Lets get started with some math and programming!

But we will also learn some more markdown and how to make things look nice!

We are using Markdown which is great for making notes and can be used to do math notation also.

This is H1 one # gives you H1 You type # H1 at the beginning of the line

this is H2 two ## gives you H2

this is h3

you can also use html notation which you have to click on the markdown to see!

So click on this cell to see how it is done. See above how this cell is Markdown. This allows us to add notes!

To see how this is done. Go to Github. Download the python notebook. Open it. Then double clikc on this. Remember to download I right click on raw and then do save as. It seems to work smoothest in chrome.

You can add an equation

$$c = \sqrt{a^2 + b^2}$$

here is a link to a cheat sheet. https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

I can also make the link inline

To make it inline you put the text in brackets [] and then the link next to it in () click to see

Here is a link to a Latex cheat sheet.

Now we can do some math!

I am going to throw some programming tricks at you!

An Aside

I put all my notebooks on github. It is a version control program. It is like dropbox but keeps track of changes. On the class webpage are all the links to the notebooks on github.

to get the notebook is a little tricky. Some web browsers are easier than others. I like chrome. In chrome after getting to the website I right click on raw and then choose save as. You should now be able to save as an .ipynb file and have exactly what I hand out.

Do not copy and paste from the online notebook. I find people who copy and paste do not learn the muscle memory and do worse in class. I give you time each day to work through the notebooks. Type it in and work through each notebook

```
In [1]: 2+2
```
```
Out[1]: 4
```
```
In [2]: 3*4
```
```
Out[2]: 12
```
```
In [3]: 6/4
```
```
Out[3]: 1.5
```

this is New!!!! Python 3 does integer math correct!

```
In [4]: type(6)
```
```
Out[4]: int
```

but what if we have a decimal?

```
In [5]: type(6.0)
```
```
Out[5]: float
```
```
In [6]: 6.0/4
```
```
Out[6]: 1.5
```

If you wanted to use integers you use //

In [7]:
```
6.0//4
```
Out[7]:
```
1.0
```

how python treats decimals is very different between Python 2 and 3. Python 2 used to default to integers. Python 3 no longer does this. It is nice!

I realize many of you haven't thought about floats and ints. Here is an overview showing why integers take less memory and are also good for looping and counting.

In [2]:
```
#but that is if we don't have a calculator.  We can use parameters in programmin
#this will move us towards something more powerful
a=6
b=4
a/b
```
Out[2]:
```
1.5
```

In [9]:
```
b=4.  # I updated b and added a decimal place to make it a float
a/b
```
Out[9]:
```
1.5
```

In [3]:
```
a%b #$this is call mod and gives the remainder!  Remember this!
```
Out[3]:
```
2
```

In [11]:
```
#I almost forgot exponents
a**2
```
Out[11]:
```
36
```

In [12]:
```
a**0.5
```
Out[12]:
```
2.449489742783178
```

Remember computers always remember PEMDAS! Do you remember PEMDAS?

In [13]:
```
2+2**2
```
Out[13]:
```
6
```

In [14]:
```
(2+2)**2
```
Out[14]:
```
16
```

In [15]:
```
c=(2+2)**2
print (c)
```
```
16
```

In [16]:
```
c
```
Out[16]:
```
16
```

this is a bit of wacky python that is helpful. You can have more than one thing on each side of an equal sign

In [17]:
```
a,b=1,2
print(a,b)
```
```
1 2
```

In [18]:
```
a,b,c=1,2,3
print(a,b,c)
```
```
1 2 3
```

In [19]:
```
a,b,c='1','hello',2
print(a,b,c)
```
```
1 hello 2
```

In [20]:
```
a,b,c='1','hello',[5,5,6]
print(a,b,c)
```
```
1 hello [5, 5, 6]
```

In [21]:
```
import numpy as np  #more to come on this!
a,b,c='1','hello',np.arange(1,10)
print(a,b,c)
```
```
1 hello [1 2 3 4 5 6 7 8 9]
```

Write down your answer to

$$7 + (6 \times 5^2 + 3)$$

then try it!

In [ ]:
```
```

What about that dumb viral math problem?

$$8 \div 2(2+2)$$

In [11]:
```
```
Out[11]:
```
16.0
```

Now to some printing....

In [22]:
```
a=6
b=4
print ('The result of a/b=',a/b)
```
```
The result of a/b= 1.5
```

In [23]:
```
print ('we can get fancier \na=',a,'\nb=',b,'\na/b=',a/b)  #the \n is a new line
                                                         #you can also to \t for a ta
```

```
we can get fancier
a= 6
b= 4
a/b= 1.5
```

In [24]:
```
output='we can get fancier\na='
#I just set the parameter output to the writing
# and then called it later on.
print (output,a)
```

```
we can get fancier
a= 6
```

In [25]:
```
output='we can get fancier\na='
print (output,float(a)) # we just cast a to a float
```

```
we can get fancier
a= 6.0
```

In [26]:
```
output
```

Out[26]:
```
'we can get fancier\na='
```

In [27]:
```
str(output)
```

Out[27]:
```
'we can get fancier\na='
```

The next set of calls shows you the formal way of formatting all your values. Plus I added the "\" (the slash doesn't always show up) to do a line break below. Play with all the settings. The format is an object oriented function you are calling to set the formating. This link goes through some string stuff and this link gets into the nitty-gritty detail of how to do the formatting. You can set widths, spacing, decimals,and centering. Go through and do some of your own! Go through this slowly. We will be using it a lot to make things nice. You will come back to this!

Take your time here and don't rush. It will save you time later on making things look nice

In [28]:
```
#fancy formatting.
print ('The value of b is {0:*^14.3f}'.format(b),'the value of a is{0:3d}'.forma
#The 0 represents the first item in the list. The star represents the fill.
# the carrot says center it.
# The 14 represents the spaces.
# The 3 repesents the decimals. the f is a float
```

```
The value of b is ****4.000***** the value of a is  6
```

Now lets go through it step by step!

In [29]:
```
print ('The value of b is {} and the value of a is {}'.format(b,a))
# the brackets tell it were to insert a value
# and then you list the values in the format string.
# It by default takes the values in the format string in order
```

```
The value of b is 4 and the value of a is 6
```

In [30]:
```
print ('The value of b is {0} and the value of a is {1}'.format(b,a))
# You can tell it what order to take the values in the format string.
#In Python we always start with 0.
```

```
The value of b is 4 and the value of a is 6
```

In [31]:
```
print ('The value of b is {1} and the value of a is {0}'.format(b,a))
```

```
The value of b is 6 and the value of a is 4
```

In [32]:
```
print ('The value of b is {1} and the value of a is {1}'.format(b,a))
```

```
The value of b is 6 and the value of a is 6
```

Now you can format the values. d is for integer and f is for float.

In [33]:
```
print ('The value of b is {:3d} and the value of a is {:.3f}'.format(int(b),floa
# or you can just not print the decimals for a float
# The 3 in front of the d is for the number of spaces.
#The .3 in front of the f is the number of decimal places.
#The total number of spaces would be in front of
# the decimal place.
```

```
The value of b is   4 and the value of a is 6.000
```

In [34]:
```
#A tricky aside.
format(b,'15.3f')
#why this works both ways is because python is object oriented....
```

Out[34]:
```
'          4.000'
```

In [35]:
```
#lets do more fomatting.  If you want 2  #the slash allows us to split up a long
print ('we are going to divide {} by {}'.format(a,b))
print ('we could turn {} into a float {:.3f}'.format(a,a))
print ('we could turn {} into a float with 10 decimal places \
and 15 spots and a plus sign with stars and centered {:*^+15.10f}'.format(a,a))
```

```
we are going to divide 6 by 4
we could turn 6 into a float 6.000
we could turn 6 into a float with 10 decimal places and 15 spots and a plus sign
with stars and centered *+6.0000000000*
```

I just made all the formatting more complicated then it needs to be to teach you. The key thing to remember is you can control the number of decimal places. That is really nice to make output look nice especially as we start reporting variables on graphs etc (this is foreshadowing, remember this).

I am sure I did a few things you missed. So a review of the critical parts.

1. Set your printed decimal points

In [8]:
```
a=3.14159265
print('pi is equal to {:.3f}'.format(a))
```

```
pi is equal to 3.142
```

1. You can line breaks wherever you want with \n to make things look nice (or ugly)

In [12]:
```
a=3.14159265
print('pi is equal to {:.3f}\nBut maybe I want 5 decimals: pi={:.5f}'.format(a,a
```

```
pi is equal to 3.142
But maybe I want 5 decimals: pi=3.14159
```

1. a tab is \t and adds a nice space

In [18]:
```
a=3.14159265
print('pi\tis equal to {:.3f}'.format(a))
```

```
pi      is equal to 3.142
```

In [ ]: