Purpose

The goal of this lab is to give you some more experience writing functions and trying to decompose problems in different ways.

Preparation

For this lab, do not use the `me499.py` script to create the lab2 directory. In other words, don't do `me499.py create lab2`; it won't work (but may work in future labs). Instead, just make a new folder "lab2" in your me499/me599 directory and create the files for lab yourself.

Assignment

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. Put this function in a file called `sum.py`

2. Write another function in the `sum.py` file called `sum_r` that calculates the sum recursively.

3. Write two functions, one recursive and the other iterative, that reverse the order of a list. The functions should be called `reverse_r` and `reverse_i`, take a list as an argument and return a list, and be in a file called `reverse.py`.

4. Do exercise 8 from chapter 6 in the book. Call the function `gcd` and put it in `gcd.py`.

5. Do exercise 5 from chapter 7 in the book. Put the function in `estimate_pi.py`.

6. Get a copy of sensor.py and null_filter.py. Read them, and make sure you understand what they're doing. Generate a file of (simulated) sensor data by running null_filter.py, and plot it with `gnuplot`. Notice that it's noisy. This is meant to simulate the (zero-mean gaussian) measurement noise of the sensor. Note that you don't need to edit either file.

7. Write a new 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 replace by 12 (since (10 + 13 + 13) / 3 = 12). Generate two files, one for the data before filtering, and one for the filtered data. Plot both of these on the same graph in `Gnuplot`.

8.

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.

9. Write a variable-width median filter. This is like the mean filter that you just wrote, except that it replaces the values with the *median*of the values, rather than the mean. Write this filter in `filters.py`.

Grading

One point for each of the six functions in the first five questions. Two points for the mean function. Two points for the variable-width mean function. One point for the median function.

Submission

Like in previous labs, you will submit a tarball of your code. Use this command to build your tarball:

```
tar -czf username.tgz sum.py reverse.py gcd.py estimate_pi.py filters.py
```

Where you replace `username` with your username. If you want to find out more about the `tar` command, then `man tar` will help you out.

Thoughts

1. You should write tests for all of the functions you write for this lab.
2. Yes, we wrote the sum functions in class (more or less) last week.
3. There are number of ways to write the recursive function. Any of them will do.
4. `gnuplot` is a graph-plotting program. Google can tell you more, but all you need to know for this lab is that you start it up in a terminal with the command `gnuplot` and plot lines with the following command:

   ```
   plot "raw" with lines, "filtered" with lines
   ```

   assuming you have two files, called `raw` and `filtered`.
5. The null filter doesn't actually do anything to the sensor readings. It's really just to show you how you might write a filter for a list of sensor values. Don't worry if you don't understand the stuff in `sensor.py`; we'll cover that in class soon. All you really need to know is that it generates a list of sensor measurements for a fictional sensor.
6. You might find this code useful, to associate list positions with values for the mean and median code:

```
for i,datum in enumerate(data)
    print i, datum
```

7.

   To see the noise in the simulated sensor signal, either make the `Gnuplot` window really big, or pass in a larger `noise` parameter to the function. Google and the Python documentation will help you answer the question associated with this part of the exercise.

8. The data from the mean filter will have two fewer measurements than the original data set, because the mean of the first and last elements is not well-defined. Don't worry about this.

9. `numpy` contains a useful set of functions. Google knows about it.

10. We're not initially giving you a grading script for this lab, to encourage you to write you own tests. When you're writing tests, you should make sure you have a source of the correct input/output pairs (either a built-in function, or something you've verified by hand), and should test your code against this known answer. For the sum functions, this is easy, since there's a built-in function that does the same thing. For calculating pi, you could verify that you're close to it after the function is done.