The goal of this assignment is to code projectile simulations (with and without drag) to determine potentially useful information.

1) If you haven't already made any simulations with motion, first learn some basics using http://www.robjorstad.com/Phys161/161Lab/161FirstMotionSims.pdf.

2) The code on the next page creates a simulation of projectile motion without drag.
Copy this code (I think you should type it yourself…not just copy and paste).
I believe you learn something when you type as you occasionally make typos and it forces you to fix.
TIP: run the code often while typing it in to check for typos along the way.

3) The following page explains how to add the drag force into the simulation.
While you modify the code, hopefully you learn about both coding and physics.

4) Finally, add in all the "bells & whistles" from the last page of http://www.robjorstad.com/Phys161/161Lab/161FirstMotionSims.pdf.
I did it already and, if you skip to the last page, you can see the kind of thing I think you can do.
Your code should output data that could, in theory, be used to compare the simulation to real life.

5) Optional: use the code to simulate motion of actual examples from lab (perhaps a project for our first oral presentation???). You may need to research drag coefficients. In particular, try to create a realistic model of a cotton ball (or coffee filter) released from rest. It would be super cool to figure out some way to show your simulation and a video of the real world running side by side. I bet you can figure it out…

Projectile

```
GlowScript 2.7 VPython
from visual import *
L = 125                          #define size parameter for drawing the ground
H = 50                           #define the initial height of the projectile
speed = 25                       #muzzle speed (initial launch speed)
theta_deg = 30                   #launch angle (in degrees)
theta = radians(theta_deg)       #converts degrees to radians
g = 9.8      #magnitude of accel due to grav (near earth’s surface) in m/s^2
m = 10                           #mass to be used in kg
b = 0.03                         #this determines the drag coefficient

#Compute initial VELOCITY of projectile.
#Recall speed is a scalar; velocity a vector.
v_0 = vec(speed*cos(theta), speed*sin(theta), 0)

#specify initial position vector of the projectile
r_0 = vec(-L/2, H, 0)

#draw a thin green sheet to represent the ground
ground=box(pos=vec(0, -0.5, 0), color=color.green, length=L, height=0.5, width=L)

#draw a pedestal to indicate where the ball is launched
pedestal = cylinder(pos=vec(r_0.x, 0, 0), axis=vec(0, H, 0), color=color.green)

#draw the ball at the correct initial position
#regarding color vectors: color.red=vec(1,0,0), color.green= vec(0,1,0), etc
#do a web search for “color vector vpython” for more help
ball = sphere(pos=r_0, color=vec(1,1,1), make_trail=True)

#compute gravitational force vector on the ball
F_grav = vec(0,-m*g, 0)

#compute initial MOMENTUM of the ball using momentum_vector = mass*velocity_vector
#we will learn more about momentum in chapter 9
p = m*v_0

#initialize time and define time step size
t = 0
delta_t = 0.1

#initialize a parameter of the ball for height IN FLIGHT
#CAPTIAL H is the INITIAL height
#lower case h will be the projectile’s height at any time as it is flying around
h=ball.pos.y

#create a loop that runs any time the ball is above ground
while h>0:
    rate(10)                         #number of steps displayed per second
    p  = p + F_grav*delta_t          #update ball momentum
    ball.pos = ball.pos + p/m*delta_t   #update ball position (p/m is ball velocity)
    t = t + delta_t                  #increment the time
    h=ball.pos.y                     #re-compute the ball’s current height
```

Now try to modify your projectile code to include drag.
First read a little bit of physics and math to help out.

The *magnitude* of high speed drag (F_{drag}) is given by

$$F_{drag} = bv^2$$

where b is a constant and v is speed. Recall $v = \|\vec{v}\|$ (speed is the magnitude of velocity). It is worth reiterating this equation gives the *magnitude* of drag but not the direction.

The direction of drag points *opposite the direction of motion*. The direction of motion is described mathematically by the unit vector $\hat{v}$. The unit vector $\hat{v}$ has magnitude 1 while pointing in the same direction as velocity $\vec{v}$. Such a unit vector can be created mathematically using

$$\boldsymbol{\hat{v} = \frac{\vec{v}}{v}}$$

Since $\hat{v}$ is the *same* direction as motion, we may use $-\hat{v}$ as *opposite* the direction of motion.
The *vector* drag force is thus

$$\vec{F}_{drag} = bv^2(-\hat{v})$$
$$\vec{F}_{drag} = -bv^2(\hat{v})$$
$$\vec{F}_{drag} = -bv(v\hat{v})$$
$$\vec{F}_{drag} = -bv(\vec{v})$$

To get the last result I used the bold equation shown above.

Finally, as we will learn in chapter 9, momentum ($\vec{p}$) is given by the equation

$$\vec{p} = m\vec{v}$$

We may use this definition to rewrite the drag force (vector) in terms of stuff we use in our code.

$$\vec{F}_{drag} = -b\left(\frac{p}{m}\right)\left(\frac{\vec{p}}{m}\right)$$
$$\vec{F}_{drag} = -\frac{bp}{m^2}\vec{p}$$

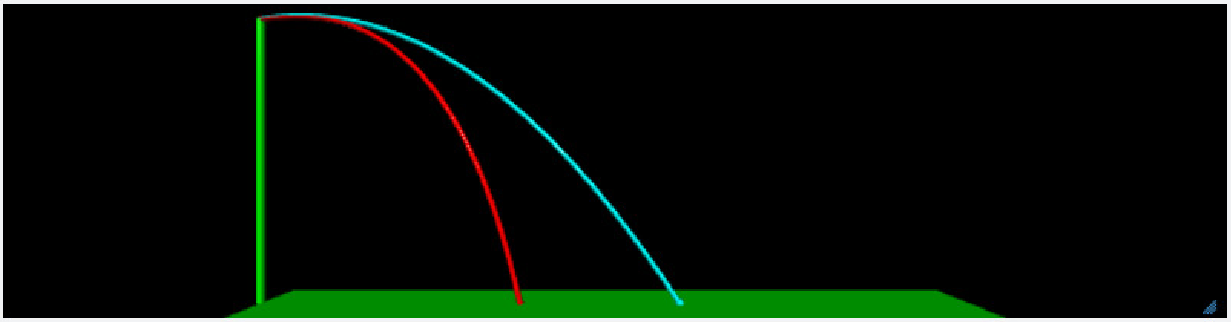
Finally consider the bit of code you want to add shown below.

```
F_drag = -b/m**2*mag(p)*p
F_net = F_grav + F_drag
```

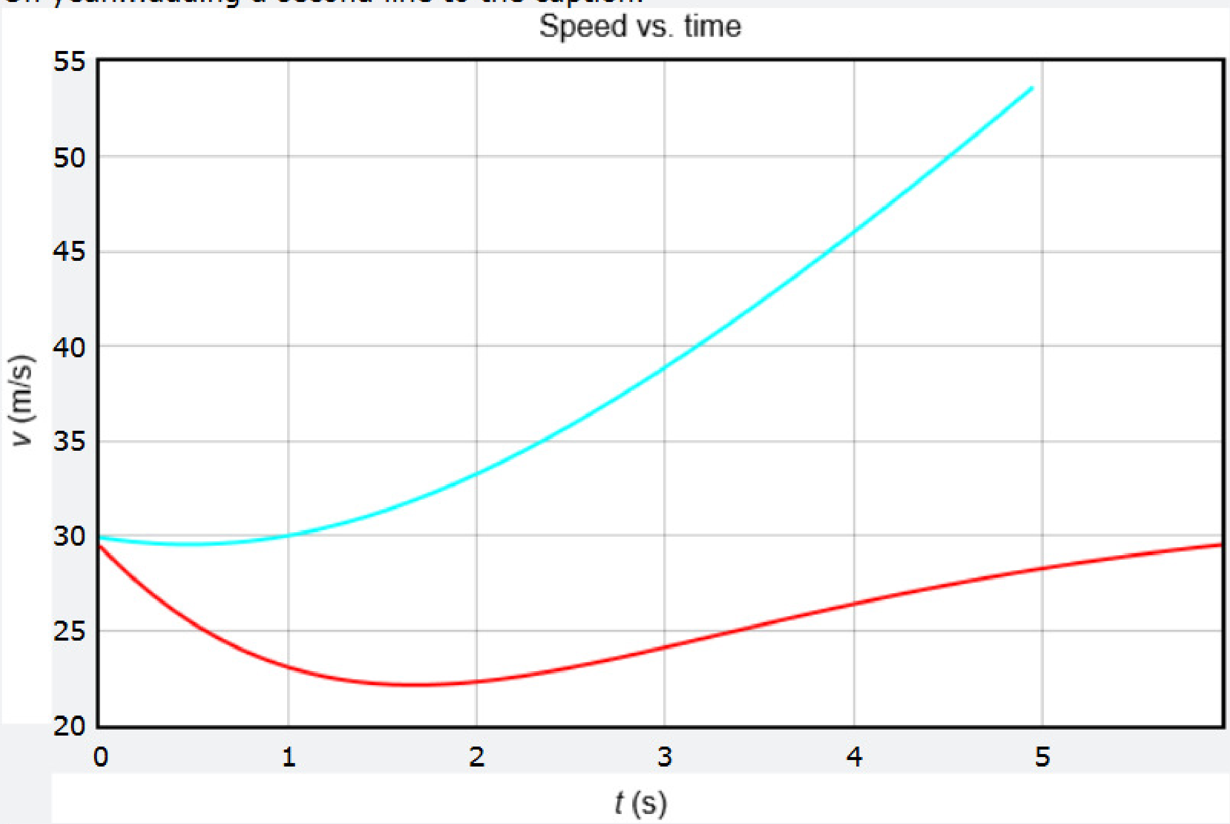
Notice:

- The first line computes the drag force; this is a *vector* quantity.
- Python uses order of operation so my code is correct
- To square the mass I used `**` (not ^)
- To get momentum magnitude $p = \|\vec{p}\|$ I used the python command `mag(p)`
- Compute *net* force is using addition; python knows `F_grav` & `F_drag` are vectors & does vector addition!

Next I created code that launched two balls (one with and one without drag). The balls were launched simultaneously. Afterwards, I added in all the tricks from the last page of
http://www.robjorstad.com/Phys161/161Lab/161FirstMotionSims.pdf.
My screen ended up looking like the screen shot on the next page. Notice I plotted speed versus time for each ball. I also had the code output the time to impact for each ball. See if you can do all that.



Click to begin the simulation.
Oh yeah...adding a second line to the caption.



Time WITHOUT drag is t2=4.99999999999999
Time WITH drag is t1=5.999999999999987