Sage is an open source system for mathematics. The goal of this sheet is to tell you how to run sage, indicate a few resources for becoming more fluent with sage, and list a few basic commands which will be helpful for the first lab.

1 Sage Access

- You can simply download Sage from

  `http://www.sagemath.org`

  and install it on the machine(s) of your choice.

- For quick calculations, go to

  `http://sagecell.sagemath.org/`

  and enter your Sage commands in the obvious box.

- Get yourself a free acount at

  `https://cocalc.com/`

  You can set up a Sage notebook and work there.

2 Learning Sage

All sorts of documentation is available at `http://doc.sagemath.org/`. You are especially encouraged to look at the Tour of Sage, which will get you started quickly.

You might also want to look at the book Sage for Undergraduates by Gregory Bard.

3 The Basics

3.1 Calculations

You can do simple arithmetic in Sage:

```
sage: 1+1
2
sage: 2+3*5
17
```

Where possible, Sage tries to work with fractions, and symbols like $\sqrt{2}$, instead of decimal approximations:

```
sage: 2018/32
1009/16
sage: sqrt(1^2+2^2)
sqrt(5)
```

If you want to turn an answer into a decimal approximation, you need to apply the function `N`, for *numeric*.

```
sage: N(2018/32)
63.0625000000000
sage: N(sqrt(1^2+2^2))
2.23606797749979
sage: N(sqrt(1^2+2^2), digits=50)
2.2360679774997896964091736687312762354406183596115
```

Sage knows about certain standard constants and functions:

```
sage: N(e)
2.71828182845905
sage: pi, sin(pi), N(pi)
(pi, 0, 3.14159265358979)
sage: abs(-sqrt(3))
sqrt(3)
sage: N(abs(-sqrt(3)))
1.73205080756888
```

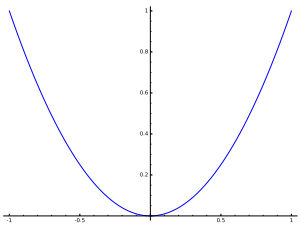
In fact, Sage is object oriented, and some functions can be applied *after* the argument. For instance, if you always compute decimal approximations like this:

```
sage: sqrt(2).n()
1.41421356237310
sage: sqrt(2).n(digits=50)
1.4142135623730950488016887242096980785696718753769
```

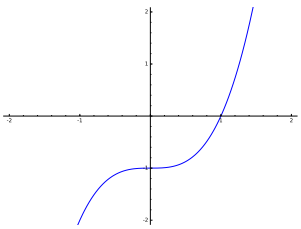
then you're free to use N and n for whatever sort of variable you like.

Sage makes it easy to plot functions:

```
sage: p = plot(x^2)
```

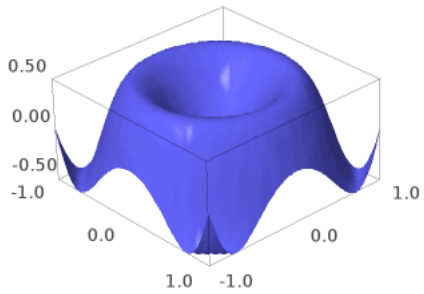


```
sage: q = plot(x^3-1,xmin=-2,xmax=2,ymin=-2,ymax=2)
```



```
sage: var('x,y')
(x, y)
```

```
sage: r = plot3d(sin(pi*(x^2+y^2))/2,(x,-1,1),(y,-1,1))
```



3.2 Other useful commands

If you want to be able to use something as a variable, you need to declare it:

```
sage: var('k')
k
```

Then you can use this as a variable in constructions like

```
sage: sum(k^2,k, 1,10)
385
sage: sum(1/k,k,1,7)
363/140
sage: N(sum(1/k,k,1,7))
2.59285714285714
```

If you'd like to learn more about, for instance, `sum`, just type:

```
sum?
```

The output of the previous command is denoted with the underscore character _. Thus:

```
sage: var('y')
y
sage: (x+y)^2-4*x*y
(x+y)^2-4*x*y
sage: expand(_)
x^2-2*x*y+y^2
sage: factor(_)
(x-y)^2
```

3.3 Loops and iterations

Sometimes, it's nice to be able to have the computer repeat commands in a loop:

```
sage: x=1
sage: while (x<10):
....:  x^2
....:  x = x+1
```

```
1
4
9
16
25
36
49
64
81
```

Please note that indentation is important in Sage

Here is a quick way to accomplish substantially the same thing:

```
sage: range(1,10)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
sage: [i^2 for i in range(1,10)]
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```