Homework 1

Introductory Programming **Allen B. Downey**
Fall 2004

The reading for this assignment is Chapters 1 and 2 of *How to think...*. The goal of this assignment is to explore the Python interpreter, to make some syntax and run-time errors, and to write and execute a Python script.

1.1 The Interpreter

1. To start Python, type `python` in a Unix shell. You should get a startup message and a chevron (`>>>`):

   ```
   Python 2.3.3 (#1, May  7 2004, 10:31:40)
   [GCC 3.3.3 20040412 (Red Hat Linux 3.3.3-7)] on linux2
   Type "help", "copyright", "credits" or "license" for more information.
   >>>
   ```

   The chevron is a prompt; it means that the interpreter is waiting for you to type an expression or a statement.

2. Type `1 + 2` and then hit return. Python evaluates this **expression**, prints the result, and then prints another prompt.

3. Type `1 2` and then hit return. Python tries to evaluate the expression, but it can't because the expression is not syntactically legal. Instead, it prints the error message:

   ```
     File "<stdin>", line 1
       1 2
         ^
   SyntaxError: invalid syntax
   ```

   In many cases, Python indicates where the syntax error occurred, but it is not always right, and it doesn't give you much information about what is wrong. So, for the most part, the burden is on you to learn the syntax rules.

   In this case, Python is complaining because there is no operator between the numbers. If you want to perform multiplication, you have to use the multiplication operator, which is `*`.

4. Type `print 'hello'`. Python executes this **statement**, which has the effect of printing the letters `hello`. Notice that the quotation marks that you used to enclose the string are not part of the output.

5. Type `print bob` without the quotation marks. The output will look something like this:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'bob' is not defined
```

This is a run-time error; specifically, it is a NameError, and even more specifically, it is an error because the name `bob` is not defined. If you don't know what that means yet, you will soon.

1.2 Running a script

When you are only evaluating a few expressions, or executing a few statements, it is convenient to use the Python interpreter in interactive mode. When you start writing bigger programs, you will want to store your programs in a file called a **script** and then use the interpreter to execute the script.

1. An easy way to create and edit files is with `emacs`. When I am working on a program, I usually have an `emacs` window to edit the program and a terminal window to run it.

   Make sure that you are in the directory where you want to put a new file and type `emacs test.py &`. Because the file name ends with `py`, `emacs` knows that it is a Python program. The ampersand is there so you can use the emacs window and the terminal at the same time.

2. Write a line of Python code in the emacs window, maybe a print statement like `print 'hello'`

3. Save the file using the File menu, the disc icon or the keyboard shortcut Control-x Control-s.

4. In the terminal window, type `python test.py`. You are starting the interpreter and telling it to execute the script `test.py`. It should print `hello`.

5. In the emacs window, change `hello` to `jello`. Now go back to the terminal window and run the script again. Did it say `hello` or `jello`? If you saved the script before you ran it again, you will see the new version; otherwise you will see the old version.

   This is, frankly, an annoying little GOTCHA! You have to remember to save your script before you execute it.

6. Add a second line to your script, something like `print 3+1`. Save the file and then go back to the terminal. Press the Up Arrow key. You should see the previous command again, and you can hit return to execute it again. That should save some typing.

7. We're almost done, but there is one more GOTCHA! I have to warn you about. Add another line to your script, but instead of a statement (like a `print` statement), just write an expression (like `5*7`). Save and run the script. Did it print `35`?

   Probably not. The reason is that when you evaluate an expression in a script, Python doesn't display the result. It you *want* to display the result, you have to use a `print` statement.

8. At this point, you can create and execute Python scripts, and that's pretty much all you need to know for the rest of the semester. Of course, there are other things you might `want` to know...

1.3 Coming attractions

1. Make sure you are in a directory where you want to put Python code and then use `wget` to download `World.py` from the class web page:

   ```
   wget http://wb/ip/code/World.py
   ```

   Now press the Up Arrow key and edit the previous command so that it reads

   ```
   wget http://wb/ip/code/AmoebaWorld.py
   ```

   Press return to download `AmoebaWorld.py`.

2. Run the program you just downloaded by typing

   ```
   python AmoebaWorld.py
   ```

   You should see the top view of a microscope slide with an amoeba in the middle.

3. Press the `Run` button. The amoeba should move toward the northeast corner of the slide, leaving a trail of slime.

4. Change the entry marked `x(t)` so that it reads `cos(t)` and change `y(t)` to `sin(t)`. Press run again.

In the next homework we will do something more serious with `AmoebaWorld` but for now I thought you would like to see it.