Enter an option: v
Sorry, that option is invalid.

What would you like to do?
[f] Record Food Consumption
[a] Record Physical Activity
[q] Quit
Enter an option: f
Okay! How many calories did you just eat? twelve
Please enter a number.
Okay! How many calories did you just eat? -17
Please enter a number greater than zero.
Okay! How many calories did you just eat? 0
Please enter a number greater than zero.
```

```
Okay! How many calories did you just eat? 600
Sweet! Your caloric balance is now -706.5

What would you like to do?
[f] Record Food Consumption
[a] Record Physical Activity
[q] Quit
Enter an option: a
Choose an activity to record
[j] Jump rope
[r] Running
[s] Sitting
[w] Walking
Enter an option: r
For how many minutes did you perform this activity? zero
Please enter a number.
For how many minutes did you perform this activity? -17
Please enter a number greater than zero.
For how many minutes did you perform this activity? 0
Please enter a number greater than zero.
For how many minutes did you perform this activity? 12
Awesome! Your caloric balance is now -842.22

What would you like to do?
[f] Record Food Consumption
[a] Record Physical Activity
[q] Quit
Enter an option: f
Okay! How many calories did you just eat? 935
Sweet! Your caloric balance is now 92.77999999999997

What would you like to do?
[f] Record Food Consumption
[a] Record Physical Activity
[q] Quit
Enter an option: q
Leaving? You should do this again tomorrow. Stay healthy!
```

Instructions

For this assignment you will need to make 2 files: `caloric_balance.py` and `main.py`. Unit tests are available for download. Your files need to be in the same folder as the unittest files.

You must follow the specifications exactly, but may choose your own method for solving the problem described for each. Once you have completed a function you should run the unittest for that function and have it pass all tests. Fix any errors, warnings, and/or failures.

Instructions for `caloric_balance.py`

Create a new class `CaloricBalance`

Data Members

`weight`

Keeps track of the user's entered weight. This is used for calculating caloric burn of activities.

`balance`

Keeps track of the user's caloric balance throughout the day.

Methods

`__init__`

The constructor takes 4 additional parameters `gender` a string (`'f'` or `'m'`), `age` a float, `height` a float, and `weight` a float. It should store `weight` as a datamember and initialize `balance` as the negative value of the

`getBMR` method. *You can't fully test this method until you complete the `getBMR` method. The unittests have been created to test appropriately.*

```
cb = CaloricBalance('f', 23.0, 65.0, 130.0)
```

`getBMR`

This method receives 4 additional parameters `gender` a string (`'f'` or `'m'`), `age` a float, `height` a float, and `weight` a float. It should calculate and return the BMR using the calculations above. If the gender is not equal to `'m'` or `'f'` then the method returns `0.0`

```
cb.getBMR('f', 23.0, 65.0, 130.0) -> 1417.9
cb.getBMR('m', 26.0, 70.5, 185.0) -> 1937.1 (rounded)
cb.getBMR('x', 30.0, 60.0, 145.3) -> 0.0
```

`getBalance`

This method receives no additional parameters and returns the value of the `balance` datamember.

```
cb = CaloricBalance('f', 23.0, 65.0, 130.0)
cb.getBalance() -> -1417.9
```

`recordActivity`

This method receives 2 additional parameters `caloric_burn_per_pound_per_minute` and `minutes`, both numbers (integer or float). It calculates the number of calories burned per minute by multiplying `caloric_burn_per_pound_per_minute` and the `weight` datamember; then calculates the total caloric burn by multiplying the previous calculation by the `minutes`. It should subtract the total caloric burn from the `balance` datamember.

```
cb = CaloricBalance('f', 23.0, 65.0, 130.0)
cb.getBalance() -> -1417.9
cb.recordActivity(0.074, 30)
cb.getBalance() -> -1706.5
cb.recordActivity(.009, 85)
cb.getBalance() -> -1805.95
```

`eatFood`

This method takes 1 additional parameter `calories` a number (integer or float). It adds the calories to the current `balance` datamemeber.

```
cb = CaloricBalance('f', 23.0, 65.0, 130.0)
cb.getBalance() -> -1417.9
cb.eatFood(400)
cb.getBalance() -> -1017.9 (rounded)
cb.eatFood(800)
cb.getBalance() -> -217.9 (rounded)
```

Instructions for `main.py`

Create the following functions in your `main.py` file. You will need to also import and use your CaloricBalance class from `caloric_balance.py`.

Because of the user input and output, not all functions are easily tested with unit tests. Be sure to test these functions by running your program and observing the correct behavior.

- `formatMenu`
- `formatMenuPrompt`
- `formatActivityMenu`
- `getUserString`
- `getUserFloat`
- `createCaloricBalance`
- `recordActivityAction`

- `eatFoodAction`
- `quitAction`
- `applyAction`
- `main`

`formatMenu`

The function `formatMenu` does not receive any parameters. It must return a list of strings that contains the lines of the menu.

```
formatMenu() -> ['What would you like to do?', '[f] Record Food Consumption', '[a] Record Physical
Activity', '[q] Quit']
```

`formatMenuPrompt`

The function `formatMenuPrompt` does not receive any parameters. It must return a string that contains the prompt to ask the user which menu option they would like to select.

```
formatMenuPrompt() -> 'Enter an option: '
```

`formatActivityMenu`

The function `formatActivityMenu` does not receive any parameters. It must return a list of strings that contains the lines of the activities menu. (Your activities can be different than the example below)

You should choose at least 4 activities you want to include. They do not have to be the same as what are listed below.

```
formatActivityMenu() -> ['Choose an activity to record', '[j] Jump rope', '[r] Running', '[s] Sitting',
'[w] Walking']
```

`getUserString`

The function `getUserString` receives one parameter, a string that contains a prompt for input. It must return a string that contains the text input by the user, with any leading and trailing whitespace removed. If the user gives an empty string, prompt them again, until they give a non-empty string. Note that this function interacts with the user, so there will be output to the screen and input from the keyboard when it is called.

```
getUserString("What is your name? ") -> 'It is Arthur, King of the Britons.'
getUserString("What is your quest? ") -> 'To seek the Holy Grail.'
getUserString("What is the air-speed velocity of an unladen swallow?") -> 'What do you mean? An African or
European swallow?'
```

`getUserFloat`

The function `getUserFloat` receives one parameter, a string that contains a prompt for input. It must return a float that contains the number input by the user. If the user enters a non-number or a number less than or equal to zero it should prompt them again. To accomplish this you might consider using a try/except clause which allows you to try an execute some code (convert user's input to a float) and recover if it fails.

Example

```
try:
    # try some code
    float('this is a string')
except:
    # this gets called if anything above fails.
    print("you can't convert that string to a float")
```

```
getUserFloat('Type 1.7 ') -> 1.7
getUserFloat('Type 1 ') -> 1.0
getUserFloat('Type in an integer ') -> 10.0
getUserFloat('Type in a float ') -> 3.142
```

`createCaloricBalance`

This function receives no parameters. It will prompt the user for their gender (f or m), their age (float or int), their height in inches (float or int), and their weight in pounds (float or int). It will create an instance of `CaloricBalance` and return that instance.

You will need to import your caloric_balance module.

```
createCaloricBalance() -> prompts user for input and returns an instance of CaloricBalance()
```

`recordActivityAction`

This function receives one parameter a `CaloricBalance` instance. It prints the activities menu and prompts the user to choose an activity.

If the user enters a valid option it then prompts the user to enter the number of minutes of activity and then calls the `recordActivity` method on the instance *(to do this you need to map the activity option to a value from the activities table linked above)*. Lastly it should print a success message to the user with their new caloric balance.

If the user enters an invalid option it should print an error message, then return.

```
recordActivity(caloricbalance) -> prints messages to the user, ask for input, and updates the caloric
balance.
```

`eatFoodAction`

This function receives one parameter a `CaloricBalance` instance. It prompts the user to enter the number of calories consumed. It then calls the `eatFood` method on the instance. Lastly it should print a success message to the user with their new caloric balance.

```
eatFoodAction(caloricbalance) -> asks the user for input, and updates the caloric balance.
```

`quitAction`

This function receives one parameter a `CaloricBalance` instance. This function will display a message to the user indicating the end of the program. It will then terminate the program using `sys.exit( 0 )`. Be sure to do the correct `import` statement. This function does not return anything.

```
quitAction(caloricbalance) -> the program will end
```

`applyAction`

This function receives two parameters a `CaloricBalance` instance, and choice a string. This function will call the appropriate action function based on the choice string. If the choice string does not match any accepted choices, it will display a message to the user. This function does not return anything.

```
applyAction(caloricbalance, "f") -> eatFoodAction gets called and caloric balance is updated.
applyAction(caloricbalance, "a") -> recordActivityAction get called and caloric balance is updated.
applyAction(caloricbalance, "q") -> the program will terminate.
applyAction(caloricbalance, "x") -> the user will receive an error message.
```

`main`

The function `main` receives no parameters, and returns nothing. This function ties everything together. Creates an instance of CaloricBalance, repeatedly asking the user their choice and taking appropriate action. *Note: everything you need to finish the main function should be contained in the functions above.*

```
main() -> the program runs.
```

Finishing Up

Lastly add this snippet at the bottom of your file which will execute your `main()` function when you run `main.py` but will allow it to be imported into the unittest files without executing the main function.

```
if __name__ == '__main__':
    main()
```

Pass-of instructions

1. To pass off this assignment you need to show your completed program to the lab assistants.
    - Show them your `caloric_balance.py` and `main.py` code
    - Run `test_all.py` - All tests MUST pass!
    - Run `main.py`
    - *The lab assistant may additional tests they want you to run*
2. Upload your `caloric_balance.py` and `main.py` files to canvas (you may zip them), please add a comment to the top of the files with your name and time your class meets.