**Doing Linear Algebra in Sage – Part 1 – Getting Oriented**

Sage is a mathematical programmable platform for doing many calculations, including ones for linear algebra.

To access Sage **use Firefox.**

Go to **sagenb.org** and create an account for yourself.

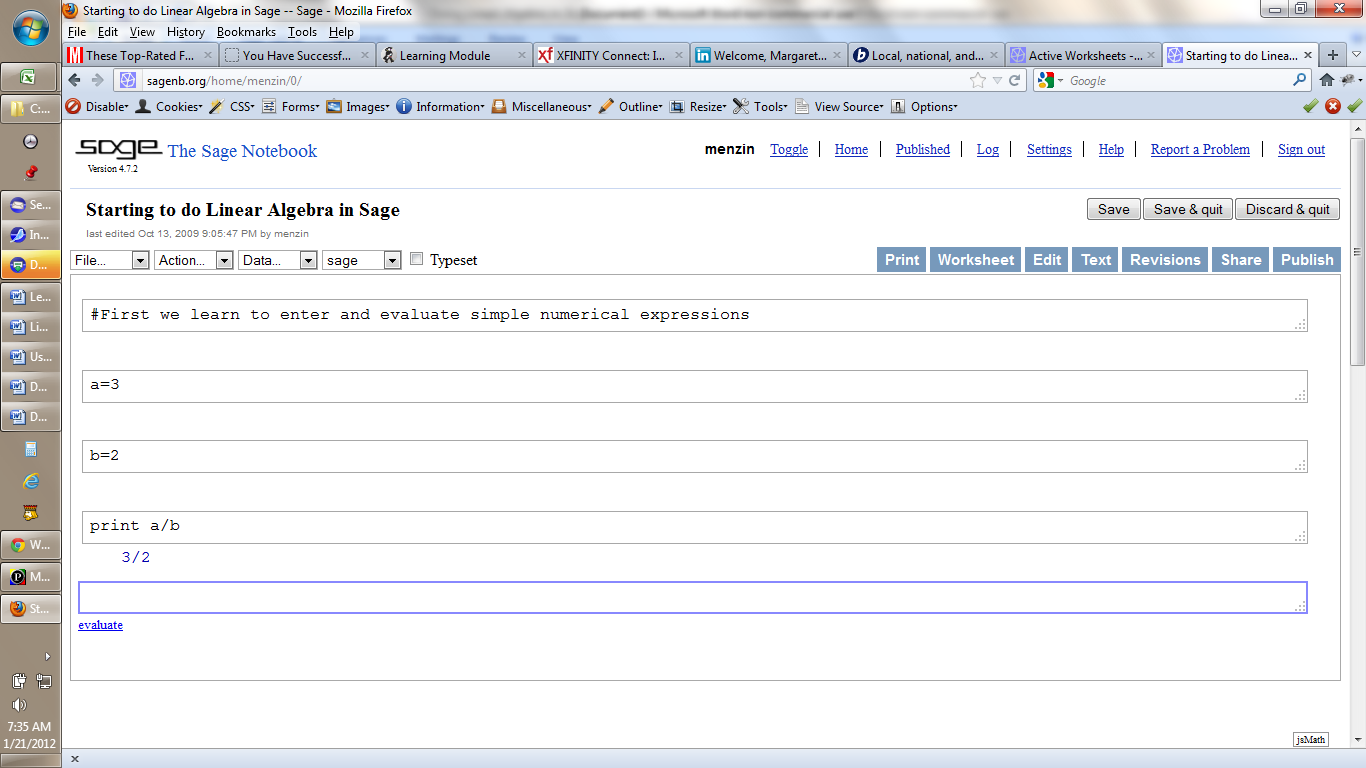
Click on New Worksheet and you will get a blank page with an outlined box.

**All your statements and expressions are entered in the blank box.**

**Below the box is the word evaluate. Clicking on evaluate will do just that.**

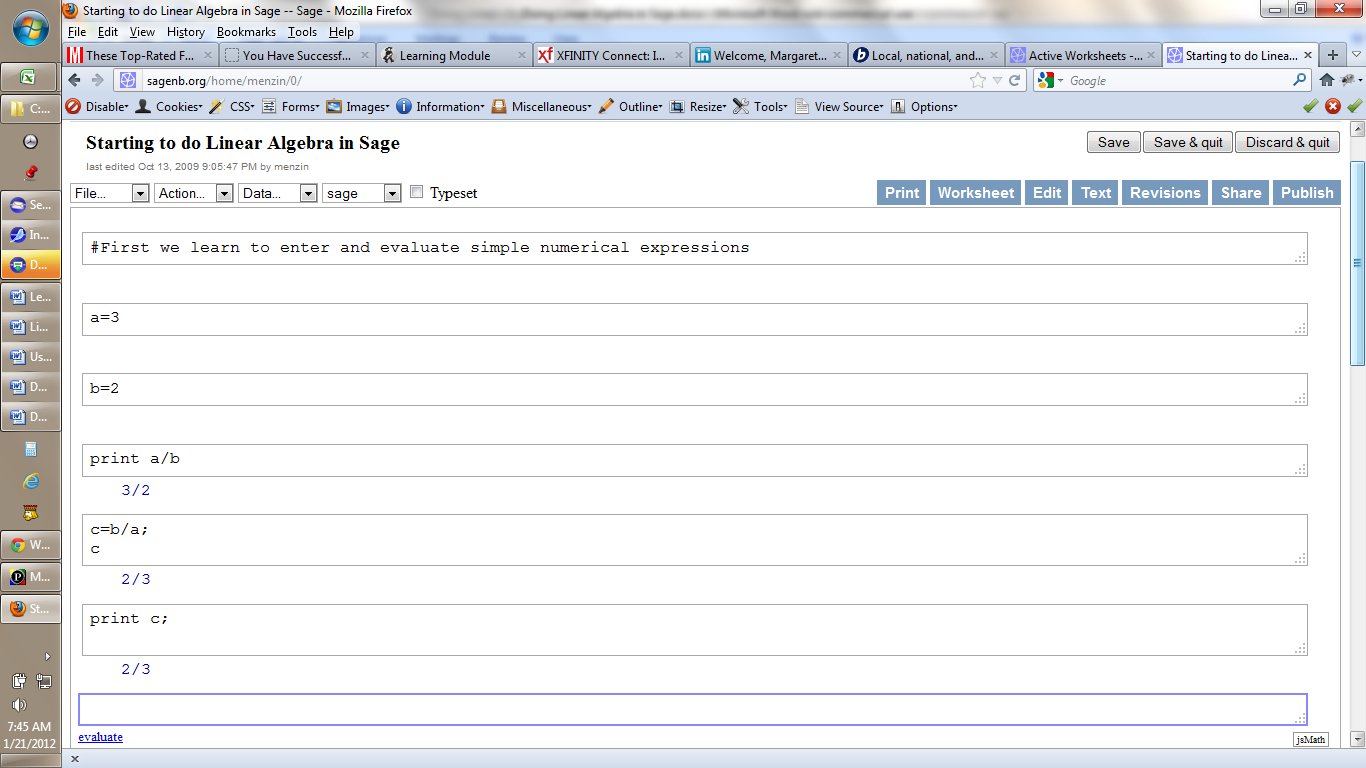
**Sage uses # to signify a** comment (which is not evaluated but will help you when you come back to your worksheet months later.)

Here is my first few statements, as I work in Sage:



Notice that **print** will cause Sage to evaluate the expression a/b and also notice that Sage gave me the value 3/2. This is because Sage works **symbolically** – that is it keeps rational expressions as rationals (rather than evaluating to 1.5--- a process which for some rational expressions would lead to potential rounding errors) and it also carries along variables such as x in that form.

I could enter multiple statements in a Sage box, as long a sI separate them with semicolons:



In the same box I have defined c as b/a, and then asked for c. When I click on evaluate, Sage responds with c. Likewise if I ask Sage to print c I must still ask Sage to evaluate.

**Please notice that Sage uses = for assignment!**

Sage is a programming package written in **python.** If you have had CS112 then you can write programs in Sage using the built in modules (for example, those for doing matrix calculations) and python.

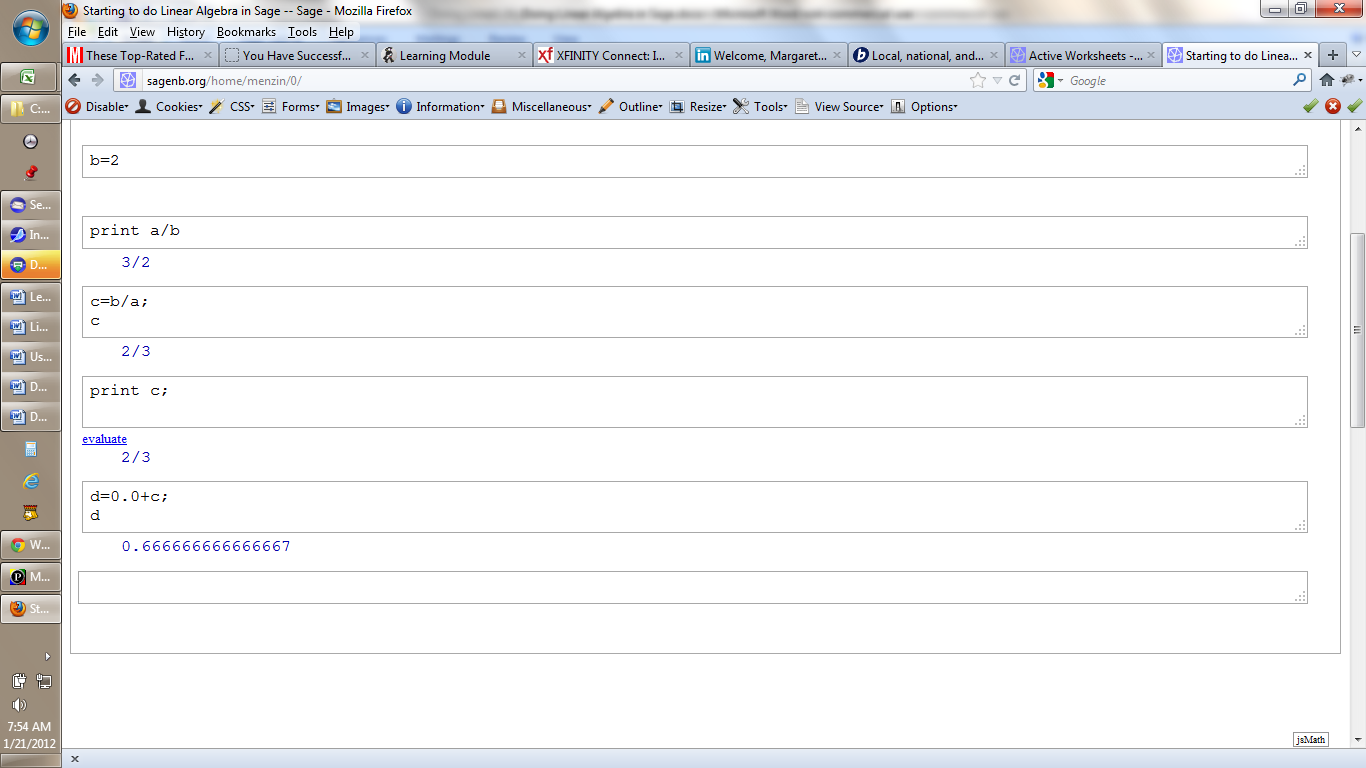
**As such Sage uses the double equal sign == to test for equality.**

That is the statement c=a + b assigns to the new variable c the value of the expression a+b.

This is NOT an equation. This is an ASSIGNMENT STATEMENT.

NOTE: On-line we are working in Notebook form. If you wish you may choose to download Sage and then work in either Notebook or Command form. Lots of information about Sage is available from my home page.

So far we have used integers, and Sage has responded with integers and rationals. If we wish to work with real numbers and to force evaluation we may do so. One way to do so is to add the value 0.0 to your number c. Sage understands this to mean that you wish to work with real numbers.



**Final reminders:**

**Sagenb on-line**

Go to sagenb.com or sagenb.org and register for an account.

Log in to your account and have fun!

When a particular worksheet is open you have the option to **publish** it (let others see it) or **share** it (let others edit it.)

Clicking on **text** will get a copy of your worksheet suitable for printing (which may be **edited)**, and **use** will get you back to the interactive version.

WARNING: When you open a previously saved worksheet you need to ask Sage to re-evaluate all the expressions by going to the **Action** drop-down menu and choosing **Evaluate all.**

**#** begins a comment (to the end of the line.)

When you cause a bunch of expressions to be evaluated, only the last one will be displayed. Use **print** if you want other one(s) to be displayed. (Sage-for-newbies section 3.4.1)

Actually you may use the print statement to format your results – e.g. ask for a certain number of digits.

**Learning more about Sage**

**Documentation** is through sagemath.org

* The Tutorial is available both at the home page and on the Documentation page.
* If you are going to work with Sage I suggest you download the Tutorial, the Constructions and the Reference Manual. (Actually I also downloaded both the long and short version of Sage for Newbies.)
* To download these as PDFs, you need to click on PDF itself – otherwise you get the interactive version (and some symbols don’t print display properly on-line.).
* When you are in the Reference Manual in the upper right-hand corner is a box with an “m”. This takes you to a list of all the modules.  
    
  Each module describes (quite clearly) the functions in the module, their limits, methods of approximation where appropriate, etc.
* Similarly, the box with an “i” in the upper right corner of the Reference Manual takes you to the index.  
    
  Finding the source code for a module is a little trickier.   
     
  First, you need to know the name of the module. Then you need to log on to your account (see Sagenb below)>  
    
  Suppose you want the code for the matrix module. It’s name is matrix, and since Sage is written in python the file is matrix.py
* Login at sagenb.com
* Start a new worksheet
* In an empty cell type  
    
   **search\_src(“matrix.py”) and shift+enter  
    
  I get the URL https://www.sagenb.org/src/matrix/matrix2.pyx/**
* Since this is a large file you will probably need to ^f to find your term.
* You will also get a ‘Browse Directory’ link with further files relevant to the module you are examining.
* **Alternately a listing of all the modules for which source code is available is at**<https://www.sagenb.org/src/modules/> and <https://www.sagenb.org/src/>   
     
  Then <https://www.sagenb.org/src/matrix/> etc hold the sub-modules associated with the matrix module.

Some modules are really sub-modules of higher level modules and need to be accessed (as far as I can tell) thru the Browse Directory link of the higher level module.

* Documentation may be found (theoretically) by typing (in an empty cell)  
    
   **search\_doc(“matrix.py”) and shift+enter**This, however, provides a search of the reference manual and is not always very helpful----for example it does not have a list of all matrix methods.
* As all method definitions begin with def it time hangs heavy on your hands you can get a hold of matrix.py with the search\_src function as above and then ^f your way down thru all the def’s.