Damian Sclafani

Lab 01 - New User's Tour

To take the tour, click in each executable cell, the ones with $\text{In }[\,]$: beside them, type the appropriate code, and execute the cell. You execute a cell by pressing $\textbf{Shift} + \textbf{Enter}$. Note that simply pressing $\textbf{Enter}$ will go to the next line in the executable cell and will not execute the code.

SageMath is a free, open-source math software that supports research and teaching in algebra, geometry, number theory, cryptography, numerical computation, and related areas. SageMath is based on the programming language Python. We will use SageMath to perform calculations and give insight into the many topics covered in Calculus.

SageMath as a Calculator ¶

SageMath's most basic function is to act as a conventional calculator. For example, we can use SageMath to simplify the expression

$$\frac{3(2+\frac{1}{2})^2}{\frac{1}{5}-\frac{2}{3}}$$

We can simply input the expression into an executable cell exactly as we would a calculator. One thing to be careful about in SageMath is you must use $*$ to represent multiplication.

```
In [1]: (3*(2+1/2)^2)/(1/5 - 2/3)
```

```
Out[1]: -1125/28
```

By default, SageMath does not display the output in a "nice" way. You can make SageMath display the output in a more convenient manner by using the $\textbf{show}(\ldots)$ command.

```
In [3]: show((3*(2+1/2)^2)/(1/5 - 2/3))
```

$$-\frac{1125}{28}$$

Since the expression only contains rational numbers, SageMath gives the output as a rational number. If we wanted a decimal approximation of the output, we could use the $\textbf{round(exp, n)}$ command, where n is the number of decimal places we want displayed.

```
In [4]: round(-1125/28, 10)
```

```
Out[4]: -40.1785714286
```

A second way in which we could get a decimal approximation for a rational expression is to make at least one of the numbers a floating-point number. This means, make one of the numbers a decimal.

```
In [5]: (3.0*(2+1/2)^2)/(1/5 - 2/3)
Out[5]: -40.1785714285714
```

Predefined Functions and Constants

SageMath has many of the common math functions and constants predefined. Most of these can be referenced by their normal notation. Note how SageMath has no issue calculating $\sqrt{100}$, $\sqrt[3]{54}$, $\cos(\pi)$ and $\arcsin\left(\frac{1}{2}\right)$.

```
In [6]: sqrt(100)
Out[6]: 10
```

```
In [7]: (54)^(1/3)
Out[7]: 3*2^(1/3)
```

```
In [8]: cos(pi)
Out[8]: -1
```

```
In [9]: arcsin(1/2)
Out[9]: 1/6*pi
```

SageMath gave exact answers for $\sqrt[3]{54}$ and $\arcsin\left(\frac{1}{2}\right)$. In the two cells below, use the $\mathbf{show}(\ldots)$ command to better display the output of these two calculations.

```
In [12]: show((54)^(1/3))
```

$$3 \cdot 2^{\frac{1}{3}}$$

```
In [11]: show(arcsin(1/2))
```

$$\frac{1}{6}\,\pi$$

```
In [13]: round((54)^(1/3),5)
Out[13]: 3.77976
```

```
In [14]: round(arcsin(1/2),5)

Out[14]: 0.5236
```

If you wish to add more cells to the notebook, simply click the $+$ button in the toolbar above. You can also click on $\textbf{Insert}$ in the toolbar above and choose either $\textbf{Insert Cell Above}$ or $\textbf{Inserct Cell Below}$. This will insert a cell either above or below the current selected cell. By default, this will add an executable cell. Go ahead and add two new executable cells and use these cells to approximate the values of $\sqrt[3]{54}$ and $\arcsin\left(\frac{1}{2}\right)$ to 5 decimal places.

You can also add a Markdown cell to the notebook. To do this, simply add a new cell and change the type from Code to Markdown. A Markdown cell is a cell which allows text. In addition to text, you can type in HTML and LaTeX code and the cell will compile this code into text. If you do not know HTML or LaTeX, then you can use the Markdown cells plainly as a cell for basic text. Use the $\textbf{Insert}$ button to add a Markdown cell at the very beginning of the document. In the cell, type your name, your section number, and any other information your TA would like you to include in your lab submission.

Assigning Expressions to Names

SageMath allows us to assign an expression to a name so that it can easily be referenced throughout the entire worksheet. For example, we can assign $\frac{\pi}{2}$ to the name a by doing the following.

```
In [16]: a = pi/2
```

Note that SageMath supresses the output whenever you are making an assignment. We can check that a really has been assigned to $\frac{\pi}{2}$ by entering into an input cell by itself.

```
In [17]: a

Out[17]: 1/2*pi
```

Now, whenever we use a throughout the entire worksheet, it will be referring to $\frac{\pi}{2}$. If we wanted to reassign a to something else, all we have to do is set it equal to the new value.

```
In [18]: a = pi/3
         a

Out[18]: 1/3*pi
```

Note that this time, SageMath did display the new value of a. This is because we both assigned a to its new value and told SageMath to return a in the same input cell. SageMath allows multiple lines of code in the same cell and will execute all lines, however, it will only display the last line of code.

```
In [19]: a
         a + 1
         a + 2

Out[19]: 1/3*pi + 2
```

If you wish to display mutliple lines, you can use the $\textbf{print}(\ldots)$ command.

```
In [20]: print(a)
         print(a + 1)
         print(a + 2)

         1/3*pi
         1/3*pi + 1
         1/3*pi + 2
```

$\textbf{Caution:}$ Be very careful about the names you choose for your assignment variables. SageMath will allow you to use a name that it has already defined and it does $\textbf{not}$ warn you that you are overwriting a predefined command. For example, if you do the assignment sin = 5, then the next time you use sin, it will think you mean 5 instead of the trig function. In order to recover the trig function sin, you need to reset SageMath by using the command $\textbf{reset}()$. This will clear all variables and functions you previously assigned, so you will need to re-execute those codes to reassign the variables and functions.

Creating Functions

We can also create functions in SageMath. For example, in order to create the function $f(x) = x^2$, we do the following.

```
In [21]: def f(x):
             return x^2
```

Simlilar to assignments, SageMath will not display any output when creating a function. It is good practice to call the function after creating it in order to verify that you created the intended function.

```
In [22]: def f(x):
             return x^2
         f(x)

Out[22]: x^2
```

Now that we have $f(x)$ defined, we can evaluate the function at different inputs.

```
In [23]: f(4)
Out[23]: 16
```

```
In [24]: f(x) + 4
Out[24]: x^2 + 4
```

```
In [25]: f(f(x))
Out[25]: x^4
```

What if we wanted to use the variable t in place of x?

```
In [26]: f(t)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-26-2837c479ee4f> in <module>()
----> 1 f(t)

NameError: name 't' is not defined
```

SageMath gives an error saying that t is not defined. The only variable that SageMath has predefined is x. If we want to use other variables, such as t, then we will need to make t a variable by using the $\mathbf{var}$ command.

```
In [27]: var('t')
Out[27]: t
```

Now, SageMath will have no problem evaluating $f(t)$.

```
In [28]: f(t)
Out[28]: t^2
```

Rather than just doing normal calculations, we can also have SageMath perform calculus.

```
In [29]: var('h')
         limit((f(x+h) - f(x))/h,h=0)
Out[29]: 2*x
```

```
In [30]: diff(f(x))

Out[30]: 2*x
```

```
In [31]: diff(f(x),2)

Out[31]: 2
```

```
In [32]: integrate(f(x),x)

Out[32]: 1/3*x^3
```

Plotting

The last part of SageMath which we will explore in this tour is its ability to plot graphs of functions. First, use the cell below to create the function $g(x) = \frac{\sin(x)}{x}$.

```
In [43]: def g(x):
             return sin(x)/x
```

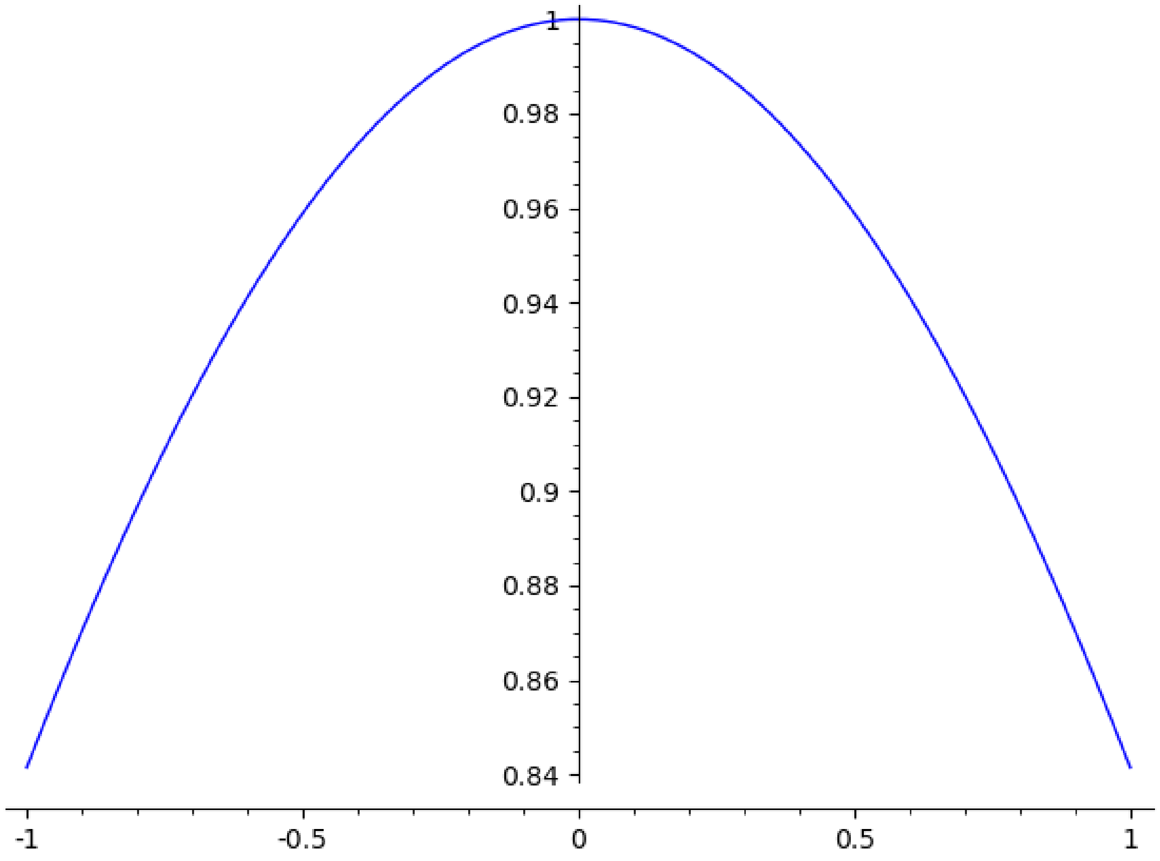
```
In [44]: g(x)

Out[44]: sin(x)/x
```

Now, plot $g(x)$ by using the function $\mathbf{plot(g(x))}$.

In [45]:
```
plot(g(x))
```

Out[45]:



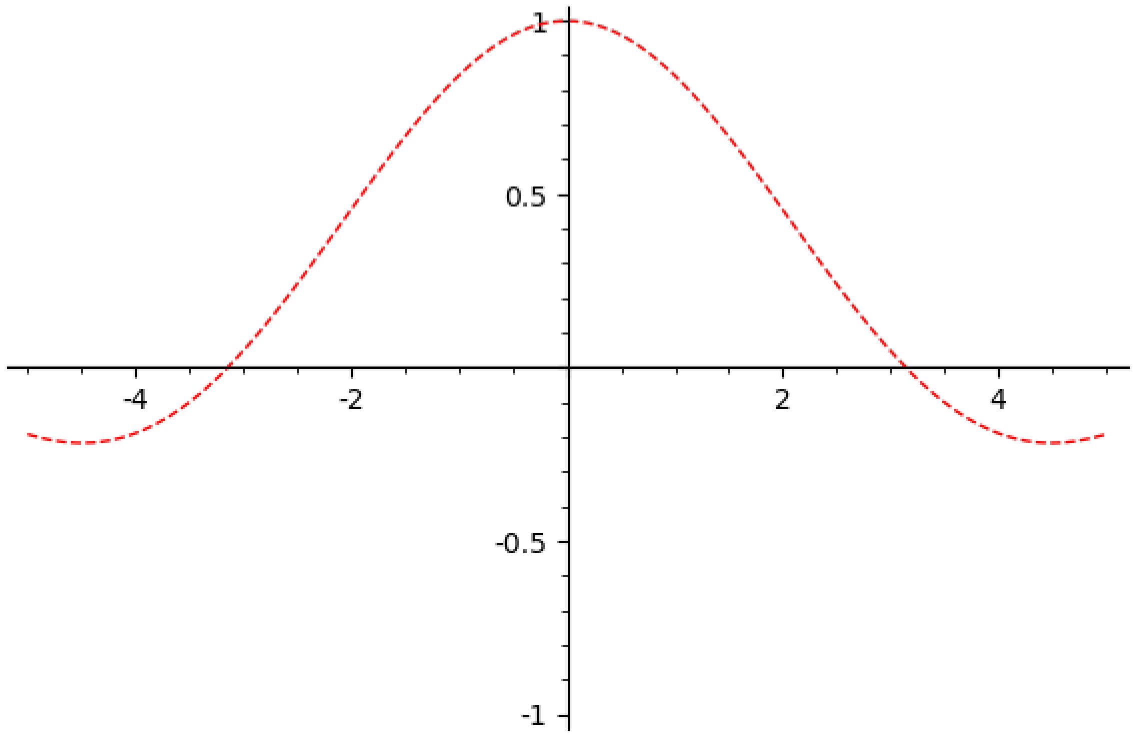
The function $\mathbf{plot()}$ can take inputs other than just $g(x)$. The other inputs allow you to change specific characteristics about the graph of the funciton such as the x-range and y-range, the color, the linestyle, etc. To see a detailed description about the plot function, or about any function, simply follow its name with $\mathbf{?}$ and execute the cell.

In [46]:
```
plot?
```

We can run the following code to display the graph of $g(x)$ in a window that goes from $[-5, 5]$ on the x-axis and $[-1, 1]$ on the y-axis, changes the graph color to red, and changes the linestyle to dashed.

In [47]:
```
plot(g(x),xmin = -5, xmax = 5, ymin = -1, ymax = 1, color = 'red', linestyle =
'dashed')
```

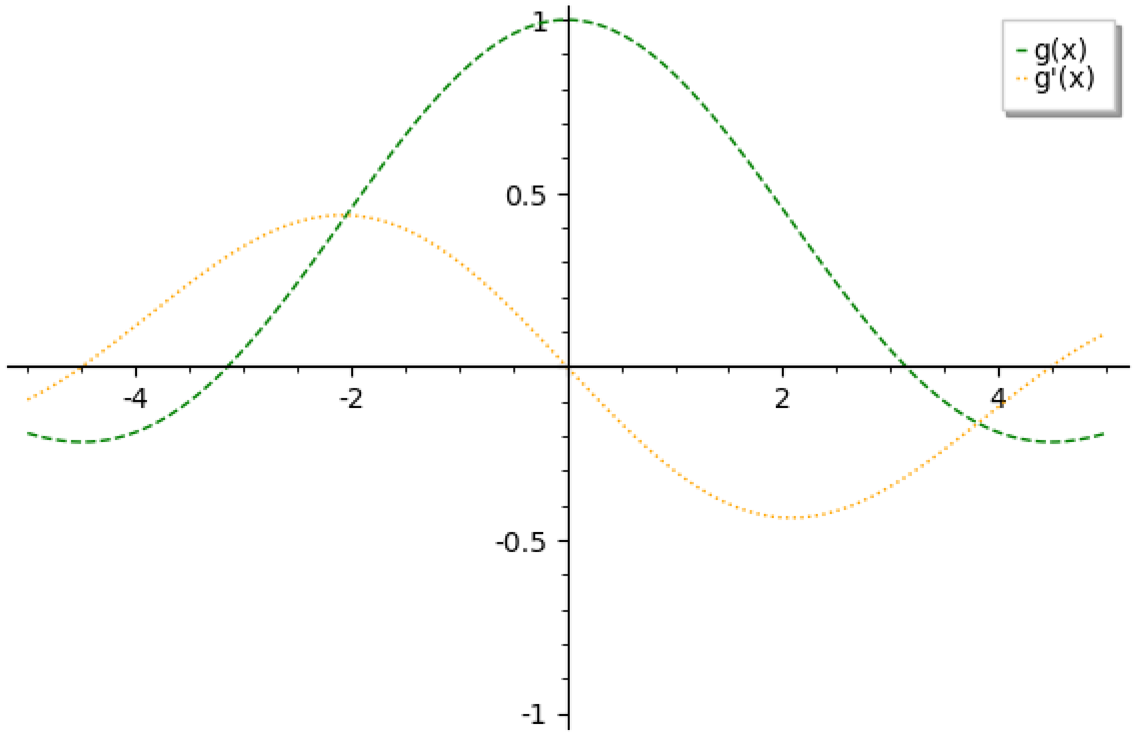
Out[47]:



SageMath also has the ability to graph multiple functions at once. When plotting multiple functions at the same time, surround them in brackets. Also, you can change properties for each function by also surrounding the properties in brackets. The following command plots both $g(x)$ and its derivative $g'(x)$ from $[-5,5]$ on the x-axis and $[-1,1]$ on the y-axis. It also plots $g(x)$ in green with dashes and $g'(x)$ in orange with dots. Additonally, we add a legend to distinguish between the two functions in the graph.

In [48]:
```
plot([g(x), diff(g(x))], xmin = -5, xmax = 5, ymin = -1, ymax = 1,
     color = ['green', 'orange'], linestyle = ['dashed', 'dotted'],
     legend_label = ['g(x)', "g'(x)"])
```

Out[48]:



Converting Notebook to PDF for Submission

Once you are done with a lab, you will submit a PDF version of the notebook to Blackboard for your TA to grade. The easiest way to convert the Notebook file to a PDF file is to choose $\mathbf{File} \rightarrow \mathbf{Print\ Preview}$ from the toolbar at the top of the page. Then, press $\mathbf{CTRL} + \mathbf{P}$ and choose to $\mathbf{Save\ as\ PDF}$. You can upload the PDF to Blackboard.

Note: These instructions may differ depending on Web Browser and Operating System.