Getting started with Sage

In this short introduction we’ll start learning to use the software package Sage. This is freely available software designed for doing all kinds of mathematical computations. To understand some of the history, you need to know that Python is a programming language that was first developed in the early 1990s. It is a very nice language that works particularly well for people with mathematical inclinations. The Sage software was built on top of Python, although at this point some of the grammar has become slightly different. You can download Sage and install it on your personal computer, but it is also available to use online via the www.cocalc.com link. This introduction will assume you are using Sage via CoCalc.

Warning: The CoCalc site doesn’t seem to work well with some versions of the Safari browser. It seems to work fine with Google Chrome and Firefox.

1. To get started with Sage, perform the following steps:
   (a) Create an account on www.cocalc.com. This should be free; don’t select any of their optional upgrades.
   (b) Create a project named “MATH205”. Then click on this project to begin.
   (c) Click the “⊕Create” tab, type in a filename like “Sage_trial”, and then select “Sage Worksheet” for the Type.
   (d) After a few moments of loading you should see a blank window with a blinking cursor on the top line. Type “5 \times 5” and then press the key combination “<Shift-Return>”. You should see the output 25 appear after a moment (the very first command sometimes takes a little while).
      Note that “<Shift-Return>” acts as the “Execute” command in Sage. If you only press “<Return>” without the Shift then nothing happens except that Sage takes you to the next line on the window.
   (e) Sage uses the standard math operators +, −, *, / and ^ (for exponentiation). It also uses parentheses in the expected way, so that you can type 600\^{(35 \times 74)}. Try using Sage to find 2017 − 919, 1.035^{10}, \sin(13.2), 2^{(2^{10})}, and 625/999.
   (f) You might have noticed that 625/999 didn’t produce the most useful answer. Try both 625.0/999 and N(625/999). The N(−) command outputs the decimal approximation to a number. Try the command “N(pi,digits=100)”.
   (g) Sage uses “==” as a question: are the two quantities equal or not? In contrast, “=” is a declaration: “x=[something]” means “set the value of x to be [something]”. Try the following commands (hit <Shift-Return> after each one):
      (i) 3==5
      (ii) 7==7
      (iii) x=5
      (iv) x
(v) \( x \times x \)
(vi) \( x == 5 \)
(vii) \( x \times x == 17 \)
(viii) \( 3 = 5 \)
(ix) \( x = 10 \)
(x) \( x \)

Note that the command “3=5” should produce an error.

(h) In Sage, a “list” is an ordered collection of objects inside of square brackets: for example, [2, 7, 3.1]. Try the following commands (hit <Shift-Return> after each one):

(i) \( l=[2,3,4] \)
(ii) \( m=[5,4,7,1] \)
(iii) \( l[0] \)
(iv) \( m[2] \)
(v) \( l+m \)
(vi) \( m+l \)
(vii) \( l.append(6) \)
(viii) \( print l \)
(ix) \( len(l) \)
(x) \( l==m \)

(i) Sage can plot functions. Try each of the following commands:

(i) \( plot(x^2,(-10,10)) \)
(ii) \( plot(x^2,(0,10)) \)
(iii) \( plot(x^2,(-10,10),aspect\_ratio=1) \)
(iv) \( plot(x^2,(-10,10),ymin=-50,ymax=200) \)
(v) \( plot(e^x,(-10,10)) \)
(vi) \( plot(e^x,(-10,10),aspect\_ratio=1) \)

(j) Sage can also plot implicit functions. If you have a function of two variables \( f(x, y) \) then it will plot solutions to \( f(x, y) = 0 \). Try

\[
\text{implicit\_plot}(x^2+y^2-1,(x,-2,2),(y,-2,2))
\]

Most likely this will give you an error message, because the variable \( y \) has not been defined in Sage yet (it starts out with the variable \( x \) predefined). Type

\( y=var('y') \)

and then go back and execute the implicit\_plot function again (you don’t have to type the whole command over, just go back to the previous line and execute it one more time). It should work this time.

Also try
(k) Here is one way to display several graphs at the same time, using different colors. Try this:

```python
P1=plot(x^2,(-10,10),color='red')
P2=plot(sin(x),(-10,10),color='blue')
show(P1+P2)
show(P1+P2,ymin=-5,ymax=5)
```

[Note: Sage knows pretty much any color you can think of. You can try purple, violet, tan, salmon, chartreuse, indigo, hotpink, darkblue and so on.]

(l) Finally, we can use Sage to plot lots of points. Try the commands

```python
plist=[(cos(x),sin(x)) for x in [-3,-2.9,..,3]]
plist
show(points(plist))
show(points(plist),aspect_ratio=1)
```

Note that Sage uses the .. notation in the list to say “continue the evident arithmetic sequence”. Try changing the 2.9 to 2.8 or 2.99 and observe what happens (you might want to skip the command that prints out the whole list of points).

(m) Sage can numerically find the roots of a function. Try the following commands:

```python
f(x)=4*x^3+5*x^2-7*x+2
plot(f(x),(-10,10),ymin=-100,ymax=100)
```

You see that $f$ has two roots, one between $-5$ and $0$ and the other between $0$ and $5$. To find them, use

```python
find_root(f(x),-5,0)
find_root(f(x),0,5)
```

The second command should have produced an error message. Note that since the function might have multiple roots, we have to provide an interval for Sage to look in: in the first command we gave it the interval $(-5,0)$, and in the second command we gave it $(0,5)$. To see why we got the error, try

```python
plot(f(x),-5,5,ymin=-5,ymax=10)
```

(n) Consider the function $A(t) = 1000te^{0.02t}$. If we want to solve $A(t) = 12000$ numerically using Sage, we have to convert it into the equation $1000te^{0.02t} - 12000 = 0$ and solve it like this:

```python
t=var('t')
A(t)=1000*t*e^(0.02*t)
find_root(A(t)-12000,0.5)
```

You should have gotten an error message again. Play around with the specified interval (drawing the graph of $A(t)$ if need be) to find the solution to $A(t) = 12000$. 

(o) Try the command “find_root(sin(x),1,50)”. The result shows why you should be careful with this command. How could the computer’s output be misinterpreted?

(p) Two final remarks about getting started with Sage. Sometimes you will make a mistake and give the computer an unsolvable problem, so that it sits there computing forever. There is a “Stop” button in Sage’s toolbar for these moments. Also, you will often need to print out the results of your Sage session (to hand in with your HW, for example). There are two ways to do this. The quickest way is to use the “Adobe” button in the toolbar, which will have Sage take the notebook contents and turn them into a PDF. Sometimes this is a little slow, but it basically works okay. Another option is to use the “Print” button, which saves the notebook as an html file. You can then open the html file in a browser and print it as you usually would.

Note that you can “hide” the output of functions, by clicking on the downarrow next to the line of output. This is useful when you have extraneous stuff in your Sage worksheet that is not relevant to the assignment and doesn’t need to be looked at (for example, if you made a mistake somewhere and got an error, or if you accidentally asked Sage to output something very large).