Ready to code? Might as well dive in…
Go to http://vpython.org/ and read the page where it says "How to Get Started".

Following the instructions, I went to glowscript.org. I clicked sign in and it let me sign in with google id. I don't particular like having to use a google Id but I figured I wanted this done so I did it. I understand if you are morally opposed. Once I got signed in I first wrote the cheesiest program ever:

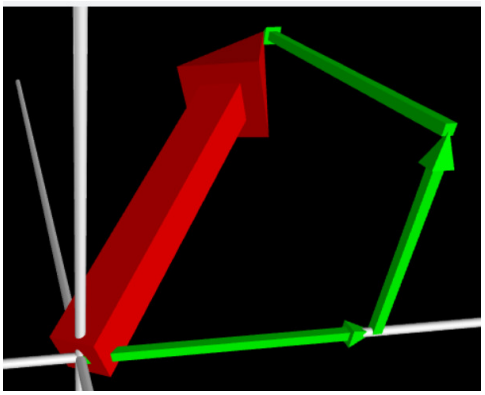
```
box()
```

I then ran the code. Once you hit run, try swiveling the point of view by holding down the right mouse button and moving the mouse. Try zooming by holding both mouse buttons and moving the mouse. Try changing the size of the black window by dragging on the bottom right corner.

Once you feel good, go on to the next page. Note: you might also click on the HELP link in the upper right corner of the glowscript page. This should link you to a page with all kinds of documentation on different types of objects. There are also some links to training videos and a tutorial if you scroll down a bit. Sometimes the documentation is not quite perfect but it is free…

From a book called Matter and Interaction more video tutorials can be found here:
www.Vpython.org/video01.html
www.Vpython.org/video02.html
www.Vpython.org/video03.html
www.vpython.org/video04.html
I started watching the first vid and it seemed pretty instructive and easy to follow.

The code below shows produces a visualization of 3d vector decomposition.



Type the code in yourself (please refrain from cutting and pasting).

You should learn a little bit more by actually typing and finding typos. It's only 30 lines of code…you can do it!

When you run the code, hopefully you end up seeing something like the code shown at right. Note: I used the mouse to zoom and rotate to get this screenshot.

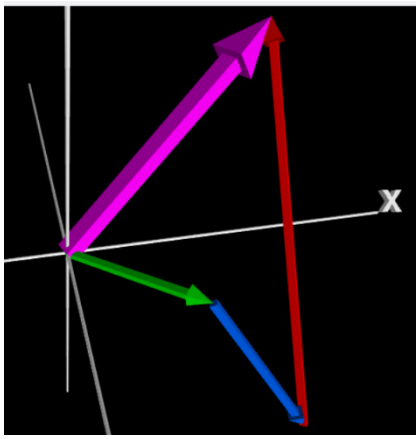
GlowScript 2.7 VPython

```
from visual import *
#this code sketches out how you might do vector addition
#to do 2D vector addition, keep z-components of any vectors equal to zero

#draw a set of coordinate axes
line_x=cylinder(pos=vec(-10, 0, 0), axis=vec(20, 0, 0), radius=0.05)
line_y=cylinder(pos=vec(0, -10, 0), axis=vec(0, 20, 0), radius=0.05)
line_z=cylinder(pos=vec(0, 0, -10), axis=vec(0, 0, 20), radius=0.05)

#create a vector named A
A = vec(6,8,7)
#draw an arrow corresponding to vector A
A_arrow = arrow(pos=vec(0,0,0),
                axis=A,
                shaftwidth=0.5,
                color=color.red)

#create 3 separate arrows for the components of A
#upon running the code, notice I used the tail-to-tip method
#of graphical vector addition: A = A_x + A_y + A_z
A_x_arrow = arrow(pos=vec(0,0,0),
                  axis=vec(A.x,0,0),
                  shaftwidth=0.1,
                  color=color.green)
A_y_arrow = arrow(pos=A_x_arrow.pos+A_x_arrow.axis,
                  axis=vec(0,A.y,0),
                  shaftwidth=0.1,
                  color=color.green)
A_z_arrow = arrow(pos=A_y_arrow.pos+A_y_arrow.axis,
                  axis=vec(0,0,A.z),
                  shaftwidth=0.1,
                  color=color.green)
```

Now create your own code to perform graphical vector addition.
In your new code:

1) Specify (up to) three vectors.
2) Cut and paste the coordinate axes (and axis labels) from the previous code.
3) Make the code do vector math to determine the resultant vector.
4) Correctly position your three vectors using the tail-to-tip method.
5) Draw the resultant from first tail (usually the origin) to the last tip.

Once your simulation runs, it might look something like the figure at right.



Once the code is working, you want to check it with a couple of cases. If only there was a resource available to you that had numerous 3D vector addition problems already solved…

Perhaps you could try the following recommendations from Chapter 3 of the workbook:

- **3.4-3.7**
- Example between **3.4** & **3.5**
- **3.12**
- **3.16**

OPTIONAL BONUS CODE 1: Visualize cross products.

1) Copy and paste the coordinate axis (and labels) codes.
2) Specify two input vectors.
3) Have available Chapter 3 of the workbook. If you look at the table of contents, hopefully it tells you the correct page for cross-product information.
4) Do a web search for "cross product in vpython". Learn how to make the code do your cross-products.
5) Display (and label) each input vector & the output vector from the cross product.
6) Test your code using problems **3.11**, **3.13** and **3.15**.

OPTIONAL BONUS CODE 2: Determine angle between two vectors.

1) Copy and paste the coordinate axis (and labels) codes.
2) Specify two input vectors.
3) Have available Chapter 3 of the workbook. If you look at the table of contents, hopefully it tells you the correct page for dot-product information. There should be instructions on how to determine the angle between two vectors on the same page as the dot product information.
4) Do a web search for "dot product in vpython" and read. Figure out how to make the code compute the angle between your two vectors.
5) Display (and label) each input vector. Label the angle in the display. Alternatively, use a print statement to output the angle between the two input vectors just below the display window.
6) Test your code using problems **3.11**, **3.13** and **3.15**.

NEED MORE? Come talk to me. I am willing to work with you to create oral presentation projects involving coding but I need you to be motivated and proactive. You also need to demonstrate you can get codes done quickly.