Homework 6 —Timed Calculator

Due Tuesday, October 23rd at 11:55pm

(50 points)

Objectives:

- Increase familiarity with for loops and accumulator patterns
- Use lists as parameters and returns
- Use list functions
- Practice modular program development by designing your program's structure
- Practice using modules with random, time, and statistics

Turn In:

- `hw6_timedcalculator.ipynb`
- `hw6_graphs.pdf`

Turn these files in on moodle under the "Homework 6 - Timed Calculator" link. This link **will** restrict you to only being able to turn in your notebook and pdfs!

Instructions:

Your job is to write a program that prompts the user for a number of random numbers to generate and then calculates a number of different statistics for the numbers in that list.

The user can sort the numbers, count the numbers in the list, sum the numbers, find the mean, median, minimum or maximum value in the list.

Note that many of these statistics have corresponding built-in functions, such as `sum()`. **Read the directions carefully about which built-in functions you are allowed to use for each part of this assignment.**

Program Structure:

The structure of your program is largely up to you. You must use functions and return values, and each function should be in charge of one and only one task.

You may not do any of the actual calculation, nor should you generate your list of values in your `main()` function. You may call built-in functions and the functions that calculate your re-implemented results. Your functions that do calculation **must not also print results.**

1) Generating the data (5 points):

Your first step will be to prompt the user for how many values they want in their list. You must then generate a list with that many values, all randomly chosen. Use the random module's randrange function and a for loop to generate your values. The range that you generate from must include negative numbers and positive numbers, but is otherwise up to you.

2) Calling the built-in functions (5 points):

Your second step will be to prompt the user for which calculation they want to make, then run a timing experiment on the built-in version of that calculation. The timing experiment should use time.time() function.

3) Re-implementing the calculations (25 points for functions):

Your third step will be to prompt the user for which calculation they want to make, then run a timing experiment on the re-implemented version of that calculation.
All re-implemented calculations must use for loops and the accumulator pattern except for the last calculation, median (that is, they must not use any built-in function).

0) sorting the values (0 points): use the provided code from Lecture 13 (Monday, October 15th) of bubble sort for your sorting function.

1) counting the number of values (2 points): for this calculation, you are implementing an equivalent to the len() function. **You are not allowed to use the len() function itself.**

2) sum the values in the list (2 points): for this calculation, you are implementing an equivalent to the sum() function. **You are not allowed to use the sum() function itself.**

3) Find the minimum value in the list (10 points): for this calculation, you are implementing an equivalent to the min() function. **You are not allowed to use the min() function itself.**
Hint: you'll need to use both an accumulator pattern **and** and a conditional statement.

4) Find the maximum value in the list (1 point): for this calculation, you are implementing an equivalent to the max() function. **You are not allowed to use the max() function itself.**

5) Find the mean of the list (5 points): for this calculation, you are implementing an equivalent to the statistics.mean() function. **You are not allowed to use the built-in sum() or len() function, or the statistics.mean() function itself.**

6) Find the median of the list (5 points): for this calculation, you are implementing an equivalent to the statistics.median() function. A median is the value in the middle of a sorted list of values. If the list has an even number of elements, this function should return the average of the two middle values.
You are not allowed to use the statistics.median() function itself. You do not have to use a for loop for this solution.

4) Comparing results (3 points):

Your final step will be to compare the results from the built-in version of the calculation and your re-implemented version. If they match, print the message "Results from the built-in and

re-implemented versions match", then print the result **unless the user was sorting the values**. If they do not match print the message "Results do not match!", then print the value of the built-in version and the value of the re-implemented version. (see example output below)

Example outputs can be found at the end of this write-up.

5) Graph your results (5 points):

Create 4 graphs, plotting time (y-axis) against number of elements (x-axis) for all. The graphs should correspond to:

a) timing experiments for the built-in functions
    i) sorting the elements
    ii) counting the elements
b) timing experiments for the re-implemented functions
    i - ii) same as above

Your x-axis should have a wide enough range that you can see how number of elements effects time for each of these functions.

Your graphs must all include include a title and axis labels. Make sure to scale your axes so that the graphs make meaningful sense.

Development:
Your code should always be in a “runnable” state. **Do not attempt to develop your entire program without running it.**
You should complete and test each calculation task before moving on to the next one.

As a reference, our solution, including blank lines and comments, is 155 lines long.

It is your responsibility to test your program. You can use an online difference checker tool (like https://text-compare.com/) to compare and contrast the provided example outputs with the output of your program.

Style & Structure (7 points)
Naming (1 point)

- Your variable names should be meaningful and concise
- Your variable names should be formatted like: user_birthday (words start with lowercase letters and are separated with underscores)
- Function names should be formatted the same way as your variables: get_today()(words start with lowercase letters and are separated with underscores)

Proper Variable Usage (1 point)

- You should not have variables outside of functions.

Functions (5 points)

- You **must** have a `main` function. The only code in your program that is not inside a function should be the statement:

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

- Do not attempt to define a function inside another function.
- Each function must do one and only one task.
- You must use returns. (For example, if you have a function that finds the minimum value in the file, that function should not print the result, instead, you should return it.)
- Do not define two main functions.

Comments (3 points)

Your code must be commented. You must include a file comment (your name, your section, the homework name, and a brief description of what the program does), inline comments (explaining any complex code), and function comments. An example function comment is shown below:

```
# This function converts a temperature in fahrenheit to celsius
# and prints the equivalence.
# Parameters: fahrenheit - int or float degrees fahrenheit
# Return: float - equivalent degrees celsius
def fahrenheit_to_celsius(fahrenheit):
      celsius = (fahrenheit - 32) * (5/9)
      return celsius
```

Extra Credit (up to + 10 points)

1) Find the mode of the list (5 points): for this calculation, you are implementing an equivalent to the statistics.mode() function, **but you are not allowed to use the statistics.mode() function itself. You may use the list.count() function to help you.** If you implement this function, you must also add an extra menu item for the mode and run the built-in version to time against your mode function.

2) Graph more of your results (median, minimum, mode) (5 points):

Create 4 **more** graphs, plotting time (y-axis) against number of elements (x-axis) for all. The graphs should correspond to:

a) timing experiments for the built-in functions and
b) corresponding timing experiments for the re-implemented functions

The requirements for these graphs are the same as above.

To earn extra credit, these graphs must be for the median, minimum, and/or mode functions.

Example Outputs

Example Output for **incorrectly** implemented solutions

All results are for random numbers generated between -100 and 100 (inclusive); your results will vary depending on the range that you generate your random numbers in.

```
number of values? 10
Here are your statistics options:
0) Sort the values in the list
1) Length of the list
2) Sum the values in the list
3) Find the minimum of the values in the list
4) Find the maximum of the values in the list
5) Find the mean of the values in the list
6) Find the median of the values in the list
> 1

Built-in functions timing
Time elapsed (seconds): 3.0994415283203125e-06

Re-implemented functions timing
Time elapsed (seconds): 5.960464775390625e-06

Results do not match!
Built-in: 10
Re-implemented: 9
```

Example Output for correctly implemented solutions (user input in **underlined bold**)

```
number of values? 1234
Here are your statistics options:
0) Sort the values in the list
1) Length of the list
2) Sum the values in the list
3) Find the minimum of the values in the list
4) Find the maximum of the values in the list
5) Find the mean of the values in the list
6) Find the median of the values in the list
> 0

Built-in functions timing
Time elapsed (seconds): 0.0003199577331542969

Re-implemented functions timing
Time elapsed (seconds): 0.14859890937805176

Results from the built-in and re-implemented versions match
```

```
number of values? 11111
Here are your statistics options:
0) Sort the values in the list
1) Length of the list
2) Sum the values in the list
3) Find the minimum of the values in the list
4) Find the maximum of the values in the list
5) Find the mean of the values in the list
6) Find the median of the values in the list
> 2
```

```
Built-in functions timing
Time elapsed (seconds): 0.00025010108947753906

Re-implemented functions timing
Time elapsed (seconds): 0.002485990524291992

Results from the built-in and re-implemented versions match
Result: 1186
```

```
number of values? 50
Here are your statistics options:
0) Sort the values in the list
1) Length of the list
2) Sum the values in the list
3) Find the minimum of the values in the list
4) Find the maximum of the values in the list
5) Find the mean of the values in the list
6) Find the median of the values in the list
> 3

Built-in functions timing
Time elapsed (seconds): 6.9141387939453125e-06

Re-implemented functions timing
Time elapsed (seconds): 2.4080276489257812e-05

Results from the built-in and re-implemented versions match
Result: -93
```

```
number of values? 50
Here are your statistics options:
0) Sort the values in the list
1) Length of the list
2) Sum the values in the list
3) Find the minimum of the values in the list
4) Find the maximum of the values in the list
5) Find the mean of the values in the list
6) Find the median of the values in the list
> 4

Built-in functions timing
Time elapsed (seconds): 6.9141387939453125e-06

Re-implemented functions timing
Time elapsed (seconds): 2.5033950805664062e-05

Results from the built-in and re-implemented versions match
Result: 92
```

```
number of values? 170
Here are your statistics options:
0) Sort the values in the list
1) Length of the list
2) Sum the values in the list
3) Find the minimum of the values in the list
4) Find the maximum of the values in the list
5) Find the mean of the values in the list
6) Find the median of the values in the list
> 5
```

```
Built-in functions timing
Time elapsed (seconds): 0.00040602684020996094

Re-implemented functions timing
Time elapsed (seconds): 5.8650970458984375e-05

Results from the built-in and re-implemented versions match
Result: -0.4411764705882353
```

```
number of values? 897
Here are your statistics options:
0) Sort the values in the list
1) Length of the list
2) Sum the values in the list
3) Find the minimum of the values in the list
4) Find the maximum of the values in the list
5) Find the mean of the values in the list
6) Find the median of the values in the list
> 6

Built-in functions timing
Time elapsed (seconds): 0.0002300739288330078

Re-implemented functions timing
Time elapsed (seconds): 0.00021505355834960938

Results from the built-in and re-implemented versions match
Result: -3
```