Calling a Function from a Loop

Once a program gets to about 50 lines long, we probably want to start modularizing it into different functions. And it turns out that some small computations that we code are actually things we want done several times.

In this simple exercise, you will write some functions that perform a computation, and then in the main program you will call these functions from inside loops. Doing this is helpful if we want to create a table of values, or if we want to perform some kind of analysis on the function.

Open a new source file.

1. Write a function called `fun1` that takes one real-number parameter (i.e. "double" data type if you are in Java). Let's say the parameter is called x. The function should return $3x^2 - 2x + 1$. If you are working in Java, your function will be a static method, because we are not creating objects.

2. Same as #1, but this time let the function be called `fun2`. It should return the value x^{-x}. Do you remember how to perform exponentiation? In Python, use the ** operator. In Java, use the built-in `pow` method in the Math package.

3. Now, we are ready to make use of these functions! In your main program, let's evaluate fun1 for all integer values from 1 to 10 inclusive. In pseudocode, we want to do this:

```
for i = 1 to 10
   print both i and fun1(i)
```

4. The second function fun2 is studied in calculus. For example, we may want to know where this function reaches its maximum value. It is somewhere between x = 0 and x = 1. Let's create a data table for fun2 that evaluates for each hundredth from 0.01 to 1.00, in other words, we should call fun2 on 0.01, 0.02, 0.03, etc. all the way up to 1.00.

```
for i = 0.01 to 1.00, incrementing by .01
   print both i and fun2(i)
```

5. Modify the code for the 2nd loop that you just wrote. We want to keep track of the exact location of the maximum value of fun2. It would go something like this:

```
max_i = 0.01
max_value = fun2(0.01)
for i = 0.01 to 1.00, incrementing by .01
   print both i and fun2(i)
   if (fun2(i) > max_value)
      update max_i and max_value
```

At the end of the program, tell the user at what value of i fun2 achieved its maximum, and what that maximum value was.