Lab 1 Instructions

Getting Started with JES

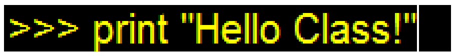
Getting Started

Open your laptop, launch VirtualBox and start Windows. Make sure JES is installed. If JES is not installed, download the Windows installer from the course web site (http://uni.edu/sergey/courses/cs1120) and run the installation.

Activity A: Understanding how to use the Command Area

As you remember from your readings, the black area at the bottom of the JES window is known as the Command Area (also sometimes referred to as the Interactions Panel). Anything that you type in this area is immediately evaluated (as soon as you hit the ENTER key). If you have typed in something that JES recognizes as a valid command than JES attempt to execute the command.

Let's start this lab by practicing a couple of very basic commands. In the command area type the following and hit "ENTER"

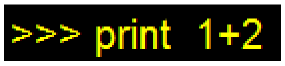


```
>>> print "Hello Class!"
```

When you hit Enter, JES 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 quotation marks. We normally refer to the text between quotes as a ***String***.

[Q1] What happened after you pressed the Enter key? (Be very specific)

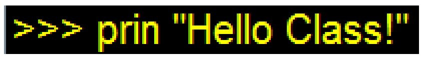
Let's try another command. At the prompt type:



```
>>> print  1+2
```

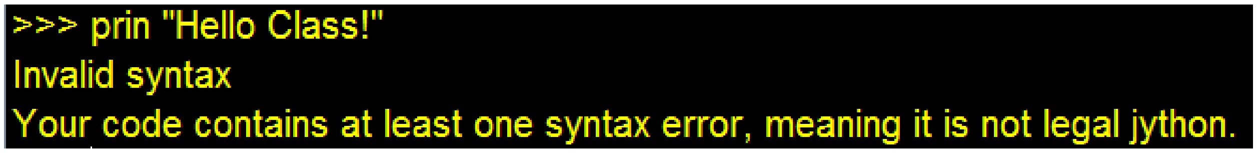
[Q2] What happens after you press the Enter key? (Be very specific)

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



```
>>> prin "Hello Class!"
```

Notice that I mis-spelled print. Python/JES does not recognize any command named "prin" Therefore, you should get a message that says something like:



```
>>> prin "Hello Class!"
Invalid syntax
Your code contains at least one syntax error, meaning it is not legal jython.
```

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

A big part of manipulating media will be performing basic mathematical calculations. Therefore, it's important to understand how basic arithmetic works in Python/JES. This activity will introduce you to the basic arithmetic operators we have available to us. You will learn that they are very similar to the four arithmetic operators that you learned in elementary school, but that there are a few small but significant differences we must make sure we observe.

[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**
print 15 + 3	
print 15 - 3	
print 15 * 3	
print 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.

However, it turns out that division may surprise you when you play with it a little more.

[Q4] 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**
print 15 / 3	
print 16 / 3	
print 17 / 3	
print 18 / 3	
print 19 / 3	

These numbers might confuse you at first (Unless, of course, you remember this from our previous class, and/or you did a really good job reading the textbook, where this "strange" answer is explained and it turns out it isn't strange at all).

It turns out that most computer programming languages care what “type” of number you are working with. Computers normally differentiate between at least two types of numbers:

- Integers – Any whole number. Think about the numbers that you learned when you learned to count (1, 2, 3, …), the negative version of these numbers (-1, -2, -3…) and the number zero.
- Floating point numbers(or floats) – In high school math you learned to call these the “real” numbers. These are any number that can be represented as a fraction.

The easiest way to tell the difference between integers and floats is to look for a decimal. If it has a decimal, than it’s a float. It if doesn’t have a decimal, than it’s an integer.

In JES – and it’s important to point out here that this distinction exists in JES but may not exist in other programming languages that you might learn – if you perform a mathematical operation on two integers, the answer MUST be represented as an integer. This is why when you did 16/3 you got 5 as an answer rather than the 5.333… that you might have been expecting.

> Think of it this way – When you first learned division you looked at a problem like 15/3 and asked yourself, “how many items would be in each pile if I divided 15 items into three even piles?”
>
> The answer, of course, is five.
>
> Then you learned that 16/3 should be thought about as “how many items would be in each pile if I divided 16 items into three even piles?”
>
> The answer, of course, is five, with one left over.

Actually, that last little bit leads us to the introduction of a new mathematical operator that exists in many programming language. In python the percent symbol is used as a mathematical operator that has NOTHING to do with percentages.

[Q5] Enter each of the following mathematical statements at the command area and press enter. Record what "action" the computer takes in response to your command

Arithmetic Expression	**Results via the command Results via the command prompt**	**Arithmetic Expression**	**Results via the command Results via the command prompt**
8 / 4		8 % 4	
9 / 4		9 % 4	
10 / 4		10 % 4	
11 / 4		11 % 4	
12 / 4		12 % 4	
13 / 4		13 % 4	

[Q6] Given what you observed in the previous step, write a short explanation of what the % operator does in Python.

This % operator may seem strange to you, but it actually useful in a number of situations. Suppose we had a number and we wanted to know if it was odd or even. How would we do that? It turns out we can do that by using % with a quotient of 2.

[Q7] Enter each of the following mathematical statements at the command area and press enter. Record what "action" the computer takes in response to your command

Arithmetic Expression	**Results via the command prompt**
8 % 2	
9 % 2	
10 % 2	
11 % 2	
12 % 2	
13 % 2	

[Q8] What do you notice about even numbers? What do you notice about odd numbers?

Let's consider another example. Suppose that you had a "large" number and you were interested in only part of the number. You can use / and % with powers of 10 to pull out interesting parts of the number. You won't do this much this semester, but it's a helpful trick.

[Q9] Enter each of the following mathematical statements at the command area and press enter. Record what "action" the computer takes in response to your command

Arithmetic Expression	**Results via the command prompt**
12345 / 10	
12345 % 10	
12345 / 100	
12345 % 100	
12345 / 1000	
12345 % 1000	

In fact, notice that with combinations of these operations you can get any single number out of our "big" number.

[Q10] Enter each of the following mathematical statements at the command area and press enter. Record what "action" the computer takes in response to your command

Arithmetic Expression	**Results via the command prompt**
12345 % 10	
12345 / 10 % 10	
12345 / 100 % 10	
12345 / 1000 % 10	
12345 / 10000	

Ok, the previous math might bother some of you. You are used to "normal" math where you get decimals when you divide 16/3. So what do you do if you want the answer to actually be 5.333? Well, the "correct" way to get a floating point answer (a number with a decimal) is to do a mathematical operation between two floating point numbers.

[Q11] 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**
print 15.0 + 3.0	
print 15.0 – 3.0	
print 15.0 * 3.0	
print 15.0 / 3.0	

But it turns out that as long as ONE of the two operands (the fancy math term we use for the two numbers we are operating upon) is a floating point number, than the answer will be a floating point number.

[Q12] 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**
print 16.0 / 3.0	
print 16 / 3.0	
print 16.0 / 3	
print 16 / 3	

All of this may seem a little bit weird, but it's important that you start the semester understanding these distinctions. If you can learn the meanings and results of these operations early in the semester then you can avoid all sorts of mistakes later on. Not only that, but you can take advantage of these strange ways that numbers work in order to make it easier to write certain kinds of actions.

Unlike "high school mathematics" - where there are four binary operators (addition, subtraction, multiplication, and division) - JES regularly uses six binary operators. You learned the fifth of these up above. The last of these is the ** symbol. While this looks like a simple mistake in using the multiplication symbol, it is not.

[Q13] Enter each of the following mathematical statements at the command area 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	

[Q14] Given what you observed in the previous step, write a short explanation of how the ** operator works 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 (/), and remainder (%)
- **Binary** Addition (+) and subtraction (-)
- Assignment operator (=) [We will learn this one next week]

[Q15] 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**
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) * 10/3		
4 + 2**5 - 3		

4 + 2**(5 – 3)		
5 / 3 – 9 % 4		
12 / 4 * -3 + -1		

Part C: Print Statements and Strings

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

[Q16] 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 "Media Computation"	
print "Media" + "Computation"	
print "Media" + " "+ "Computation"	
print 5+3	
print "5 + 3"	
print "5" + "3"	
print "5 + 3 = " + 5+3	
print "5 + 3 = " + str(5+3)	

[Q17] What happens when you join two strings together using the plus sign?

[Q18] What happens when you join a string and a number together using the plus sign?

[Q19] What do you guess the computer is doing when you use str(5+3)?