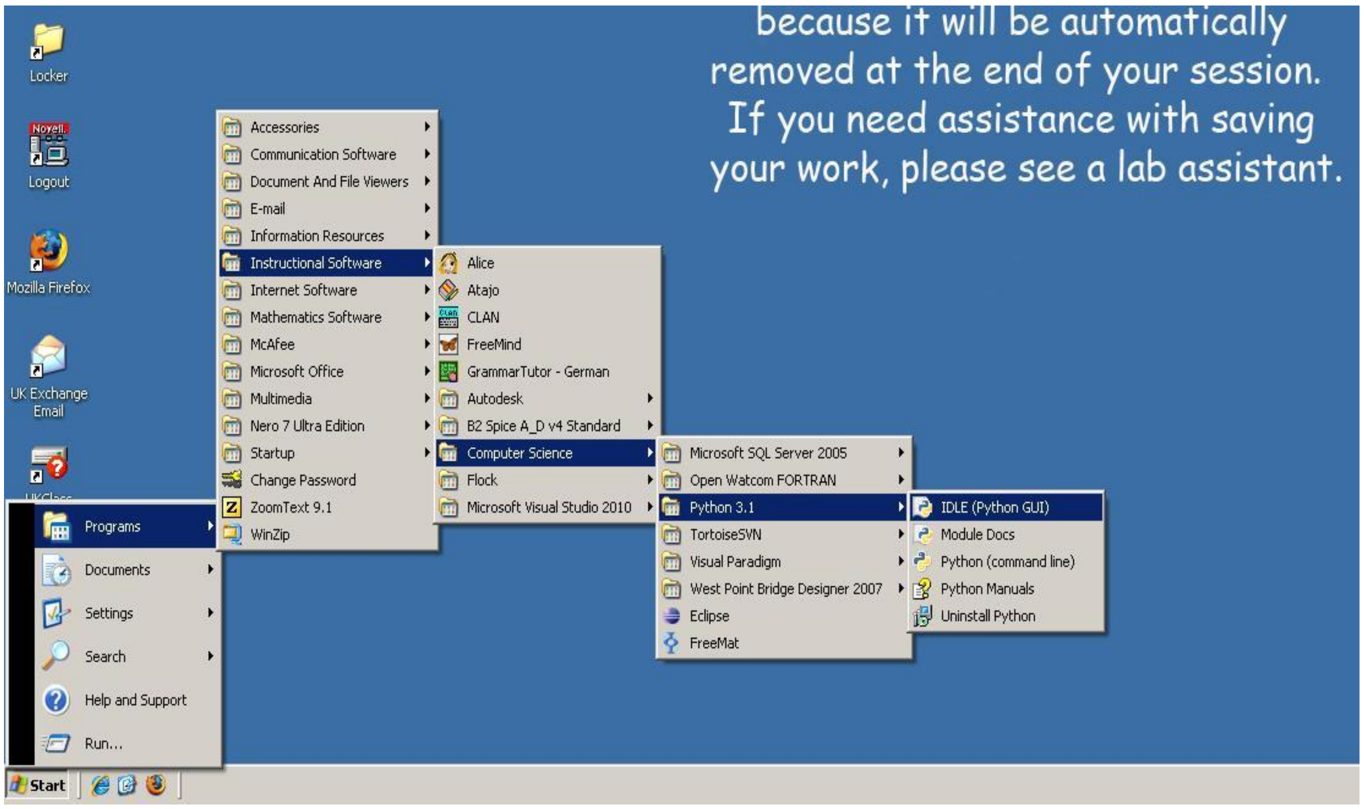
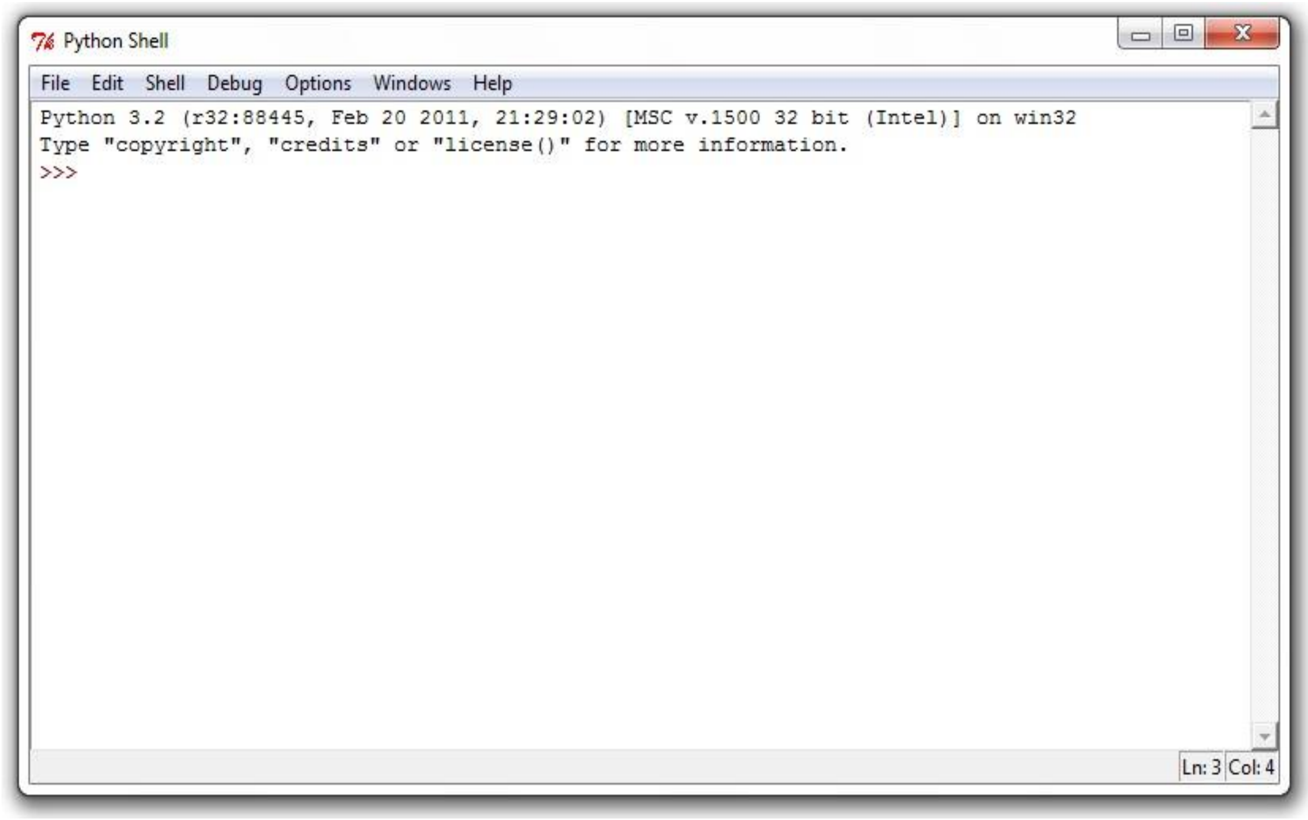
Introduction to the Python Development Environment

Python is freeware that can be installed on your home workstation or laptop. See the instructions at: https://www.python.org/downloads/. You can also reference our course *Python3 Installation* page.

The current version of Python is Python 3.x. The first thing we'd like to do is actually start running the Python program development tool named IDLE, which should be listed in installed programs under Python tab.


because it will be automatically removed at the end of your session. If you need assistance with saving your work, please see a lab assistant.

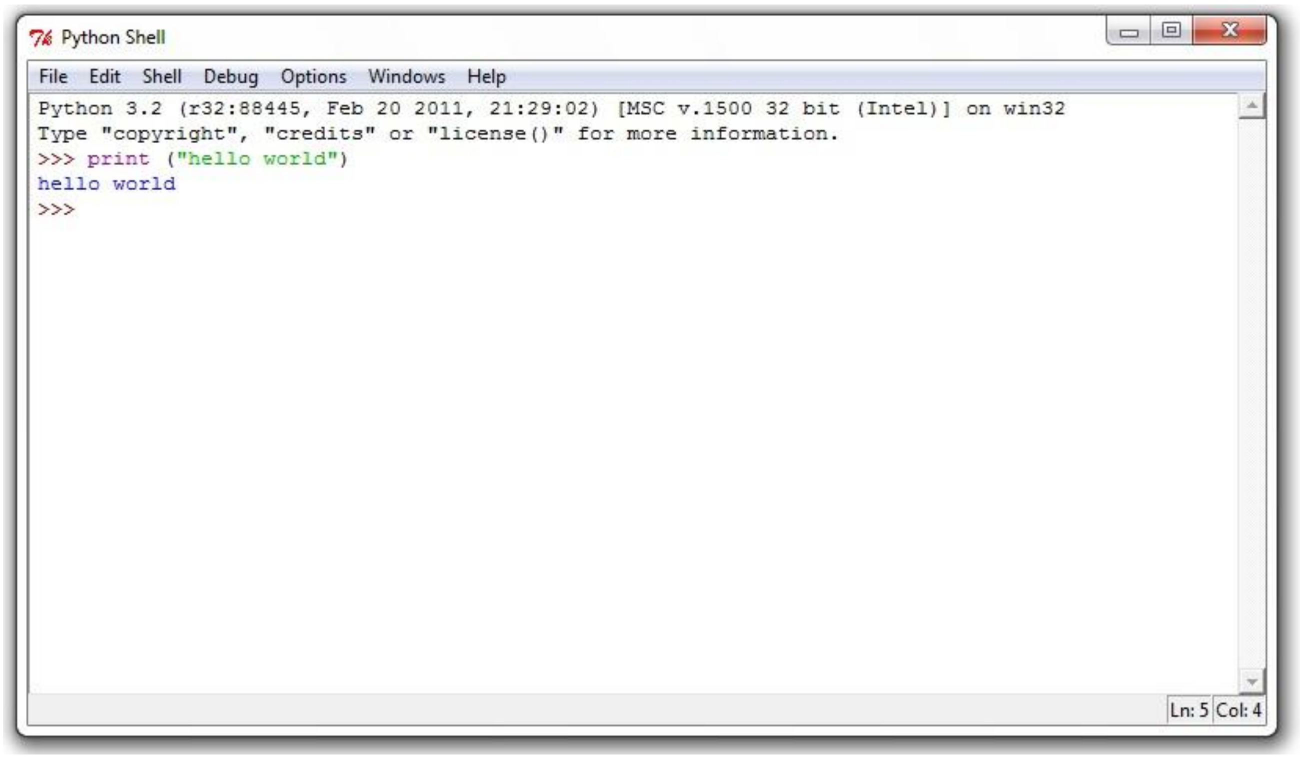
We'll see that a new window now will open up:



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

This is the main window to IDLE, and what we see right now is called the "Interpreter" (or "shell") window. The Interpreter allows us to enter commands directly into Python, and as soon as we enter in a command, Python will execute it and display its result. We'll be using this Interpreter window a lot when we're exploring Python: it's very nice because we get back our results immediately. If it helps, we can think of it as a very powerful calculator.

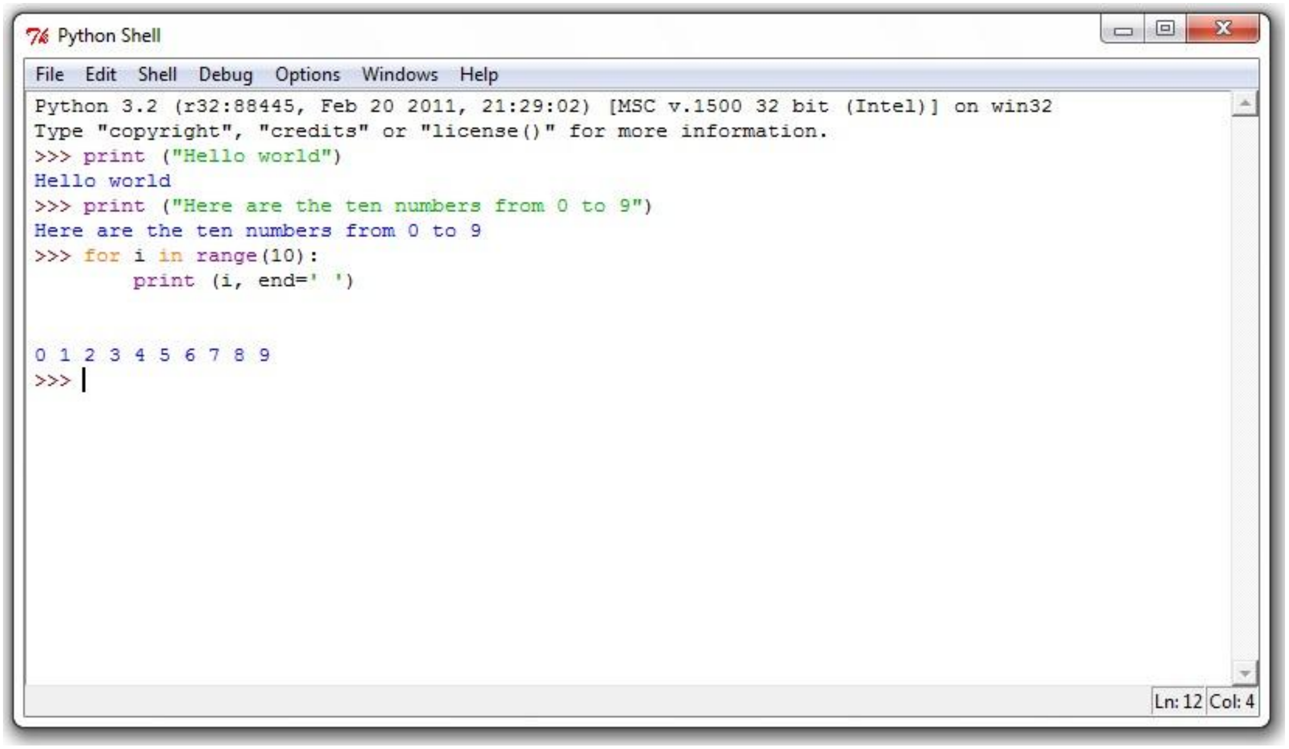
As per tradition, let's get Python to say the immortal words, "Hello World".



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print ("hello world")
hello world
>>>
```

Those '>>>' signs act as a prompt for us: Python is ready to read in a new command by giving us that visual cue. Also, we notice that as we enter in commands, Python will give us its output immediately.

Ok, this seems pretty simple enough. Let's try a few more commands. If we look below:



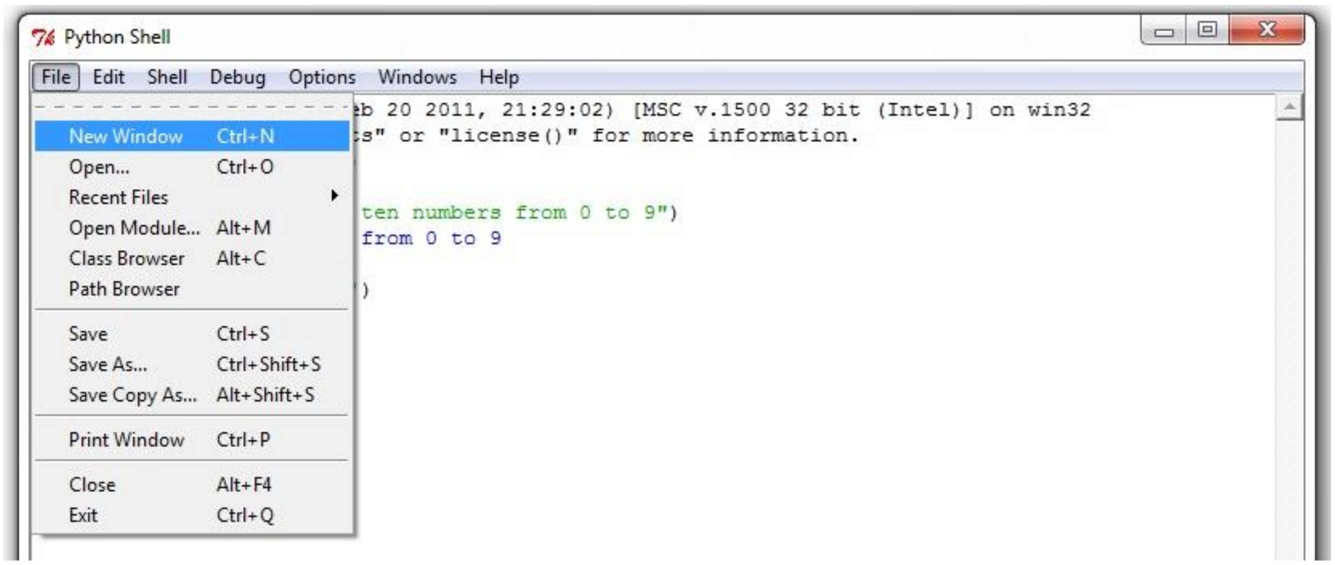
```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print ("Hello world")
Hello world
>>> print ("Here are the ten numbers from 0 to 9")
Here are the ten numbers from 0 to 9
>>> for i in range(10):
        print (i, end=' ')


0 1 2 3 4 5 6 7 8 9
>>> 
```

Note that the "print i," line is indented. This is a Python rule for the "for" statement that will be covered later. Don't worry too much about knowing the exact rules for making programs yet: the idea is that we can experiment with Python by typing in commands. If things don't work, then we can correct the mistake, and try it again.

If we close down Python and start it up again, how do we get the computer to remember what we typed?

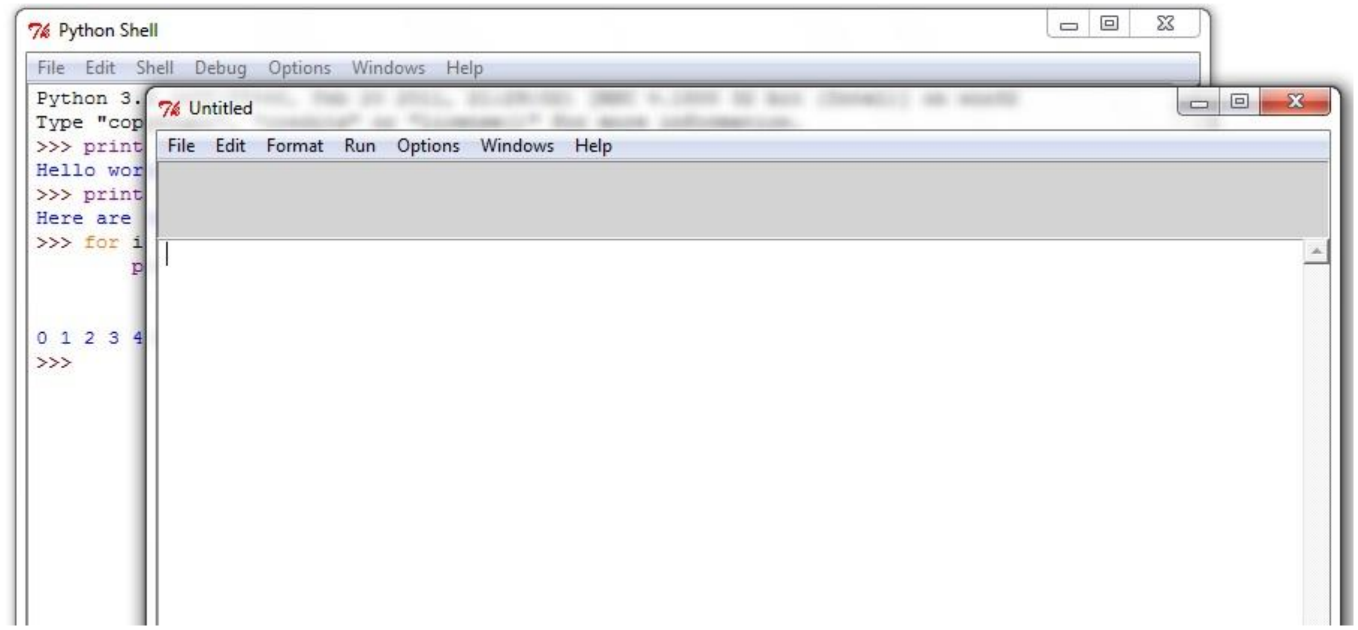
The solution is a little subtle: we can't directly save what's on the interpreter window, because it will include both our commands and the system's responses. What we'd like is to make a prepared file, with just our own commands, and to be able to save that file as a document. We can later open that file and "run" Python over it, saving us the time of retyping the whole thing over again.

Let's try this. First, let's start with a clean slate by opening up a new window.



```
eb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
ts" or "license()" for more information.


 ten numbers from 0 to 9")
 from 0 to 9

)
```

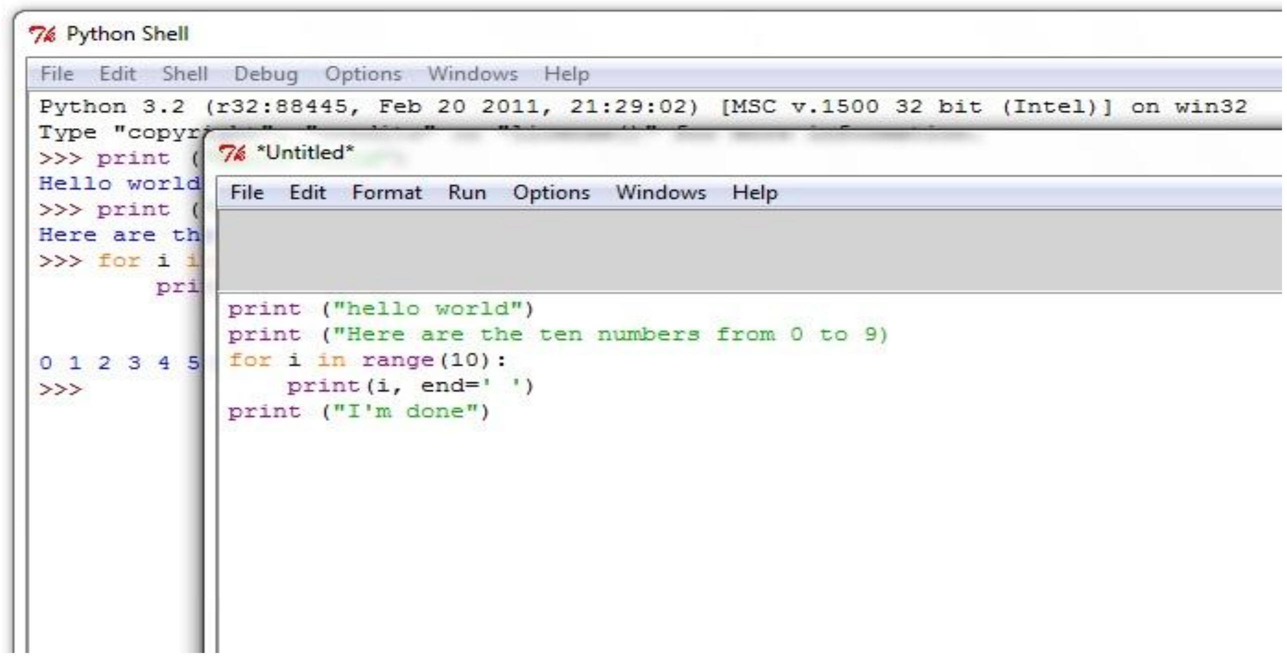
Here's the result of that menu command:


Python Shell
File Edit Shell Debug Options Windows Help
Untitled
File Edit Format Run Options Windows Help

```
Python 3.
Type "cop
>>> print
Hello wor
>>> print
Here are
>>> for i
        p


0 1 2 3 4
>>>
```

We notice that there's nothing in this new window. What this means is that this file is purely for our commands: Python won't interject with its own responses as we enter the program, that is, not until we tell it to. I'll call this the "Program" window, to distinguish it from the Interpreter window.

What we wanted to do before was save our program that we had tried out on the interpreter window. Let's do that by typing (or copy/pasting) those commands into our Program window. You can copy/paste by highlighting (hold the right mouse button down while you move the mouse over the text to be copied) what is to be copied, click on the "Edit" option to the right of the "File" option, click the "Copy" option, then in the Program window, move the cursor to where you want to copy the text, then click "Edit" and "Paste".

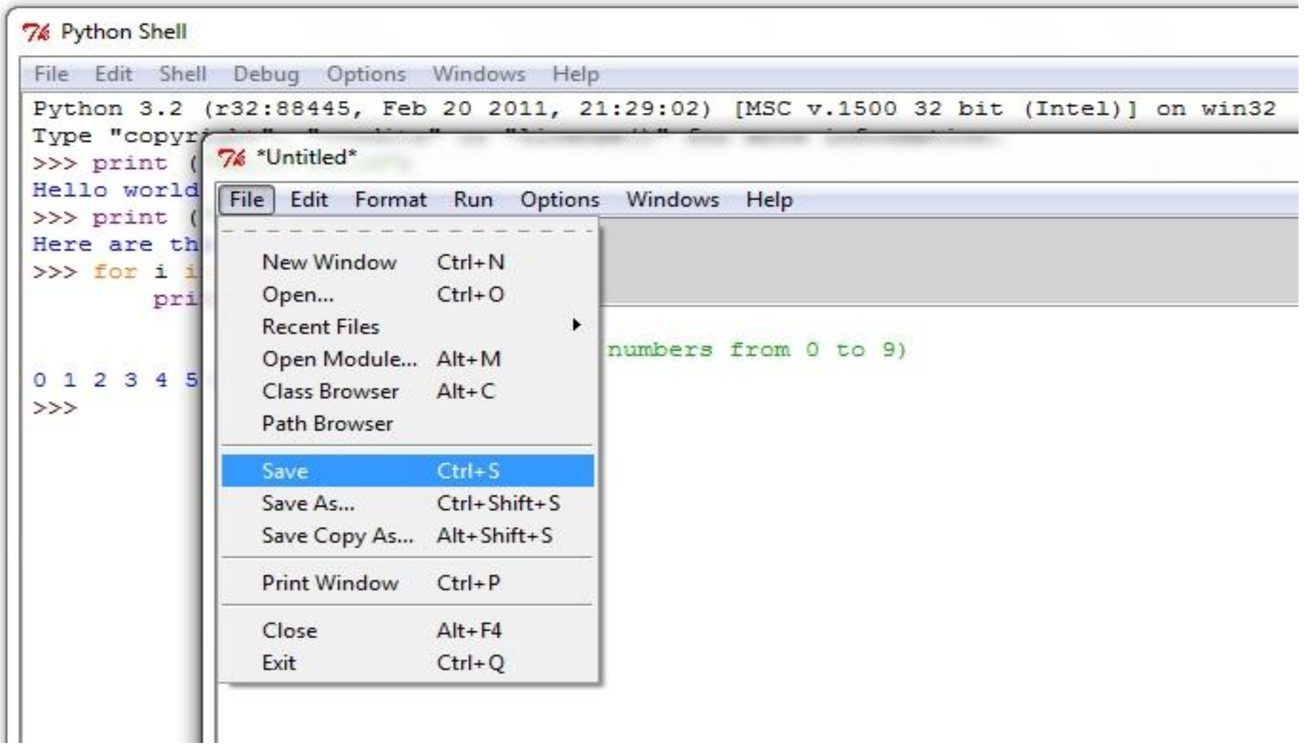

Python Shell
File Edit Shell Debug Options Windows Help
Untitled
File Edit Format Run Options Windows Help

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyr
>>> print (
Hello world
>>> print (
Here are th
>>> for i i
        pri


0 1 2 3 4 5
>>>
```

```
print ("hello world")
print ("Here are the ten numbers from 0 to 9)
for i in range(10):
    print(i, end=' ')
print ("I'm done")
```

Ok, we're done with copying and pasting. One big thing to notice is that we're careful to get rid of the ">>>" prompts because they're not really part of our program. The interpreter uses them just to tell us that we're in the interpreter, but now that we're editing in a separate file, we can remove the artifacts that the interpreter introduces.

Let's save the file now. The Save command is located under the File menu:



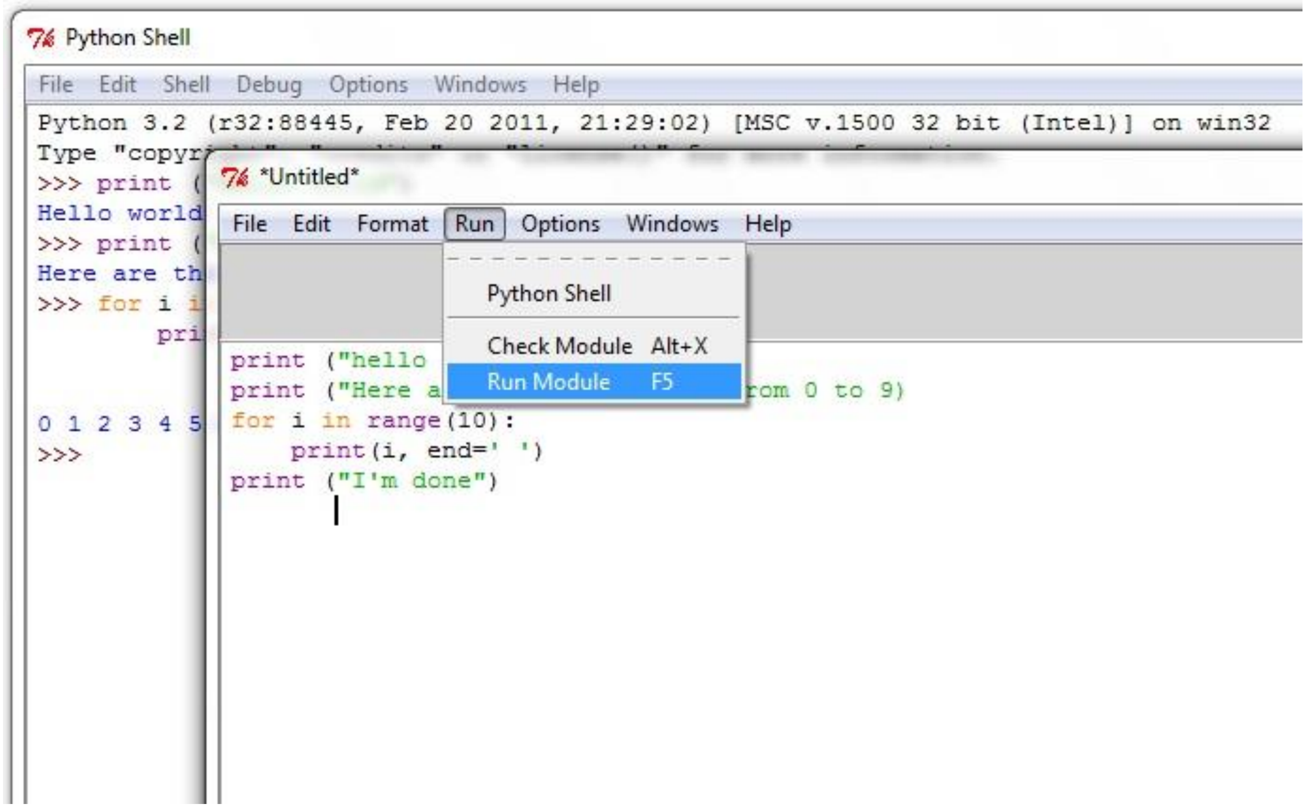
```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyr
>>> print (
Hello world
>>> print (
Here are th
>>> for i i
        pri

0 1 2 3 4 5
>>>
```

```
numbers from 0 to 9)
```

When you click "Save" you will be asked where to save the file. To save your programs for access when you leave the lab, save it to a flash drive or to your locker. For example, click "computer", click on the L drive, click on the folder where you want to save the file. Supply a file name for example: **yournameLab0.py**

Don't forget the file extension ".py". If you forget it, it will be more difficult to find the file when you want to reopen it later.

Now that we've saved the program, how do we run the program? If we look at the menus on our program window,

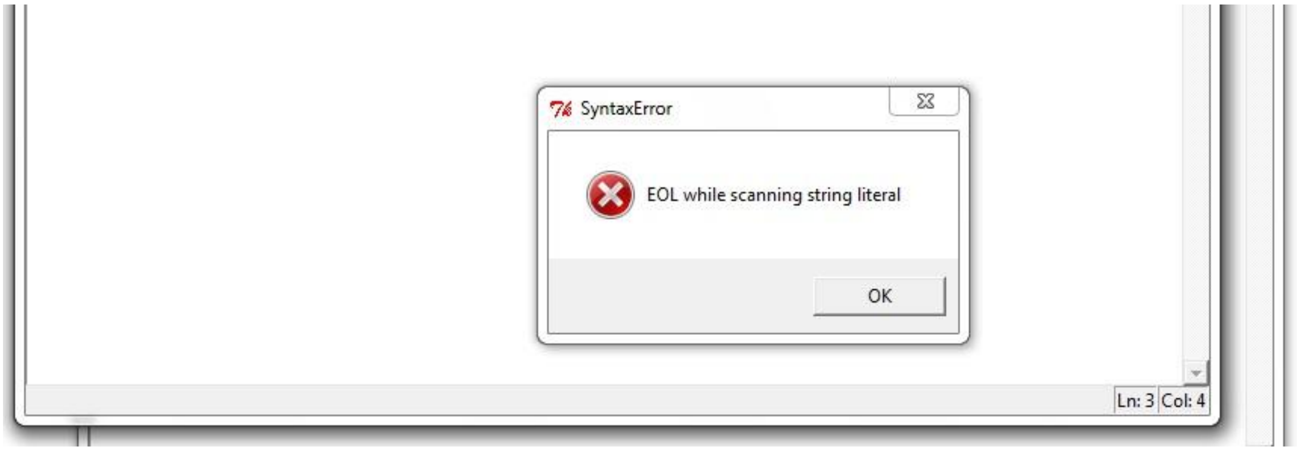

Python Shell
File Edit Shell Debug Options Windows Help
Untitled
File Edit Format Run Options Windows Help
Python Shell
Check Module Alt+X
Run Module F5

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyr
>>> print (
Hello world
>>> print (
Here are th
>>> for i i
        pri


0 1 2 3 4 5
>>>
```

```
print ("hello
print ("Here a                          rom 0 to 9)
for i in range(10):
    print(i, end=' ')
print ("I'm done")
```

we'll see that there's a menu option "Run Module", and that's what we'll do. What we want to see is Python running through the program, and displaying its results in the Interpreter window.

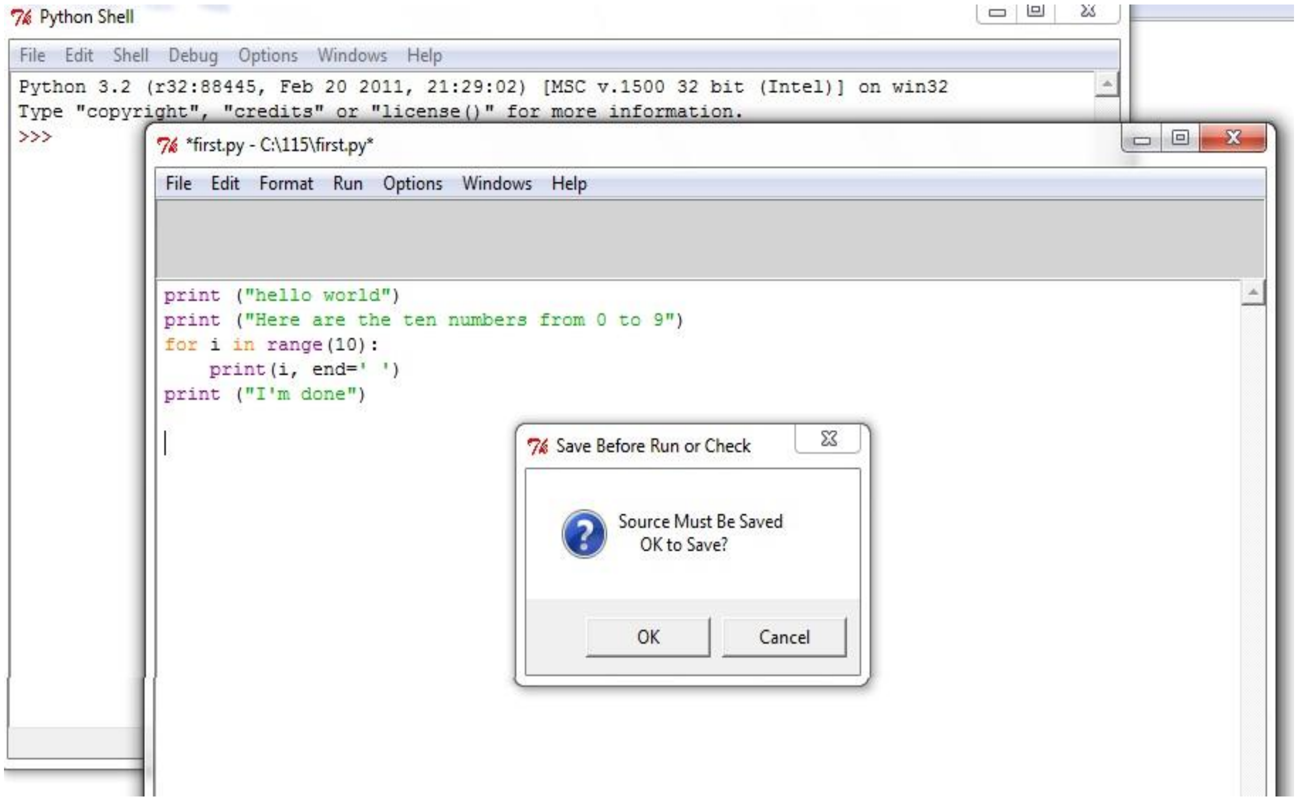
One thing to notice is that I made a typo: I didn't quite copy exactly what I had entered in the interpreter window before. Does this affect things? To see what happens when you have a typing error, remove one of the double quotes (") in the print statement as was done in this example.



EOL while scanning string literal

A programming language has a set of strict rules about what is allowed in programs written in the language. These are called "syntax rules". When you learned English grammar, you were learning English syntax rules. When you learn a language, that is half of what you have to learn (the other half is called "semantics" and you will see that later). If the Python interpreter finds that your program doesn't follow the rules, it will tell you about it. Sometimes the message is helpful, sometimes it is not. You will learn the rules pretty quickly because you get instantaneous feedback from Python. Python is often perceptive enough to direct us toward the problem, and in this case, it's telling us that we forgot to put something at the end of this line. In this case, we need to add an additional quotation mark. Let's add that in now.

Ok, let's say that we fixed that typo. Let's try to run the program again.


Python Shell
File Edit Shell Debug Options Windows Help

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

first.py - C:\115\first.py
File Edit Format Run Options Windows Help

```
print ("hello world")
print ("Here are the ten numbers from 0 to 9")
for i in range(10):
    print(i, end=' ')
print ("I'm done")
```

Save Before Run or Check

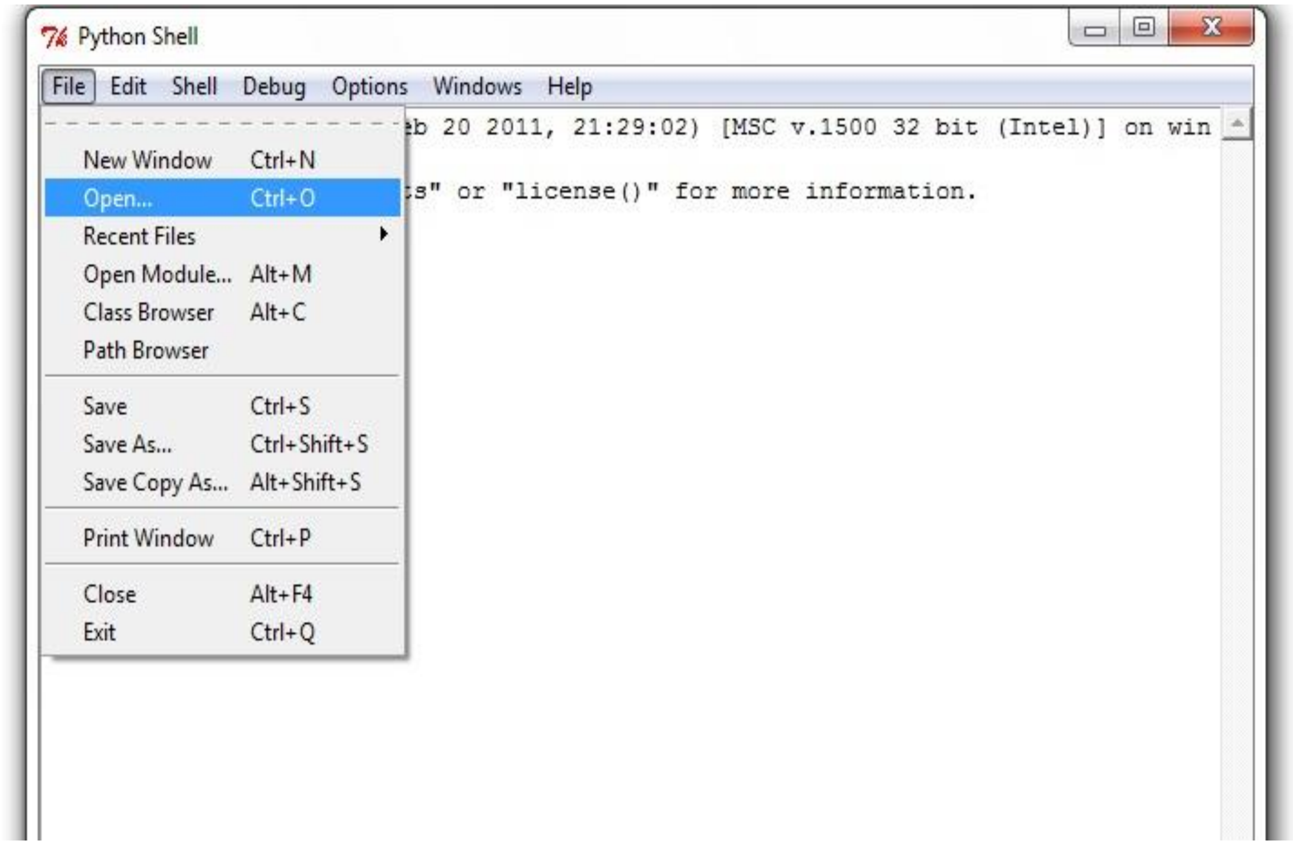
Source Must Be Saved
OK to Save?

You should always save your work before you run the program. You may be sure that your program is just fine, but if it has a logical mistake in it, it could be serious enough to make IDLE lock up or crash. That could cause you to lose all your recent work, if it hadn't been saved. IDLE is set up to make you save the work first!

They say that third time's the charm, so let's try running it again. Hopefully, it should run well now.

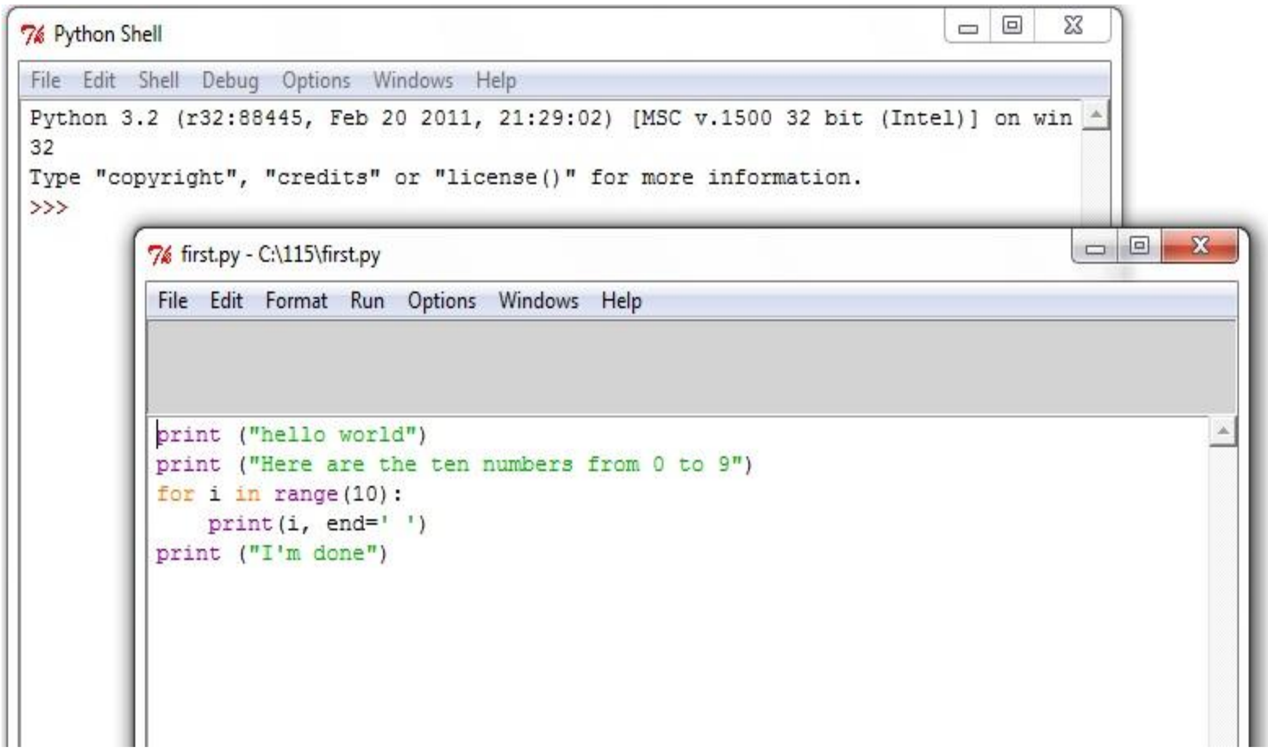
As we play with Python, we'll find ourselves "switching modes" between the Interpreter window and the Program window. The reason for this is because we can use the Interpreter as if it were a small laboratory and experimentally try small programs. After we're satisfied, we can save what we've learned into a program file.

You often work on a program more than one time. You can always reload your program file into Python if you saved it. You'll find the Open command under the File menu:



```
eb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
ts" or "license()" for more information.
```

then change to the location where you saved the file (flash drive or locker), and you should see a new Program window open up with your old program. You can save you old work and open it



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>>
```

```
print ("hello world")
print ("Here are the ten numbers from 0 to 9")
for i in range(10):
    print(i, end=' ')
print ("I'm done")
```

up at a later time.

This is all you need to know to start using IDLE to develop Python programs. This guide has skipped a lot of the features of IDLE: IDLE is much more than a mere editor, but it takes some

time to explore all of its features.