Sage is a free, open-source, mathematics software that can be used as an alternative to Magma, Maple, Mathematica, and Matlab. The software is based on the popular scripting language Python. We will use Sage to tackle the computational aspects of Math 373: Elliptic Curves, so it will be advantageous to spend a reasonable amount of time becoming comfortable with this tool. The following tutorial is meant to introduce you to Sage and get you started on your first assignment. If you are a more experienced programmer and find this tutorial a little slow, feel free to jump right into the tutorials available on the website (http://sagemath.org/doc/tutorial/index.html). They are most excellent (and I am merely borrowing bits and pieces of them here).

Go to http://www.sagemath.org/ and click on Try Sage Online. You will be prompted to sign into the Sage Notebook v5.4 by selecting the OpenID provider of your choice (e.g. if you have a gmail account, you would click on the big Google icon). You will then be prompted to create a Sage user name and click Create profile. Now you are ready to start using Sage.

Click on New Worksheet. You will be prompted to enter a name for this worksheet (e.g. My First Worksheet). Click inside the box above the evaluate button. A cursor should appear and the box will become highlighted in blue. First try typing:

1+1

in the box and then click on evaluate. Sage should return

2

in blue right below it. In Sage (and Python), any text typed after a pound sign, #, will be invisible to the compiler. This is the comment character. For example, let us try the same thing again, including a comment before the command:

# Here, we sum one and one. Ha-ha I can type anything I want.
1+1

After clicking evaluate you should get the same output you got before. We can also have text that actually gets displayed. Try typing:

"Hello World"

with quotations around the text and click evaluate. We might also assign this text to a variable name so that we can use it later. For example try typing:

message ="Hello World"

and click evaluate. To print the message type:

print message

and click evaluate. Now let us suppose we want Sage to understand the function \( f(x) = 1 - \sin^2(x) \). We are not even sure if Sage recognizes such functions as sin. Try typing:

sin?

and then click on evaluate. Notice that Sage does indeed recognize the function sin and tells us what it knows about it and provides some examples for its usage. In general, we can type in any
function or constant followed by a question mark, and Sage will return any built-in documentation it has on the object. Now try typing and evaluating each of the following statements in succession and see what you get:

\[ f = 1 - \sin(x)^2 \]
print f
print maxima(f)
f.simplify_trig()
f(x=\pi/2)
f(x=\pi/3)
integrate(f, x).simplify_trig()
print maxima(integrate(f, x).simplify_trig())
f.differentiate(2).substitute({x: 3/\pi})
print maxima(f.differentiate(2).substitute({x: 3/\pi}))

Enough with the calculus, let us try to make a list and a matrix. In Sage a list is like a vector that stores elements of arbitrary type. The elements in the list are indexed starting from zero. For example, evaluate:

\[ v=[2,3,\text{range}(4,7), \mathbb{Q}[x]] \]
v
We can call a particular list element by evaluating commands like the following:

\[ v[0] \]
or
\[ v[2] \]

To make a matrix, we can do the following:

\[ \text{matrix}([[1,2], [3,4]]) \]

Or perhaps we want the \(3 \times 3\) identity matrix:

\[ \text{matrix}([[1,0,0], [0,1,0], [0,0,1]]) \]

We can call elements of a matrix, similar to how we would in a list:

\[ M=\text{matrix}([[1,42], [3,4]]) \]
\[ M[0,1] \]

Sage should return 42, the element in the 0th row and the 1st column of the matrix, \(M\).

Now for some maths relevant to elliptic curves. Page 2 of your text claims that given a rational solution \((x, y)\) to the Diophantine equation

\[ y^2 - x^3 = c, \]

we can use Bachet’s duplication formula,

\[ \left( \frac{x^4 - 8cx}{4y^2}, \frac{-x^6 - 20cx^3 + 8c^2}{8y^3} \right), \]

to find infinitely many rational solutions to the equation provided that \(xy \neq 0\) and \(c \neq 1, -432\). Let us try to reproduce the example in the book by choosing \(c = -2\). First, we need to tell sage that \(x\) and \(y\) are variables over the rational field by typing and evaluating:

\[ R, (x, y) = \text{PolynomialRing}(\mathbb{Q}, 2, \text{‘xy’}).\text{objgens()} \]
Then we should check that $(3, 5)$ is a solution by typing and evaluating the following:

```python
f = y^2 - x^3 + 2
f(3, 5)
```

Sage should return ‘0’ to this series of commands, so we see that indeed, $(3, 5)$ is a solution! Next, let us define the duplication formula for Sage. We will call the “$x$-coordinate” of the formula $g$ and the “$y$-coordinate” of the formula $h$:

```
g = (x^4 - 8*(-2)*x)/(4*y^2)
h = (-x^6 - 20*(-2)*x^3 + 8*(-2)^2)/(8*y^3)
```

Then we can plug in the point $(3, 5)$:

```
g(3, 5)
h(3, 5)
```

What is the output? It should be similar to the point listed on page 2. I get $(129/100, 383/1000)$. Why is there a sign difference? Does that sign difference matter? Is this point a rational solution to our equation? Let us check by typing and evaluating the following command:

```
f(129/100, 383/1000)
```

We can repeat this process again using our new rational solution:

```
g(129/100, 383/1000)
h(129/100, 383/1000)
```

Finally, we check that the point we obtain $(2340922881/58675600, 113259286337279/449455096000)$ is a rational solution to our equation by typing and evaluating:

```
f(2340922881/58675600, 113259286337279/449455096000)
```

Indeed we should find that Sage returns 0 and this rational point is, in fact, a solution. What is another rational solution to the Diophantine equation

$$y^2 - x^3 = -2?$$

What if we were asked for 20 rational solutions? Personally, I do not care to waste my time copying and pasting very large numbers. So we should use a slightly more sophisticated approach and try to develop some code to do perform this process for us, and print out the desired number of rational solutions. First we need to set all our inputs, we can type these and evaluate them all together:

```
N = 3  # Number of solutions desired
x_new = 3  # Set the initial ‘x’ value
y_new = 5  # Set the initial ‘y’ value
```

Then we create a “for loop” to tell generate the desired solutions. Please be aware that indentation is VERY important in Sage (and Python). Sage worksheets will automatically indent for you after you press enter when needed. Try to evaluate and understand the following program:

```
for i in range(N):
    x_old = x_new
    y_old = y_new
    x_new = g(x_old, y_old)  # The new x-value from the duplication formula
    y_new = h(x_old, y_old)  # The new y-value from the duplication formula
    print (x_new, y_new)  # Prints the point from the duplication formula
    print ("Is this a solution?")
    if f(x_new, y_new) == 0:  # Checks if new point is a solution to the eqn
        print ("Yes!")
```
Try changing ‘N’ from 3 to 5, and re-evaluate. You should get some VERY LARGE numbers that would be a nightmare to copy and paste by hand! Finally, we can plot the graph of the elliptic curve by typing and evaluating:

```
implicit_plot(y^2 == x^3-2,(x,0,3),(y,-3,3))
```

If we want to plot two equations at the same time, we might do the following:

```
p1 = implicit_plot(y^2==x^3-2,(x,-3,3), (y,-3,3),color= 'red')
p2 = implicit_plot(y^2==x^3+2,(x,-3,3), (y,-3,3),color= 'blue')
show(p1+p2)
```

As output, we should obtain the following pictures:

![Elliptic Curve Plots](image)

That concludes this mini-tutorial. To gain more comfort with Sage, I highly recommend going through some of the sections of A Guided Tour which can be found here: [http://sagemath.org/doc/tutorial/tour.html](http://sagemath.org/doc/tutorial/tour.html). In particular, the sections:

- Assignment, Equality, and Arithmetic
- Getting Help
- Functions, Indentation, and Counting
- Solving Equations
- Some Common Issues with Functions
- Loops, Functions, Control Statements, and Comparisons
- Lists, Tuples, and Sequences

If you have taken linear algebra and abstract algebra I and II, some of the following sections will also be understandable and extremely useful:

- Basic Rings
- Linear Algebra
- Polynomials
- Elliptic Curves

As you read through various sections, try to reproduce the examples presented in a Sage worksheet. Try changing aspects of various examples and see what happens. In my experience, this is the best way to learn a new software or programming language.

There are also some tutorials directed specifically developed specifically for undergraduates that can be found here: [http://www.sagemath.org/doc/prep/index.html](http://www.sagemath.org/doc/prep/index.html)
As the course progresses, you may find that running your computations online using the shared Sage servers is too slow. If this proves to be the case, you may wish to download Sage onto your own computer. Go to \url{http://www.sagemath.org/} and click on Download 5.5. From here you should follow the directions on the website to download Sage onto your own machine.

Alternatively, Sage is already installed on the two machines in the \textit{Undergraduate Computer Lab}. This lab is located in the basement of Herman Brown Hall in room 34. Your student ID should give you access to this room. After you log on, double click on the VMBox icon on the desktop. Once Virtual Machine has started up (and this could take a while), click on Sage. Next you will need to \textbf{WAIT} for sage to start up. It took like 5 minutes when I tried it the other day. Go walk around and get a snack to save your sanity. When it finishes loading, you should see a Sage worksheet that looks just like the online interface.