First, a few tips:

1. When you sit down to program, have documentation nearby. No programmer remembers the details of all the different syntactic idiosyncrasies of a particular language. Instead, we have a source of information nearby to search for how to use the certain tool.
   a. This is one important source to figure out most things in Python: \texttt{http://docs.python.org/}
   b. This is a good source for basic things like, syntax, if else, while loop, for loop, strings, and lists, functions, file input/output: \texttt{http://www.tutorialspoint.com/python}
   c. There are many other good sources on the internet. Ask your peers what references they use to help them program. Just asking them should make you realize that no-one programs with a blank page in front of them.
      a. (It’s definitely Okay to not know these things by heart. I don’t know them in the programming languages that I’ve programmed for many years. You simply need to figure out good ways to find how a certain tool in that language is used.

2. Test things out in the interpreter. If you are going to use a function inside your program, but you are not sure exactly what it should receive or what it returns, then tested in the interpreter.
   a. For example, if you are trying to create a for loop, from 1 to X, and you know \texttt{range(1,X)} creates a list numbers. Type in \texttt{range(1,10)} in the interpreter and see the list of numbers that it generates. Is that what you need?
   b. Also, don’t forget to do \#1, that is, try to find the documentation of that function first - read it, and follow the examples, as you test them on your own: \texttt{http://docs.python.org/library/functions.html?highlight=range#range}

3. After you’ve tried \#1 and \#2, if you are still stuck, seek help. Most of your classmates will be more than happy to help. And I’m available most Mondays, Wednesdays, Friday afternoons.

You can send me your answers to each of these problems and I’ll give you feedback. You can ask me questions about any of these problems and I can give you specific tips for how to solve them. Upon request, I can send you answers to any of this questions.
Exercises

**Linear programming.** The following exercises involve the ability to define a function. And to know the difference between a function that simply prints out a result, and one that returns a value. Become familiarized with the syntax (e.g., [http://www.tutorialspoint.com/python/python_functions.htm](http://www.tutorialspoint.com/python/python_functions.htm)).

a. Write a function that sums 3+5 and prints out the answer. Call this function test(). Whenever you call the function test(), the program should print out the answer: 8.

b. Write a function that calculates \( y = 13x + 4 \) for any \( x \) given by the user. The function should return \( y \). Call this function simpleline(x). When you call it with \( y = \text{simpleline}(1) \), \( y \) should contain 17.

**Branching.** The following programs involve using the “if .. then” tool. Become familiarized with the syntax of the if .. elif .. else tool (e.g., [http://www.tutorialspoint.com/python/python_if_else.htm](http://www.tutorialspoint.com/python/python_if_else.htm)).

c. Write a function that takes in two numbers, and returns the smallest of the two. Call this function min2(x,y). When you call it with \( z = \text{min2}(2,3) \), \( z \) should contain 2.

d. Write a function that takes in three numbers, and prints out smallest. Call this function min3(x,y,z).

e. Write a function that takes in three numbers, and prints out the relationship between the first one and the other two. Call this function relation(x,y,z). If x is the smallest number, print a message that says “x is smallest”. If x is the largest, print a message that says “x is the largest.” If x is less than one, but greater than the other, print a message that says “x is in the middle.”

f. Write a function that takes one number and prints out whether that number is negative or positive, and whether the number is small or large. For the purpose of this function, we’ll consider a number small if its absolute value is less than 100. Use the built-in function abs(x), that returns the absolute value of x ([http://docs.python.org/library/functions.html#abs](http://docs.python.org/library/functions.html#abs)).

**Iteration.** The following programs involve using the *while* loop and the *for* loop. For each exercise write a version of the program using both techniques. First, become familiarized with the syntax of these two tools: *while* (e.g., [http://www.tutorialspoint.com/python/python_while_loop.htm](http://www.tutorialspoint.com/python/python_while_loop.htm)) and *for* (e.g., [http://www.tutorialspoint.com/python/python_for_loop.htm](http://www.tutorialspoint.com/python/python_for_loop.htm)).

g. Write a function that prints out the numbers from 1 to 10. Call the function counter().

h. Write a function that prints out the numbers from 10 to 1, and then prints out “Blast off.” Call the function countdown().

i. Change the function above so that it receives one input X, and prints out the numbers from X to 1, and then prints out “Lift off.” Call the function customcountdown(X).
Iteration with branches. The following programs combine iterative structures together with branching programs.

j. Another variation on the last three exercises. Make a function that receives two inputs: From and To. The function should print out the numbers from From to To. Call this function counter(From,To). For example, if the program is called with counter(1,4) then the output should be 1, 2, 3, 4. If the program is called with counter(4,1) then the output should be 4, 3, 2, 1.

k. Write a function that prints out the even numbers from 1 to 10. Call the function even().

l. Change the previous function so that it receives input X and prints out the even numbers between 1 to X. Call the function even(X).

Data structures. The following programs require the use of lists. Knowing how to handle lists is absolutely crucial to programming. Take a few minutes to become well familiarized with the structure of lists in python (http://docs.python.org/tutorial/datastructures.html).

m. Starting from the last exercise, modify that function such that instead of printing all of the even numbers, you add them to a list. Make the function return the list with all the even numbers between 1 and X. Call the function y = even(x). When you call “y = even(5)”, nothing should print on the screen. But when you type “print y”, the list [2, 4] should be printed.

n. One of the items of a list can be another list. Make a function that generates the numbers from 1 to X and returns a list whose first component is a list with all of the odd numbers and whose second item is a list with all of the even numbers. Call it oddeven(x). When you call y = oddeven(5), nothing should print on the screen. But when you type print y, you should see the composite list [[1,3,5],[2,4]].