CS 101 Introduction to Computing

Lab 3[*]

August 29, 2016

Objective The objective of this exercise is to become familiar with the IPython Notebook while introducing basic mathematical operations, variable types, and printing options.

Background

Virtually all modern programming languages make us of an IDE, or Integrated Development Environment, which allows the creation, editing, testing, and saving of programs and modules. In Python, the IDE is called IDLE (like many items in the language, this is a reference to the British comedy group Monty Python, and in this case, one of its members, Eric Idle). However for simplicity we will be using ipython notebook (a browser based framework).

Here, it is worth recalling that there are three basic types of simple variables in Python: integers (whole numbers, commonly used as an index), floats (that is, numbers with a decimal point, AKA real numbers), and strings (collections of alphanumeric characters such as names, sentences, or numbers that are not manipulated mathematically such as a part number or zip code). A legal variable name must start with a letter. It is then optionally followed by some collection of letters, numerals and the underscore. It cannot contain any other characters or spaces, and cannot be a reserved word (i.e., a word with a special meaning in the language such as a command or operator). In Python, variables may be created by simply declaring them and assigning a value to them. Examples include:

```
a=2.3
name="Joe"
```

It is best to think of the equal sign as "gets". That is, think of the first example as "the variable a gets the floating point value 2.3" and the second as "the variable name gets the string Joe". An assignment command such as these literally reserves space in the computer's memory for the variable and tags it with the variable name. Then, it stores the appropriate value at that location for future use.

Open notebook by selecting typing "ipython notebook" in your terminal. A simple browser window will open. Now click on new notebook and a new notebook will be created.

This window serves two functions. First, it can serve as a sort of scratch pad to try snippets of code (shown in the steps below). Second, it can serve as the text output window for larger programs. We shall use this window to create a few variables and perform some basic manipulations on them. Type the following and then hit the Ctrl+Enter key (or click play button on the top):

```
a=5
b=13
x=5.0
y=13.0
m="Mary"
n="Nancy"
```

It is very important that the ".0" portions be included. This is how integers and floats are distinguished: floats always have a decimal point, integers don't. Also, it is possible to define the strings using the apostrophe' versus the quote ". This can be handy if you need to have a string that includes a quote or apostrophe within it; merely define the string with the other character. In any case, the computer's memory now looks something like this.

```
name  value
a         5
b         13
x         5.0
y         13.0
m         Mary
n         Nancy
```

The trick now, of course, is to access these values, manipulate them, and see the results. An important command for this process is the print command. print will print what follows it, either variables or expressions, on to the output

[*]Errors and Omissions expected.

window. Note that like all built-in commands and functions in Python, this command is all lower case. Capitalizing it will generate an error. Also, note that commands will be color coded orange-red.

At the prompt, type the following and press Ctrl+Enter or click play button on the top:

```
print( a )
```

The output should be the integer 5

Now type:

```
print( a, x, m, n )
```

In this case, the following sequence should result:

```
5 5.0 Mary Nancy
```

Continue with the following expression:

```
print( a + b )
```

This results in the value 18. This line retrieves the values of a and b from memory, adds them together, and prints the result on the output window. Neither a nor b are altered in the process. Alternately, we could have created a brand new variable and printed it. The result will be the same. Enter the following two lines to verify this:

```
c = a + b
print( c )
```

The only difference is that this version adds a new "slot" called c to the memory map above. It is worth noting that once a variable is created, its value may be recomputed over and over if desired. For example, type the following:

```
c = 20 + a
print( c )
```

The result should be 25. The first line computes a new value which then overwrites the prior value of 18.

Besides addition, the other main math operators are -, * (multiplication), / (division), ** (exponents, which can also be performed using the function pow(x,y) for xy), % (modulo), and // (floor divide). Parentheses () may be used to force the execution of some operations before others. Parentheses have the highest precedence and are followed by multiplication, division, addition and subtraction. That is, the expression a=b+c*d will multiply c by d before b is added. To force the addition first, use parentheses: a=(b+c)*d Remember, think of the equal sign as "gets" as in "a gets the value computed by…". It is an assignment, not a true mathematical relation. That is, if at some point in the future the value of b was to change, a will not automatically be altered to reflect that change. This allows you to do the following:

```
c = c + 1
```

Type this in. What do you think the result will be?

The line above may appear a little odd. After all, how can something equal itself plus one? Remember, this is an assignment, not a mathematical relation. What it says is, "Retrieve the current value of c, add one to it, and store the result back in c (overwriting the original value). Print out the value of c. You should be get 26 (the prior value of 25 plus one).

Continuing with the other math operators, type:

```
print( y/x )
```

The result should be 2.6. Now try the following:

```
print( b/a )
```

The result is also 2.6 even though both variables are integers (an integer, of course, can't contain a fractional portion). In essence, Python promotes the variables to floats in order to maintain precision, producing a floating point answer. Now try:

```
print( b/x )
```

In this case the answer is again 2.6. This is because in a mixed calculation between a float and an integer, the integer is again promoted to a float in the calculation in order to maintain the precision of the floating point variable. You can force a variable to be promoted (or demoted) by using the float() and int() functions.

Now try:

```
print( b%a )
```

The result should be 3. The modulo operator produces the remainder of the divide, that is, a goes into b two whole times with 3 left over. Finally, we have floor divide:

```
print( 18.2//4.1 )
```

The result should be 4.0. You can think of floor divide as like integer divide for floats. That is, 4.1 goes into 18.2 4.0 times (with 1.8 left over, which you can verify with

```
print( 18.2%4.1 ).
```

What do you expect the results to be from the following?

```
print( y//x )
print( y%x )
```

Type in the above lines and see if you were correct.

At times it is useful to limit the number of digits that are printed. By default, Python uses up to 18 digits if required. Try this:

```
print( x/y )
```

The result is 0.384615384615384615384 To limit this to fewer digits, the round() function may be used. The first argument is the value of expression to be rounded and the second is the number of digits after the decimal point. Now enter this:

```
print( round(x/y,3) )
```

The result should be 0.385 (rounding to the third digit). There are also limited operations on strings. The + and * operators when used with strings perform concatenation and iteration. That is, combining strings and repeating strings. Type:

```
print( m, n )
```

You should see:

Mary Nancy

Now try:

```
print( m+n )
```

The result should be:

MaryNancy

Note the lack of a separating space. Now type:

```
print( m*3 )
```

The result should be:

MaryMaryMary

The string is repeated three times. Note that it doesn't make sense to ask for things like m*2.6 or m-n. We can't have a fractional copy of something and what would it mean to subtract "Nancy" from "Mary"? These sorts of statements will generate errors.

Question 1 Evaluate the following numerical expressions in your head, and write the answers:

```
5 ** 2
9 * 5
15 / 12
12 / 15
15 // 12
12 // 15
5 % 2
9 % 5
15 % 12
12 % 15
6 % 6
0 % 7
```

Question 2 What is the order of the arithmetic operations in the following expression. Evaluate the expression by hand and then check your work.

```
2 + (3 - 1) * 10 / 5 * (2 + 3)
```

Question 3 Many people keep time using a 24 hour clock (11 is 11am and 23 is 11pm, 0 is midnight). If it is currently 13 and you set your alarm to go off in 50 hours, it will be 15 (3pm). Write a Python program to solve the general version of the above problem. Ask the user for the time now (in hours), and then ask for the number of hours to wait for the alarm. Your program should output what the time will be on the clock when the alarm goes off.

Question 4 It is possible to name the days 0 through 6 where day 0 is Sunday and day 6 is Saturday. If you go on a wonderful holiday leaving on day number 3 (a Wednesday) and you return home after 10 nights. Write a general version of the program which asks for the starting day number, and the length of your stay, and it will tell you the number of day of the week you will return on.

Question 5 Take the sentence: "All work and no play makes Ahmad a dull boy". Store each word in a separate variable, then print out the sentence on one line using *print*.

Question 6 The formula for computing the final amount if one is earning compound interest is given on Wikipedia as:

$$A = A_0 \left(1 + \frac{r}{n}\right)^{nt}$$

Where,

A_0	=	Principal Amount (Initial Investment)
r	=	annual nominal interest rate (as a decimal)
n	=	number of times interest compounded per year
t	=	number of years

Write a program that assigns the principal amount of 10000 to variable A_0, assign to n the value 12, and assign to r the interest rate of 8% (0.08). Then have the program prompt the user for the number of years, t, that the money will be compounded for. Calculate and print the final amount after t years.

Question 7 Write a program that will compute the area of a rectangle. Prompt the user to enter the width and height of the rectangle. Print a nice message with the answer.

Question 8 Write a program that will compute the area of a circle. Prompt the user to enter the radius and print a nice message back to the user with the answer.

Question 9 Write a program that will compute MPL for a car. Prompt the user to enter the number of miles driven and the number of liters used. Print a nice message with the answer.