Programming Exercises:
Coding Exercises 1-5 and 7 are in file ch1All.py
Coding Exercise 6 is in problem6.py

```
>>> print("hello, world!")
hello, world!
>>> print("hello", "world")
hello world
>>> print(3)
3
>>> print(3.0)
3.0
>>> print(2+3)
5
>>> print(2.0+3.0)
5.0
>>> print("2" + "3")
23
>>> print("2 + 3 =", 2+3)
2 + 3 = 5
>>> print(2*3)
6
>>> print(2**3)
8
>>> print(2/3)
0.6666666666666666
>>> print (2.0/3.0)
0.6666666666666666
>>>
```

1. **Instructions: Start up the python console and type in the following commands (print statements that are seen on the right).**There were no inputs in this problem. The outputs were certain strings or literals that needed to be printed (see image). There wasn't much of a process to solving this problem; it was just a series of print() statements.
2. **Instructions: enter and run the chaos program from section 1.6. Try it out with various inputs.** There was one input between 0 and 1. This is the number that gets the action "x = 3.8 *x *(1-x)" performed on it. There were ten outputs for this problem, one after each iteration of the loop. It gives the current value of "x" after however many times being put through the formula "x = 3.8 *x *(1-x)" (see the image on the right). I didn't go through a process to solve this problem because the code was given in the textbook.

```
>>> main()
This is a chaotic function
Enter a number between 0 and 1.002
0.0077843999999999995
0.030122832154895994
0.11394024383766943
0.39373567222033923
0.9309607813019936
0.2506639394203436
0.7325429626890099
0.7641027049648893
0.7029740687978917
0.8143259568429521
>>>
```

3. **Change the chaos function so that the multiplier is 2.0 instead of 3.0**. See documentation for problem 2. Every part of the code is the exact same except for the fact that the formula is "x = 2.0 *x *(1-x)" instead of "x = 3.8 *x *(1-x)". You can see the difference in the image of the output below.

With 3.9 as multiplier -- With 2 as multiplier

```
>>> main()
This is a chaotic function
Enter a number between 0 and 1.01
0.03861
0.14476514481
0.482851970866745
0.9738531858776958
0.09930631711087586
0.348833832721725
0.8858802804945467
0.3942759955892548
0.9314074960762914
0.24916153208382014
>>>
```

```
>>> main()
This is a chaotic function
Enter a number between 0 and 1.01
0.0198
0.038815920000000004
0.0746184887091072
0.1381011397037520 7
0.2380584298325536 5
0.3627732276364255 5
0.4623376258933515
0.4971630911533017
0.4999839036896391
0.4999999994818309
```

4. **Instructions: Modify the chaos program such that it prints out 20 values instead of 10.** After looking at the problem, I realize that to print 20 values, I have to run the program 20 times. That means the input is still a number between 0 and 1. The process is still running it through the formula "x = 3.9 * x* (1-x)." The only thing that changes is the output. There are 20 outputs instead of 10. To do this, I changed the code from **in range(10)** to **in range(20)**.
5. **Instructions: Modify the program so the user can specify the number of outputs**. Now, there are multiple inputs to the program. The first input is the number between 0 and 1. The

second number is the number of iterations. To solve this problem, I thought about the construction of a for loop, the type of loop I used. I know that the in range(<expr>) is where I can specify the number of iterations. I know I have the variable as an input, so I can just pass it into the range() part of the for loop to make the number of iterations the number specified by the user. After that, the output process is similar, except the outputs is the number the user specifies. The program still runs the input between 0 and 1 through "x = 3.9 *x* (1-x)."

6. This problem had me change the formula from **x = 3.9 *x*(1-x)** to other formulas. The input was the same. It was a number between 0 and 1. My idea was that I could print all the values in a table so that I could see the change in values between the different formulas. I needed to make three variables that were all the same value because each of the values was going to be put in a different formula. Inside the for loop, the formulas made changes to their respective variable. At the end of the loop, I printed a value. Below are the important values being compared.

a. 3.9 * x * (1-x): **First value:** 0.03861 **100th Value:** 0.3406455860976413
b. 3.9 * (x-x*x): **First value:** 0.038610000000000005 **100th Value:** 0.34361776630094903
c. 3.9 * x -3.9*x*x: **First value:** 0.03861 **100th Value:** 0.15944528810647585

7. For this problem, I needed to take in two inputs, run them through the same formula, and put them in a table. I needed to take in 2 numbers. The process I came up with was to run the values through the for loop, and then print out a concatenated string which contains both of the variables separated by hyphens. This method ended up working. The output was two variables, separated by hyphens.