CSI Lab 02

Tuesday, January 21st

Objectives:

- Explore some basic functionality of python

Introduction

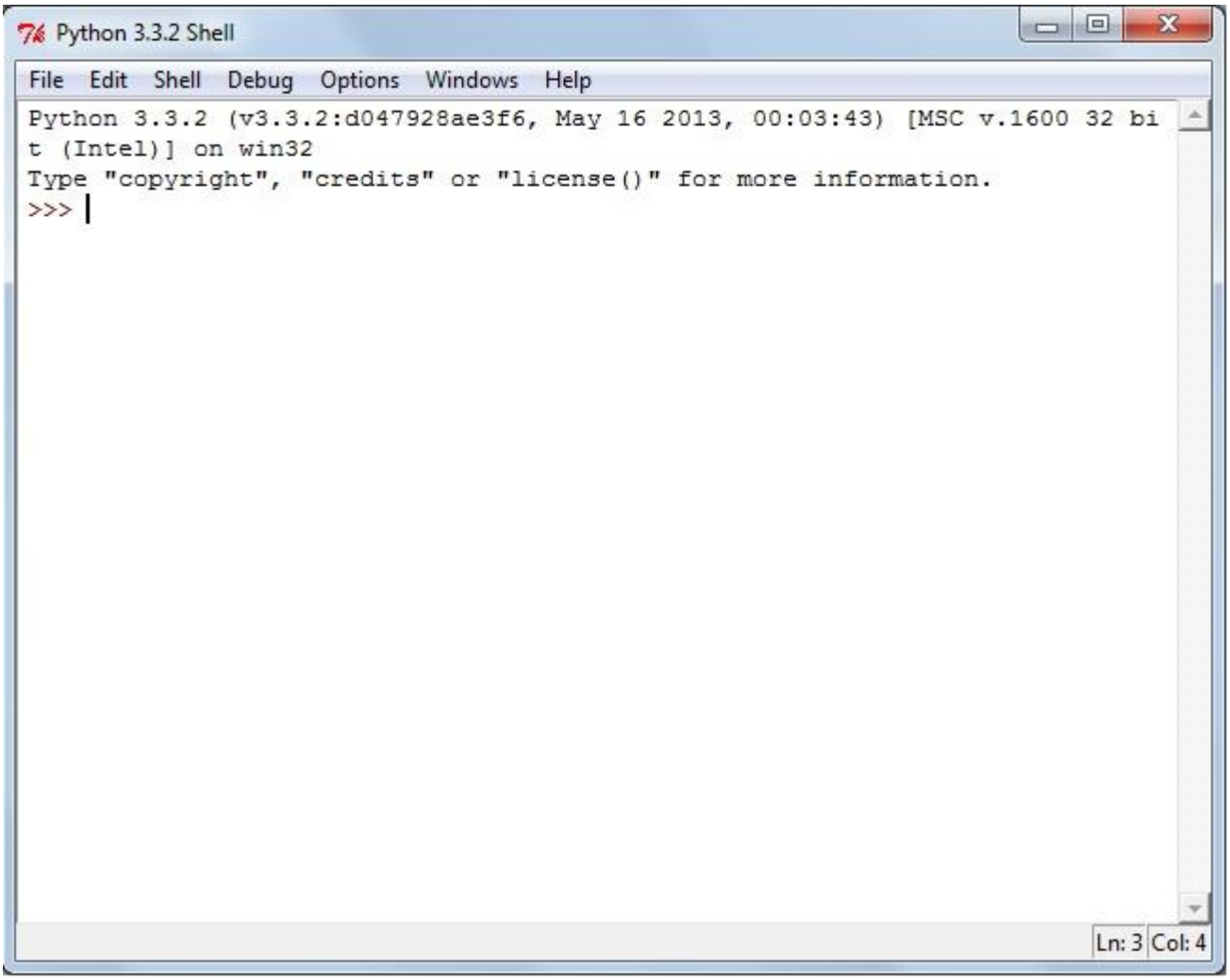
Last week we talked about the fact that a computer is, among other things, a tool to perform high speed arithmetic and logic operations. In order to learn how to write programs in Python, we need to explore the basic building block instructions for these operations, and how Python requires that we configure these operations (syntax).

During this lab you will complete a set of activities based on the material in chapter 1 in your textbook. You were supposed to have read chapter 1 already, but you probably didn't have the ability to actually DO what the book asked. In this lab you will actually DO activities like those listed in chapter 2.

Part A: Using IDLE as a Tool

During this course you will use IDLE (Integrated DeveLopment Environment) as a tool to write and explore Python code. IDLE is a free IDE (Integrated Development Environment) of the Python programming language. IDLE is **not**python -- instead, it is just one way to create and run python programs (like using Microsoft Word is just one way to create a document file). While you will not explore ALL of IDLE during today's lab, we want to get started so you have a little bit of a feel for how things work.

1. Assuming that you are sitting at a Windows computer, launch IDLE by selecting "Start | All Programs | Programming | Python 3.3 | IDLE (Python GUI)"
2. When IDLE starts up it should look like this:



The good news is that it will look like this whether you are in the labs on a Windows machine or at home on your Windows, Mac, or Linux based machine. Please remember that Python is free to download and you SHOULD install version 3.3 on your computer at home if you have the option! (Download from [www.python.org)](http://www.python.org)

This is Python's interactive mode window (often called "the shell"). In this mode of working with Python you can type in almost ANY valid Python statement at the prompt (the ">>>"). When you do so, and then hit the Enter key, you send that command off to the Python interpreter to be converted to machine language and then executed (assuming that it had proper syntax and the semantics were meaningful to the interpreter).

Let's try some very basic commands. In this shell window, at the command prompt type

```
print("Hello World!")
```

When you hit Enter, IDLE goes off to interpret what command(s) you gave it. In this case, it interprets your command as an instruction to print (to the screen, not the printer) whatever message is included inside of the parentheses and quotes (in computer vocabulary we normally refer to the message between quotes as a ***String***).

[Q1] What happens after you press the Enter key? (be very specific)

Let's try another commands. At the prompt type:

```
1 + 2
```

[Q2] What happens after you press the Enter key? (be very specific)

Of course, if you typed something into the shell that IDLE doesn't recognize, you will get an error message. To demonstrate this, type:

```
prin("Hello World!")
```

(notice that I mis-spelled print). This gives you a message that should say something like:

```
>>> prin("Hello World")
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    prin("Hello World")
NameError: name 'prin' is not defined
>>>
```

Whenever you get a message like this, pay close attention to what I ASKED you to type and what you ACTUALLY typed. While sometimes I might have a mistake in my lab, more often, you will have made a typo.

Activity B : Python Arithmetic

So we have talked about the fact that Python is very good at high speed arithmetic and logic operations. Let's look at how python works with numbers.

[Q3] Enter each of the following mathematical statements at the command prompt and press enter. Record what "result" the computer returns in response to your command

Arithmetic Expression	Results via the command prompt
15 + 3	
15 - 3	
15 * 3	
15 / 3	

The results here shouldn't surprise you once you recognize that the asterisk (*) and the forward slash (/) are the operators that python uses for multiplication and division.

Unlike "high school mathematics" - where there are four binary operators (addition, subtraction, multiplication, and division) - python regularly uses seven binary operators. These additional operators include the **, // and % symbols.

[Q4] Enter each of the following mathematical statements at the interactions pane prompt and press enter. Record what "action" the computer takes in response to your command

Arithmetic Expression	Results via the command prompt
2 ** 1	
2 ** 2	
2 ** 3	
2 ** 4	
3 ** 1	
3 ** 2	
3 ** 3	
3 ** 4	

[Q5] Given what you observed in the previous step, write a short explanation of how the ** operator works in Python.

[Q6] Enter each of the following mathematical statements at the interactions pane prompt and press enter. Record what "action" the computer takes in response to your command

Arithmetic Expression	Results via the command prompt	Arithmetic Expression	Results via the command prompt	Arithmetic Expression	Results via the command prompt
8 / 4		8 // 4		8 % 4	
9 / 4		9 // 4		9 % 4	
10 / 4		10 // 4		10 % 4	
11 / 4		11 // 4		11 % 4	

12 / 4		12 // 4		12 % 4	
13 / 4		13 // 4		13 % 4	

[Q7] Given what you observed in the previous step, write a short explanation of how the /, //, and % operators work in Python.

For the most part, mathematics in python works the way that it worked in your high school algebra class. That is, the order of operator precedence in Python (from highest to lowest) is:

- Operations in parentheses
- Exponentiation (**)
- **Unary** negation (-) and positive (+)
- Multiplication (*), division (/), integer division (//) and remainder (%)
- **Binary** Addition (+) and subtraction (-)
- Assignment operator (=)

[Q8] Given this knowledge, **predict** what will happen when you invoke the following statements. **After** making your prediction, enter the statement at the interactions prompt and check if you were correct. If you were incorrect, determine why.

Arithmetic Expression	Predicted Results	Actual Results via the command prompt
4 * 3 + 2 * 9 / 3		
4 * 3 + 2 * (9 / 3)		
4 * (3 + 2) * 9 / 3		
20 / 4 * 6 / 2		
20 / 4 * (6 / 2)		
(20 / 4) * 6 / 2		
(20 / 4) * (6 / 2)		
(2 - 3 + (2 * (12 / 4) + 1))		
5 - (2 + 5) * 10 / 2		
4 + 2 ** 5 - 3		
- 3 / 5 - 9 % 4		
- 3 / + 5 - 9 % 4		
12 / 4 * -3 + -1		

Activity C : Print Statements and Strings

Strings

A String is a sequence of characters (letters, numbers, etc). Print statements are intended to display these Strings.

[Q9] Enter the following statements into the interactions pane and observe what happens (note, one of these will cause an error).

Print Expression	Results via the command prompt
print("Computer Science")	
print("Computer" + "Science")	
print("Computer" , "Science")	
print("Computer , Science")	
print(5+3)	
print("5 + 3")	
print("5" + "3")	
print("5" , "3")	
print("5 + 3 = " , 5+3)	

print("5 + 3 = " + 5 +3)	
print("5 + 3 = " + str(5+3))	

[Q10] What happens when you join two strings together using the plus sign? What happens when you join two strings together using the comma?

Activity D : Writing a simple program

Suppose you needed to perform mathematical calculations to calculate information about a particular sphere. If you remember your high school Geometry class you will remember these formulas are:

$$c = \pi \cdot d = 2 \cdot \pi \cdot r.$$

$$surface = 4\pi \times radius^2$$

$$volume = \frac{4}{3}\pi \times radius^3$$

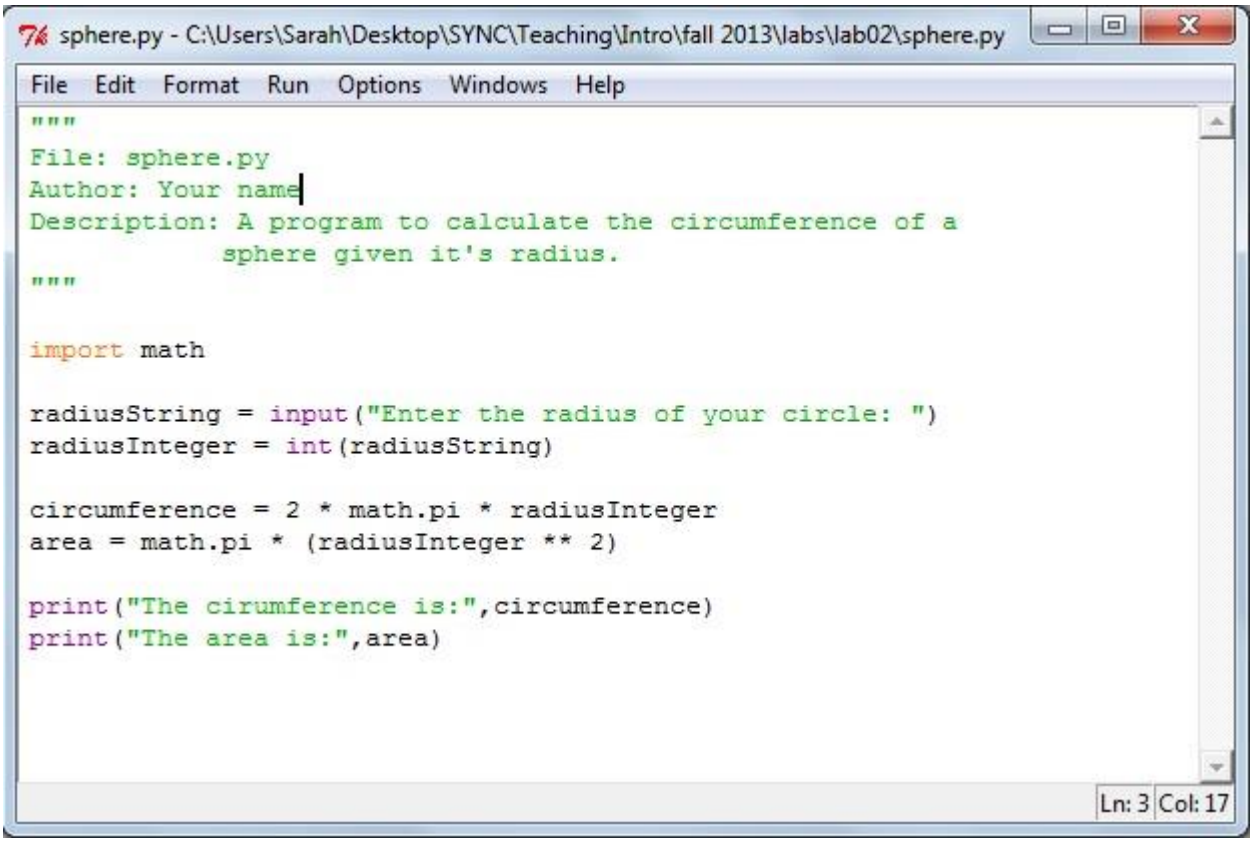
Let's look at how we might write the instructions to calculate the circumference of a sphere.

1. We need to know the radius of the sphere.
2. We need to calculate the circumference using the formula above
3. We need to print out the results

We want to have the programmer write an algorithm, translate it to a working program, and than allow the user to run our program (in this context, called a script).

In the python shell window open a new programming window by selecting "File | New Window"

In this window type in the Python program below which calculates a sphere's circumference from a provided radius. (Note: This is ALMOST Code Listing 1.1 from your textbook except I have added a header block to fit the style we will use in this course and I have changed the way some of the printing is done to make it easier to modify/add later.).



When you have typed this in make sure that you save it. I would suggest saving it to your flash drive as "sphere.py" (since that's what the comment at the start of the program says it will be called.)

When you are ready to run this program, select "Run | Run Module" (notice you can achieve the same result by pressing the F5 key)

Ok, this was a GREAT start, but we know that we can get it do to much more. After you get this working properly you should go back to your window with the sphere.py program in it and **add** in code to calculate and print information about the :

- diameter (twice the radius)
- surface area

- volume

You should not only make sure that your program loads and runs, but you should make some calculations by hand to confirm that you are producing the correct results.

[SIG1] When you are happy with your results from Activity D, raise your hand and demonstrate your program.

Activity E : You create one from scratch

You are planning a trip to Europe. Your bank will exchange US Dollars for Euros by first subtracting a $10 service fee from your US Dollars and then processing the conversion on the remainder of your US Dollars at the rate of 0.781. This means that if you give them $110 they will give you 78.10 in Euros. If you give them $500 they will give you 382.69 Euros.

Write a script, in a file called currencyConverter.py, that:

- takes in an amount of US dollars,
- calculates the equivalent value in Euros given the situation above,
- and prints a nice message with this information.

To get a new file in IDLE select "File | New Window" This will launch a blank code editor.

- Select "File | Save As.." and give it the name currencyCoverter.py.
- Add the standard header similar to the one you see in sphere.py in Activity D.
- Add the code.
- Run and test your code

[SIG2] When you are happy with your results from Activity E, raise your hand and demonstrate your program.

Finishing Up

Congratulations!

Whew!! You accomplished a lot this week.

You are now free to stick around and use the machines to try some other examples provided in your textbook, check the class web site, check your email, browse the web etc. On the other hand, you are also free to leave early.

When you are ready to leave, make sure that your activity log is completed before submitting it to Dr. Diesburg.

Do not forget to "log off" of the machine you are working on.