Exercises

1. Write an English sentence with understandable semantics but incorrect syntax. Write another English sentence which has correct syntax but has semantic errors.
2. Start a Python in Idle. Type `1 + 2` and then hit return.
   Python *evaluates* this *expression*, prints the result, and then prints another prompt. `*` is the *multiplication operator*, and `**` is the *exponentiation operator*. Experiment by entering different expressions and recording what is printed by the Python interpreter. What happens if you use the `/` operator? Are the results what you expect?
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.

   See if you can find a few more examples of things that will produce error messages when you enter them at the Python prompt. Write down what you enter at the prompt and the last line of the error message that Python reports back to you.

4. Type *print('hello')*. Python executes this statement, which has the effect of printing the letters h-e-l-l-o to the screen. Notice that the quotation marks that you used to enclose the string are not part of the output. Now type `"hello"` without the print function and describe your result. Make note of when you see the quotation marks and when you don't.
5. Type `print(cheese)` without the quotation marks. The output will look something like this:

   ```
   Traceback (most recent call last):   File "<stdin>", line 1, in ?
   NameError: name 'cheese' 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 *cheese* is not defined. If you don't know what that means yet, you will soon.

6. Type `'This is a test...'` at the Python prompt and hit enter. Record what happens.

   Now create a python script named `test1.py` with the following contents (be sure to save it before you try to run it):

   ```
   'This is a test...'
   ```

   What happens when you run this script? Now change the contents to:

   ```
   print('This is a test...')
   ```

   and run it again.

   What happened this time?

   Whenever an *expression* is typed at the Python prompt, it is *evaluated* and the result is printed on the line below. `'This is a test...'` is an expression, which evaluates to `'This is a test...'` (just like the expression `42` evaluates to `42`). In a script, however, evaluations of expressions are not sent to the program output, so it is necessary to explicitly print them.