Name: Evan Leglar	**Lab Time** T 12:00 PM

Names of people you worked with:
● Kristina Owen ● Andrea Oswalt

Websites you used:
● StackOverflow ● GeeksforGeeks ● PythonLearn.com

Approximately how many hours did it take you to complete this assignment (to nearest whole number)?	12

The Rules: Everything you do for this lab should be your own work. Don't look up the answers on the web, or copy them from any other source. You can look up general information about Python on the web, but no copying code you find there. Read the code, close the browser, then write your own code.

By writing or typing your name below you affirm that all of the work contained herein is your own, and was not copied or copied and altered.

Note: Failure to sign this page will result in a 50 percent penalty. Failure to list people you worked with may result in no grade for this lab. Failure to fill out hours approximation will result in a 10-percent penalty.

Turn .zip files of Python code to Canvas or your assignment will not be graded

BEFORE YOU BEGIN

1) Make a new project for this lab
2) Grab sensor.py and amp_filter.py

Learning Objectives:

You should be able to do the following:

- Use functions to decompose a problem in different ways
- Use lists to solve problems
- Create/use a generator function

Thoughts (come back and read these after you've read the problems):

1. You should write tests for all of the functions you write for this lab. Write the tests before you write the code, and put them after the `if __name__ == '__main__':` line. We're going to import the functions from your files for testing and, if you don't put your tests after this line (and indented to the if block), then it might confuse our testing code. Also, this if good programming practice.

2. Yes, we (more or less) wrote the sum functions in class last week.

3. There are a number of ways to write the recursive functions. Any of them will do.

4. There are likely to be built-in functions for a number of the things we're asking you to do for this lab. Please don't use them. If you do, you'll get zero points for that part of the assignment, since we want you to write them from scratch (this time).

5. The amplitude filter only affects sensor readings beyond a certain threshold. These sensor readings get clipped to the maximum allowed amplitude.

6. When you write the variable-width mean filter, you might find array slicing useful. Also, the filtered arrays will be shorter than the input arrays. For a filter with width 3, the filtered lists will be 2 elements shorter than the original list. For a width of 5, it will be 4 shorter, and so on.

7. If you catch the function caller using bad parameters, then you should raise an exception, such as a `ValueError`.

8. Make sure you write testing code for all of the functions that you write.

Grading Checkpoints

1. Correct iterative sum [1 point] and recursive sum [1 point] functions. Correct testing code for both functions [1point].

2. Correct iterative reverse function [1 point] and recursive reverse function [1 point]. Correct testing code for both functions [1 point]

3. Code that correctly uses the functions in amp_filter.py and sensor.py [1 point]. Correctly generate two files, with data and filtered data [1 point].

4. Mean filter function works and calculates the correct value [1 point] for width of 3. Mean filter function works and calculates the correct value of user-specified widths [2 points]. Mean filter function checks for reasonable errors in parameters and takes appropriate action [1 point].

For this assignment, everything will be turned into gradescope. There will be one submission for this pdf and one for the code. Do not turn in a zip file of the code. Instead, turn them all in at the same time.

Problem 1 sum

1. Write a function called `sum_i` that takes a list of numbers as its argument, and returns the sum of all the numbers in the list. You should calculate the sum using a `for` loop. Do not use the built-in `sum` function (although that's the right thing to do outside of the context of this lab). Put this function in a file called `sum.py`. We should be able to call this function with `s = sum_i(l)`.

2. Write another function in the `sum.py` file called `sum_r` that calculates the sum recursively. We should be able to call this function with `s = sum_r(l)`.

Comments for grader/additional information (if any)

sum.py

```
#!/usr/bin/env python

def sum_i(l):

    total1 = 0

    for i in range(0, len(l)):
        total1 += l[i]
    return total1


def sum_r(l):

    if len(l) == 1:
        return l[0]
    else:
        return l[0] + sum_r(l[1:])


if __name__ == '__main__':
    l = [1, 2, 3, 4]
    print("iterative sum of all the elements in the list is ",
sum_i(l))
    print("recursive sum of all the elements in the list is ",
sum_r(l))
    print("using built-in sum of all the elements in the list is ",
sum(l))
```

Console output

```
/Applications/miniconda3/bin/python "/Users/evanleglar/Desktop/ME499
Python/Lab 2/sum.py"
iterative sum of all the elements in the list is  10
recursive sum of all the elements in the list is  10
using built-in sum of all the elements in the list is  10

Process finished with exit code 0
```

Problem 2 reverse.py

1. Write two functions, one recursive and the other iterative, that reverse the order of a list. t. These functions should be called `reverse_r` and `reverse_i`, take a list an argument, return a new list, and be in a file called `reverse.py`. We should be able to call these functions with `l2 = reverse_r(l)` and `l2 = reverse_i(l)`. Input/Output Examples:
    1. `reverse_i('hello') == reverse_r('hello') -> ['o', 'l', 'l', 'e', 'h']`
    2. `reverse_i([1, 2, 3, 4, 5]) == reverse_r([1, 2, 3, 4, 5]) -> [5, 4, 3, 2, 1]`

Comments for grader/additional information (if any)

reverse.py

```
#!/usr/bin/env python


def reverse_i(l):
    if range(len(l)) == 0:
        return 0
    else:
        for i in l[:: -1]:
            if i == 0:
                return l[0]
            else:
                print(i)


def reverse_r(l):

    if not l:
        return []
    else:
        return [l[-1]] + reverse_r(l[:-1])


if __name__ == '__main__':
    # l = [1, 2, 3, 4]
    l = "hello"

    print(reverse_i(l))
    print(reverse_r(l))
```

Console Output

```
/Applications/miniconda3/bin/python "/Users/evanleglar/Desktop/ME499
Python/Lab 2/reverse.py"
o
l
l
e
h
None
['o', 'l', 'l', 'e', 'h']

Process finished with exit code 0
```

Problem 3 Create (and plot) example data

1. Grab a copy of sensor.py and amp_filter.py. Read the code in these files, and make sure you understand what they're doing. Write some code in **test_filter.py** to generate two files; one of 200 points of (simulated) sensor data (generated using the function in sensor.py) and another with the same data after you pass it through the amplitude filter (from amp_filter.py). Plot both of these files on the same graph, using your favorite plotting program. Notice that the data are noisy. This is meant to simulate the (zero-mean Gaussian) measurement noise of a (simulated) sensor. Note that you should not edit or copy the code in the files we give you; use `import` to bring them into your own code.
    1. Note: You are not plotting within Python (yet!) but instead outputting a .csv file and plotting in, eg, Excel

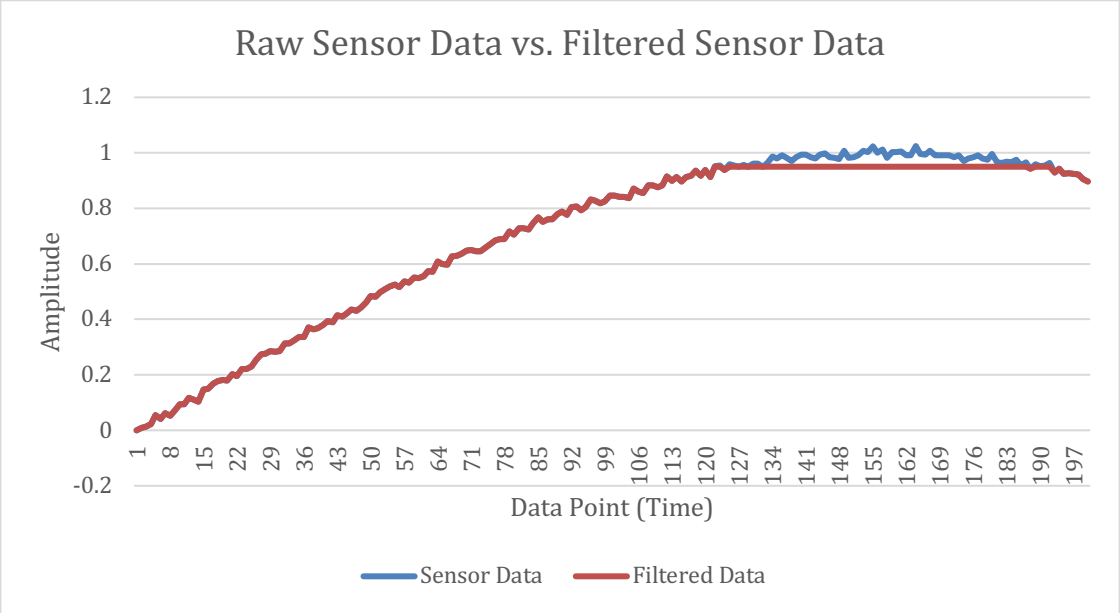
Comments for grader/additional information (if any)

test_filter.py

```
#!/usr/bin/env python3


import sensor as sen
import amp_filter as amp

X = sen.generate_sensor_data(200)
Y = amp.apply_amp_filter(X)

with open('me499_lab2_3_1.csv', 'w') as f_1:
    for i in X:
        f_1.write(str(i)+'\n')
    f_1.close()


with open('me499_lab2_3_2.csv', 'w') as f_2:
    for j in Y:
        f_2.write(str(j)+'\n')
    f_2.close()
```

```
if __name__ == '__main__':
    for k in Y:
        if k > 0.95:
            raise ValueError

    print('complete')
```

Plot image



Problem 4 Write a filter

1. Write a new filter, called `mean_filter` in `filters.py` that replaces each sensor reading with the mean of itself and the readings on either side of itself. For example, if you have these measurements in the middle of the list

   ```
   10 13 13
   ```

   then the middle reading would be replaced by 12 (since (10 + 13 + 13) / 3 = 12). Write code to generate two files, and plot the data in these files, showing the effect of applying your mean filter. We should be able to call your filter using `l2 = mean_filter(l)`.

2. Modify your code to have a variable filter width. Instead of a width of 3, add a parameter to the filter function that specifies the filter width. This parameter must be positive and odd. Test this with a variety of widths, and see what it does to the data. We should be able to call your code with `l2 = mean_filter(l, 5)`. We should also be able to call the code with `l2 = mean_filter(l)`, in which case the width should default to 3.
   Input/Output Examples:
   1. `mean_filter([1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1]) -> [2, 7/3, 2, 5/3, 2, 7/3, 2, 5/3, 2, 7/3, 2]`
   2. `mean_filter([1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1], 5) -> [9/5, 2, 11/5, 2, 9/5, 2, 11/5, 2, 9/5]`

Comments for grader/additional information (if any)

filters.py

```
#!/usr/bin/env python3

import sensor as sen


def mean_filter(data, n=3):

    m_avg = []

    if (n % 2) == 0:
        raise ValueError
    else:
        for i in range(len(data)):
            avg = sum(data[i:i+n])/n
            m_avg.append(avg)
        return m_avg
```

```
if __name__ == '__main__':
    data = sen.generate_sensor_data(200)
    mov_avg = mean_filter(data, 5)
    print(mov_avg)

    with open('me499_lab2_4_1.csv', 'w') as f_1:
        for i in range(len(data)):
            f_1.write(str(data[i])+'\n')

    with open('me499_lab2_4_2.csv', 'w') as f_ma:

        for i in range(len(mov_avg)):
            f_ma.write(str(mov_avg[i])+'\n')
```

Console Output

Copy and paste the results of your checks here (at a minimum, input/output examples from above

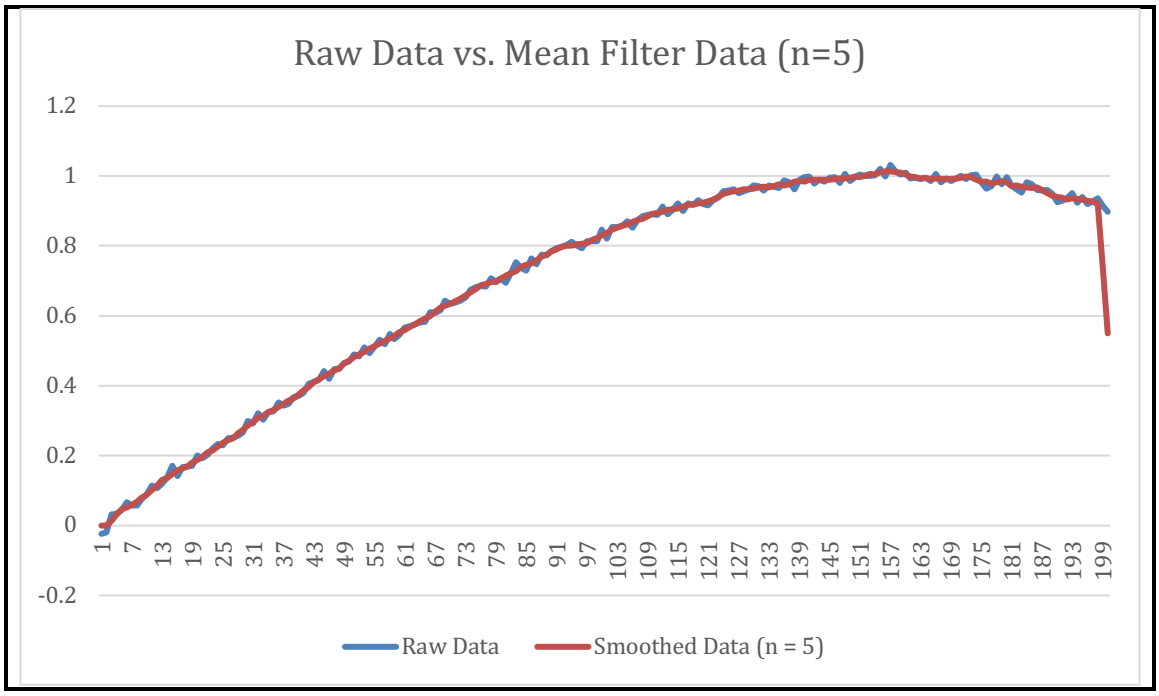
mov_avg = mean_filter(data)

```
/Applications/miniconda3/bin/python "/Users/evanleglar/Desktop/ME499 Python/Lab 2/filters.py"
[2.0, 2.3333333333333335, 2.0, 1.6666666666666667, 2.0, 2.3333333333333335, 2.0, 1.6666666666666667, 2.0, 2.3333333333333335, 2.0, 1.0, 0.3333333333333333]

Process finished with exit code 0
```

mov_avg = mean_filter(data, 5)

```
/Applications/miniconda3/bin/python "/Users/evanleglar/Desktop/ME499 Python/Lab 2/filters.py"
[1.8, 2.0, 2.2, 2.0, 1.8, 2.0, 2.2, 2.0, 1.8, 1.6, 1.2, 0.6, 0.2]

Process finished with exit code 0
```

Plot


Raw Data vs. Mean Filter Data (n=5)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
1
7
13
19
25
31
37
43
49
55
61
67
73
79
85
91
97
103
109
115
121
127
133
139
145
151
157
163
169
175
181
187
193
199
Raw Data
Smoothed Data (n = 5)