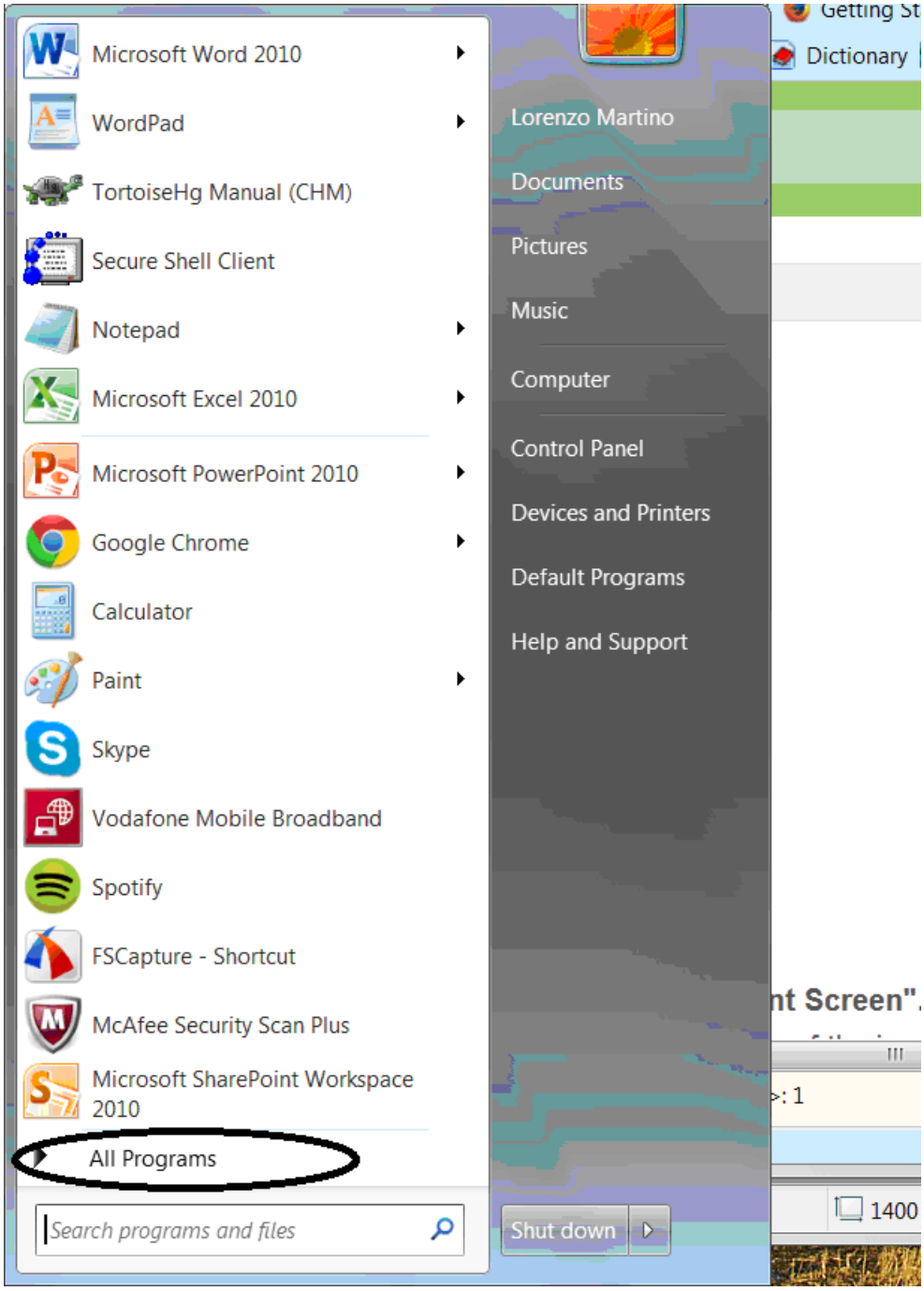
1. Familiarizing with IDLE

IDLE is the standard Python development environment. Its name is an acronym for “Integrated Development Language Environment. If you want to “tell” Python to do something, you can do that using IDLE.

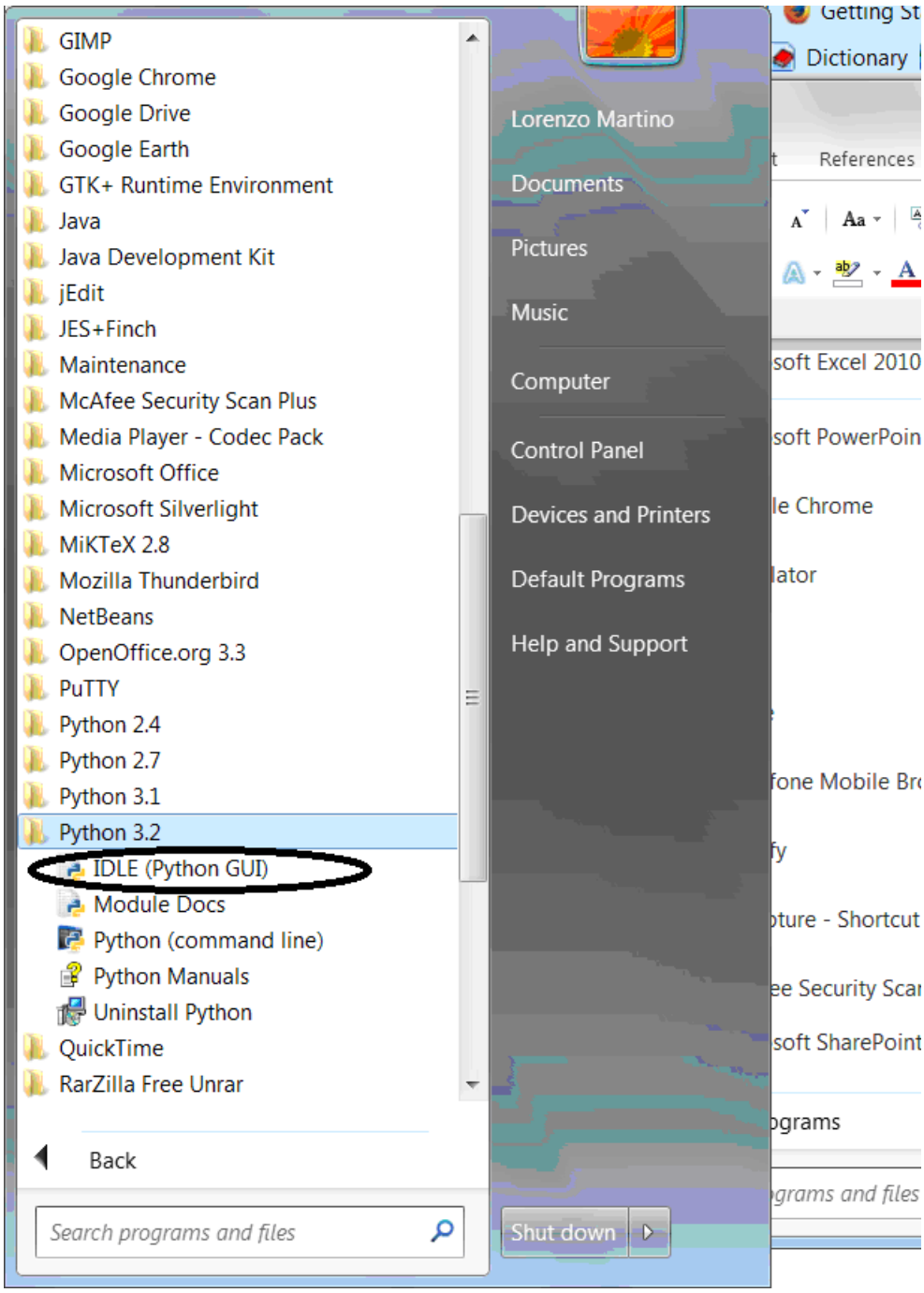
Where can you find the Python IDLE in your CS Windows workstation?

Just click:

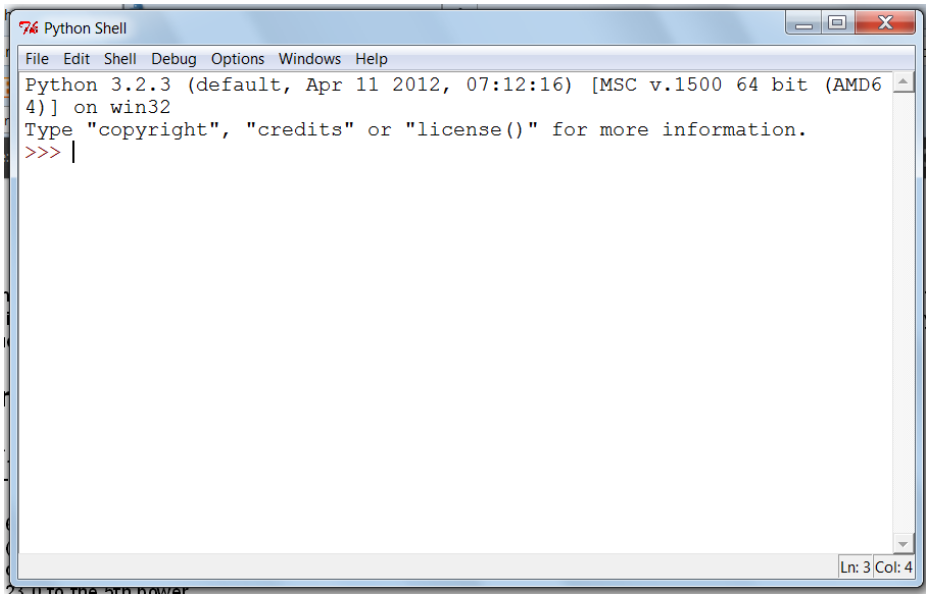
Then select All programs (see figure below):



Then select Python 3.2 (see figure below), and finally, click IDLE (Python GUI).



The following window will appear:

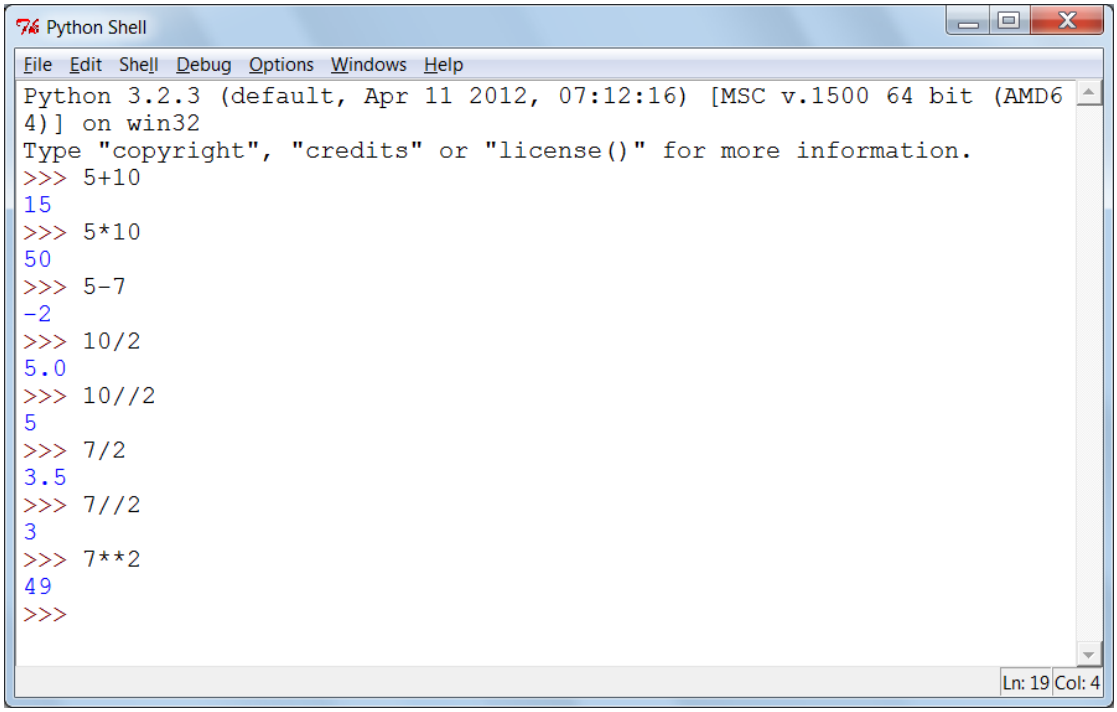


This is the **IDLE Python Shell window**. The **>>>** symbol is the **Python Shell prompt**; it indicates that the Python Shell is ready to get Python commands.

In the IDLE Python Shell windows you can type the commands (statements) that Python understands.

Let's try some simple commands.

2. Doing simple arithmetic operations in the Python Shell window.



We just told Python to sum 5+10, to multiply 5 by 10 (5*10), to subtract 7 from 5 (5-7).

Then we asked Python to divide 10 by 2 (10/2). But as you can notice we got as result 5.0. Also notice the following command (10//5). As you can see the result was 5. So what is the difference?

Well, the Python "/" operator indicates (we say "denotes") the division while the "//" operator denotes the integer division. The first operator produces a decimal number (a number with decimal digits) while the second one produces only an integer, discarding any fractional result.

Try by yourself:

7/2 and

7//2

Lastly, the ** symbol used in the last command (7**2) denotes the exponentiation operator. The command (statement) 7**2 tell Python to raise the number 7 to the second power.

Note also that the numbers we used in the operations above (10, 5, 7, 2, and so forth) are all **numeric constants (more specifically integer constants)**. Of course a numeric constant can have fractional part, such as 10.322, -8.76548.

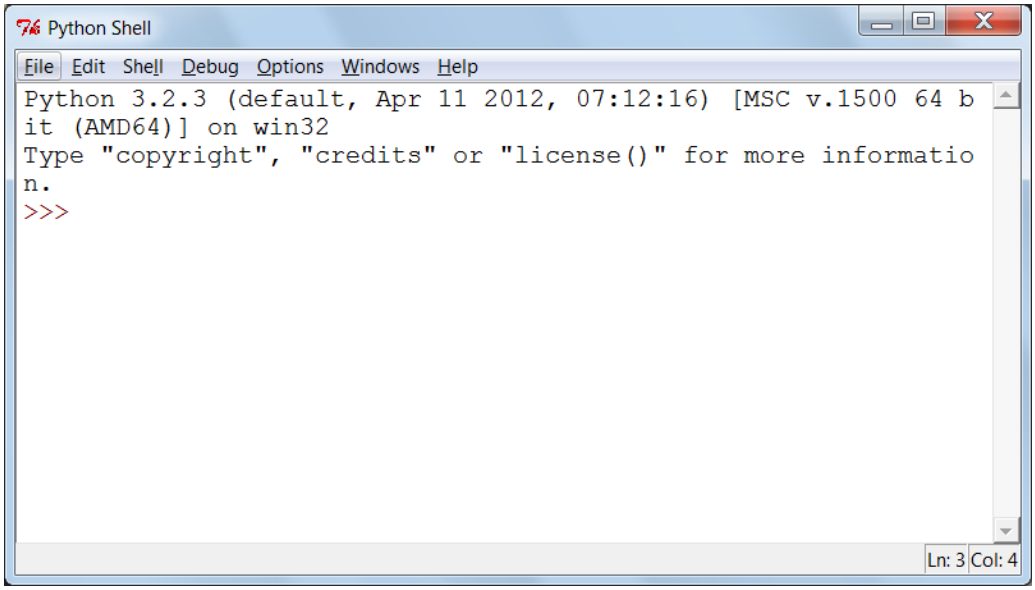
2. Writing a simple program, saving it, and executing it.

Note that all the statements you typed in the Python Shell window are kept by the Python program (also called Python interpreter) in its main memory. Main memory is volatile. This means that when you close the Python Shell window, all the statements (commands) that you wrote in it will be lost!

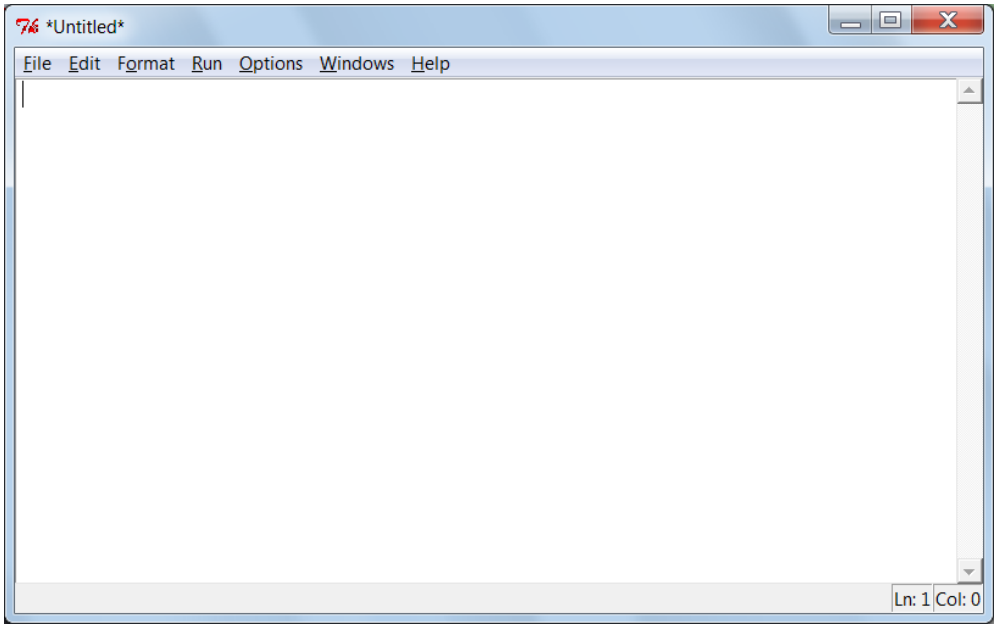
Now, suppose that you wrote (and tested!) a sequence of commands (statements) in the Python Shell window and that you want to execute them at a later time, say, after one day. How to do that? You need to save the statements in a **file** and then ask Python to execute the statements contained in that file.

2.1 Writing and saving a simple program

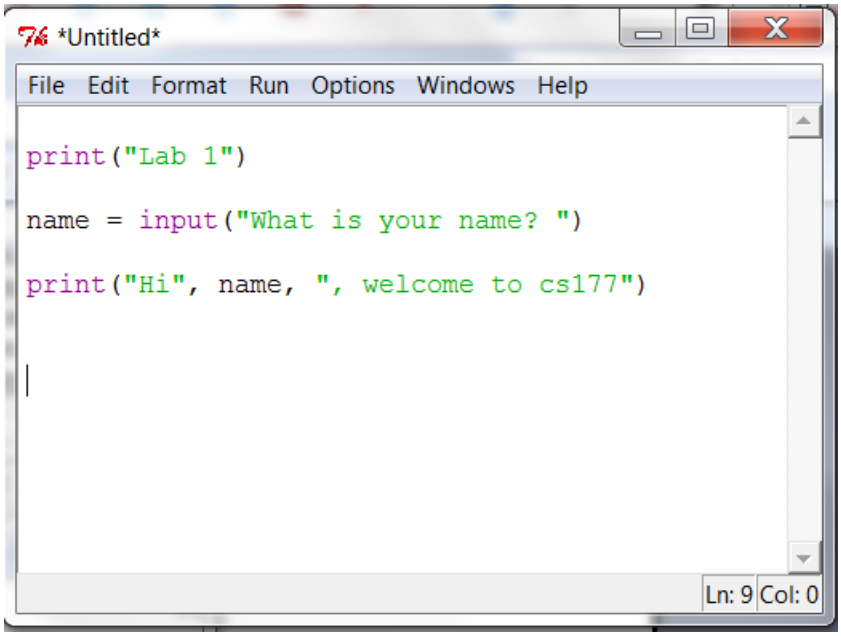
To write your simple program just click **File** in the Python Shell window



and then **New Window.** The following window will appear:



You can now write in this window the statement(s) you like, say:

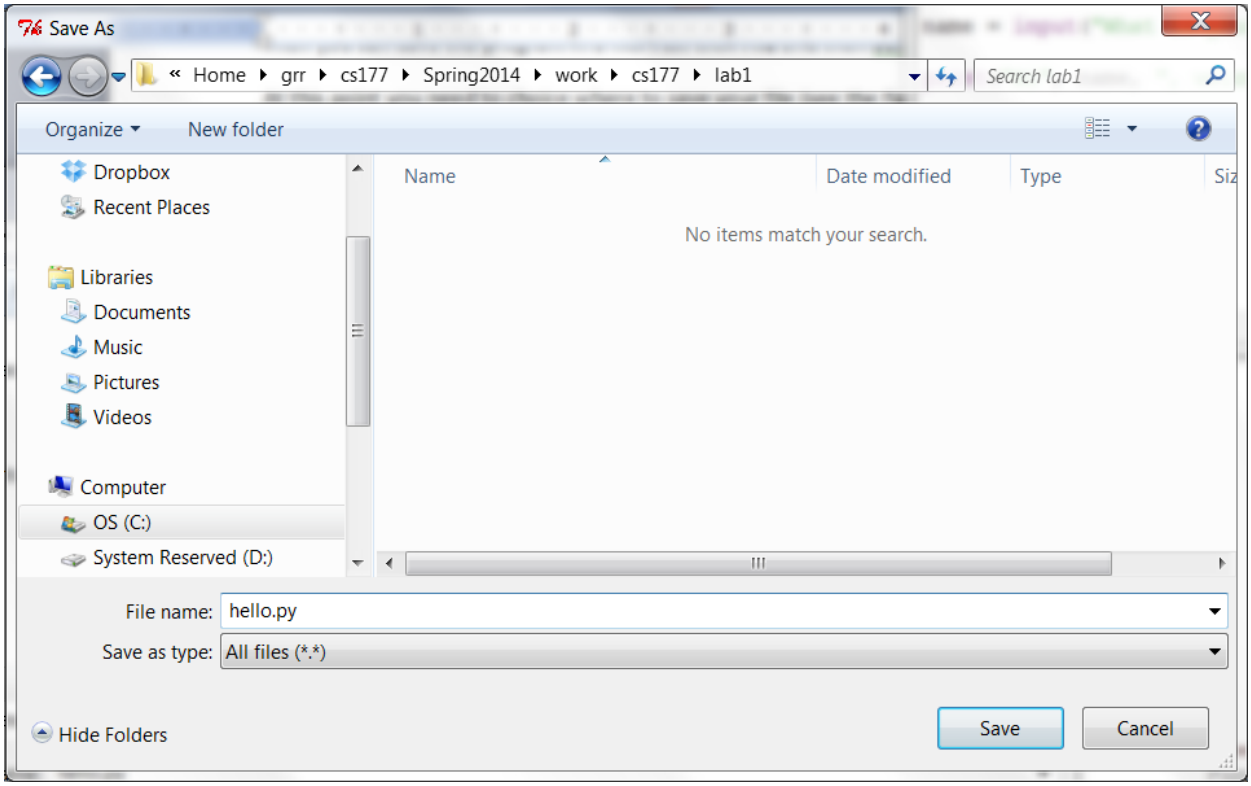


IMPORTANT NOTE:

You must **NOT** write the Python shell prompt sign (>>>) before your statement(s)!

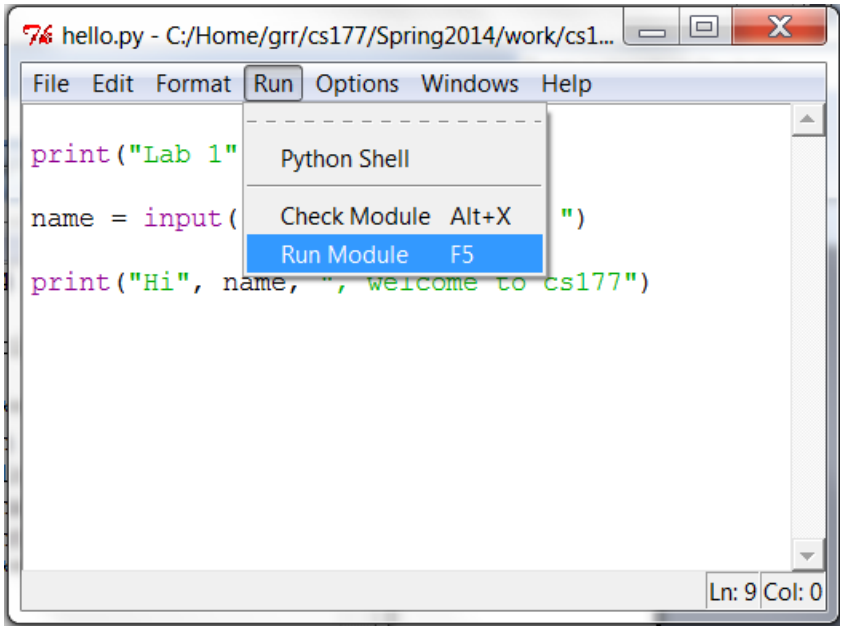
Then you can save the program in a file. Just click **File** and then **Save As…**

At this point you need to choice where to save your file (see the figure below) and assign it a name:

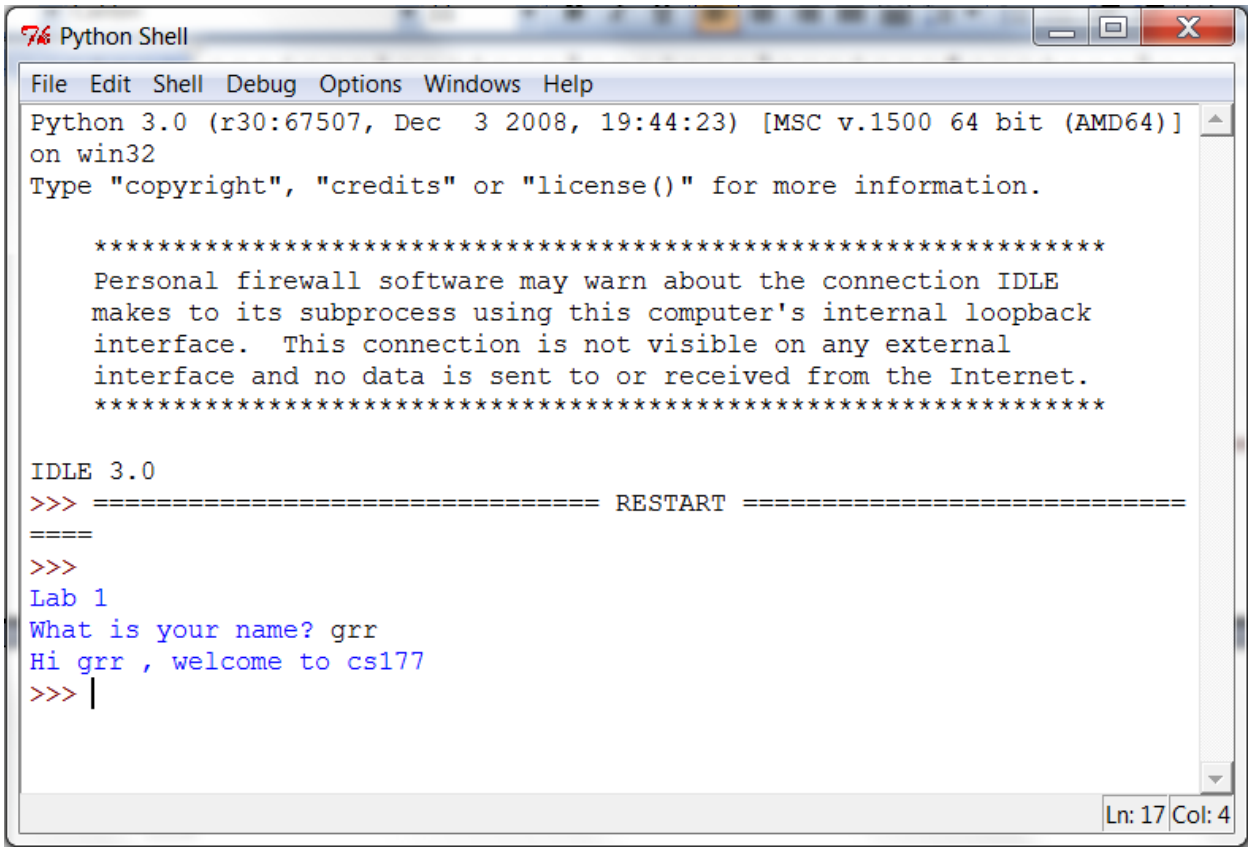


Navigate to the lab1 directory that you created during setup. Then type the File Name as “hello.py”.

Now, to execute this very simple program, in the editor window select **Run**->**Run Module**.

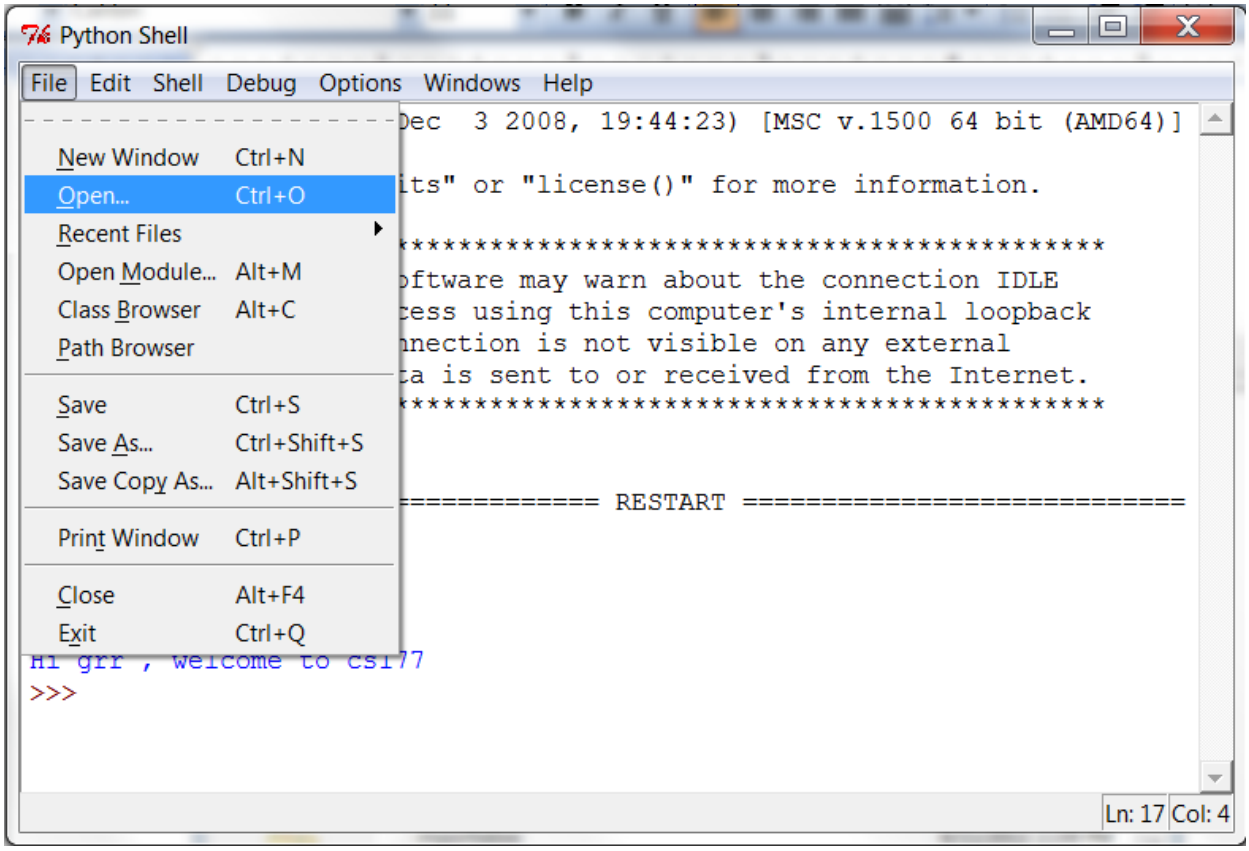


Now in the shell window type your name and type the <Enter> key.

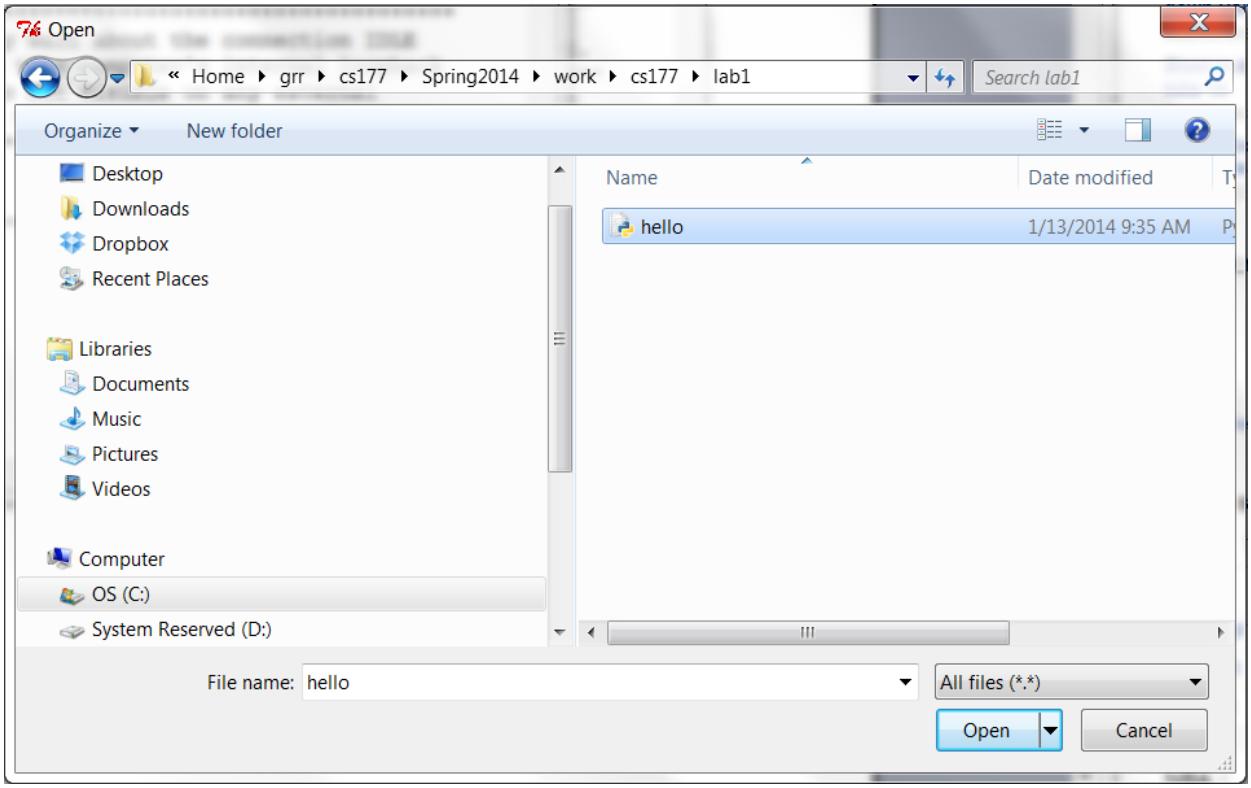


You have written your first program!

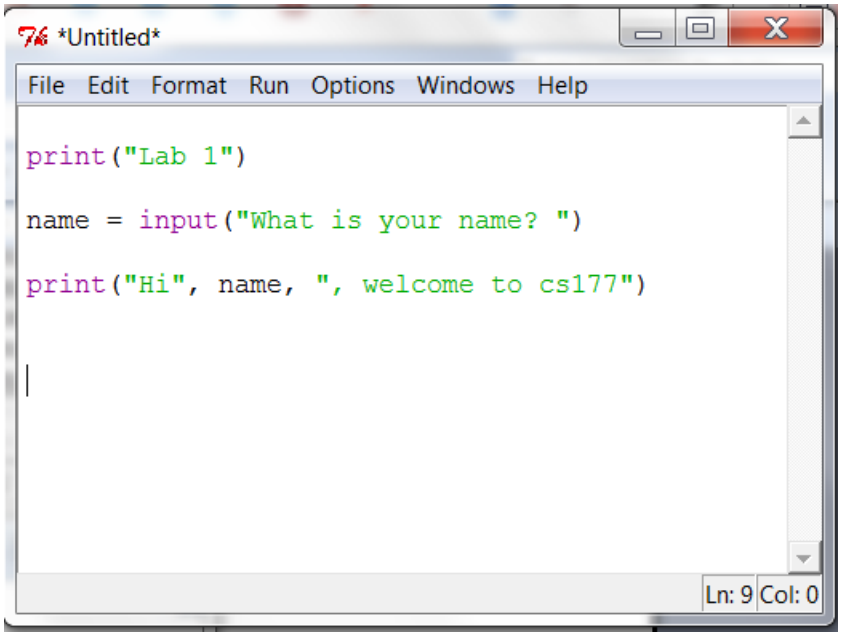
If you want to load a program that has already been saved, go to the “Python Shell Window” and selecting File->Open



Then navigate to the place where your program was saved.



This will open your file:



Then you may run it again by going to the Editor Window and selecting Run->Run Module as described before.

3. Saving your lab assignment(s) work.

You need to save each lab assignment you do in one (or more) files in your CS Windows machine.

To this end you need to create a folder named CS177 and, under it, folders named Lab1, Lab2, and so forth.

To do that, see the companion document setup-instruction.pdf.

4. Submitting your lab assignment.

At the end of the lab you need to submit the file(s) you created so that we can test the code you wrote and grade it.

To do that, see the companion document turnin-instruction.pdf