PHYS 391- Lab 1: Introduction into Programming

This lab is to get you acquainted with programming in python, so all directions are python specific. If you wish, you may use any language (including shell scripts) you like EXCEPT Excel or other spreadsheet programs. If you have never programmed before, use python (2.7.x). It is strongly recommended that you complete a tutorial at codecademy.com on python as well. DO NOT WORK WITH A LAB PARTNER FOR THIS LAB! Use the python cheat sheets on the website and examples to get you started. Save all scripts to a file such that if I copied and pasted your script as is, it would give me the correct output (i.e. it works). You will turn in your lab by emailing two separate files (part I and part II) to me.

Goals

- Reading in data files
- Plotting
- Coding simple mathematical operations
- Writing Loops
- Creating functions and calling on command lines.

Preliminary Items: Using the Virtual Machine

##

Starting the Virtual Machine

Click on the virtual machine (VM) icon; click start. After the VM loads, type the following commands in the open terminal window (or any other terminal window in the VM):

- You should see a terminal window in a "virtual" environment. This environment is closed off to the rest of the computer except for a folder called pyraf_share which allows you to drop files in and appear on the pc or mac side.
- In the lab391 directory (you start in this directory), type python, which will start up the python environment.
- Open another terminal and make your own directory using your email name or whatever as long as it's not confused for someone else. Use the command 'mkdir'

  mkdir yourdirectory

  use this to store your files TEMPORARILY. Copy or move these files to pyraf_share so you can retrieve them later. Note: if you don't store your files in a separate directory, someone could copy over them with the same file name. Start the python environment by typing:

  python
- Once you are in the **python** environment, change to your directory by typing the following two commands

  import os

  os.chdir("yourdirectory***")***

Gedit

Before we go any further, go to another terminal window and start the text editor gedit: Type at the prompt:

gedit

This should bring up a text editor window with the file. If you want to edit an existing file, type ***gedit*** filename at the prompt.

Okay now go back to your python window…

Saving work
The virtual box has a linked folder that connects it to the rest of the computer called **pyraf_share.** If you wish to save your work you can copy your files to this folder and retrieve them outside the virtual box. You can also **sftp** your files to your duck account too.

Python environment
Python is like other scripting language such that commands and mathematical expressions are calculated right on the command line. To do math calculations in python type:

import math

Now try these calculations by typing in the following:

sqrt(x)
log10(x)
x**2

see the cheat sheet for more tips.

Important! Turning off Virtual Box etc:
When you are ready to turn off the VM, exit out of ds9 and pyRAF. Close any gedit windows too. Shut off the VM by going under the VM menu item 'system' and select Shut Down... **Don't just click to close the window;** that may screw up the VM and any work you have contained in it. And it will also render the VM useless until we can fix it.
##

Part I: Manipulating a Data file

Data File Description
The data file snlist.txt http://homework.uoregon.edu/pub/elsa/snlist.txt lists all supernovae reported since 1885 through November 2014. It also contains basic information about the properties of the supernova and host galaxy (the galaxy where the supernova exploded). (Note you can save this text file and import it via python_share folder)
Briefly, the columns are:

- Supernova name
- Host galaxy name
- Sky coordinates of the supernova measured in hours minutes seconds. There are six columns; the first three describe the **right ascension (RA)** position and the second set of three describe the **declination (Dec)**.
- Columns **HRV** and **Z** are distance measurements to the supernova in two different unit systems
- **Bmag** is the brightness of the host galaxy measured in blue magnitudes (smaller the number the brighter the object)
- **LogD25** is a logarithmic measurement of the apparent diameter of the galaxy
- **MaxMag** is the brightness of the supernova at its absolute peak
- **Type** is the supernova type (e.g. Ia, II, etc)
- **Disc** is the discoverer.

It's not important that you fully understand what each column means. More information will be given if/as needed and you can always google for more info. Missing numerical data is

substituted with 99s or 99999 depending on the quantity. Missing character data is simply blank.

Instructions
Do the following tasks. **Present your answers as scripts to generate the output or plot exactly. Also show the first and last 10 lines of the output or plot. Save this all to one file.** Email part I to me elsa@uoregon.edu

1) Extract out the Galaxy and **HRV columns.**
2) Extract out rows that contain only NGC galaxies.
3) Plot z vs MaxMag
4) Plot the supernova position in units of degrees. RA is the horizontal coordinate ranging from 0 to 360 degrees and Dec ranges from -90 to 90. The conversion formulas are:

RA (deg) = 15*(H +M/60 + S/3600)
Dec (deg) = (D + M/60 + S/3600)*(-1 if D is negative or +1 if D is positive)

For example 1885A has the coordinates: 00 42 44 +41 16 08
RA = 15*(00 +46/60+44/3600) = 11.68
Dec = (41+16/60+8/3600)*(+1) = 41.269

5) Plot the positions of only Ia and Ia subtypes (e.g. Ia pec etc)

Part II Coding and Plotting a Physics Function:
Instructions
For this portion you will create a complete python script (program) and call it on the command line. You do not need to include the output. Other than changing the path in then #! line, I should be able to run it without any other changes. Hint, Hint, Hint: see the cheat sheets for help. Email part II to me: elsa@uoregon.edu

A. Using your new python skills**, create a program that will plot the position as a function of time for a ball falling from a height of 1000 meters.** You can code this with or without a 'for' loop. Take initial velocity = 0 and g=9.8m/s^2. Note that if your time array is too long, the ball will have already hit the ground and any distance values after this point will be nonsensical since the ground is defined at 0 m. You can do this in a few different ways:
   a. Simply set lower limit of the y axis to 0 when plotting.
   b. Use an 'if statement' that will give a value of 0 for time after the ball has hit the ground or use a 'while' loop.
   c. Or try the python function 'clip' or 'where' on your distance array
      (e.g.
      ```
      >>>a=[9,-10,0,1,2]
      >>>b=a.clip(0)
      >>>b=[9,0,0,1,2])
      ```
   Note! the above function will only work on NumPy arrays. It won't work on lists!
   There are many solutions – find the one that makes sense to you!

B. Make a second plot that graphs velocity as a function of time. Have the velocity equal zero when it hits the ground. This means changing the values to zero if distance =0, which is easily implemented in a loop. If you haven't used a 'for' loop and you can

change all values at a certain velocity to be zero. (Python Code Hint: for multiple plots & after importing matplotlib as plt, use:

```
>>> fig1=plt.figure()
>>> fig2=plt.figure()
>>> dist=fig1.add_subplot(111)
>>> vel=fig2.add_subplot(111)
>>> dist.plot(t,y) #plot the distance
>>> vel.plot(t,v) #plot the velocity
```

C. Part A should be relatively straightforward to code. Unfortunately most physics problems aren't so nice. Let's make this harder and show the use of loops:
Calculate the position when drag force F_D is included. Recall that drag force increases as v^2 meaning that your acceleration is no longer due to just gravity. In fact it becomes the equation of motion:

$$m\frac{dv}{dt} = mg - cv^2$$

where c is the drag coefficient.
Technically you can solve this analytically, but let's pretend that you can't and must solve it numerically which will require a 'for' loop. This will entail correcting the total acceleration on the ball for each time step to account for changing velocity. Basically start with very small steps in time, calculate the ball's velocity and position for the first time increment, then use the former to update the drag force (ergo the total acceleration) on the ball. Now go to the next time step in the loop.
Or more explicitly:
At $t=t_0$:

$v_0(t_0)=0$
$a(t_0)=9.8$
$y(t_0)=1000$
$k=c/m$
$\Delta t = 0.1$

1st time step ($t = t_0+\Delta t$):

- $v(t_0+\Delta t) = v_0(t_0)-a(t_0)* \Delta t$
- $y(t_0+\Delta t) = y(t_0) +v(t_0+\Delta t)*\Delta t$
- $a(t_0+\Delta t) = g-k[v(t_0+\Delta t)]^2$

2nd time step ($t= t_0+2\Delta t$):

- $v(t_0+2\Delta t) = v_0(t_0+\Delta t)-a(t_0+\Delta t)* \Delta t$
- $y(t_0+2\Delta t) = y(t_0+\Delta t) +v(t_0+2\Delta t)*\Delta t$
- $a(t_0+2\Delta t) = g-k[v(t_0+2\Delta t)]^2$

etc….

Please see the cheat sheets for making for loops. You will need create a list of time increments and apply these to v,y,a.

Experiment with different values of k. Drag coefficients (constant c) usually range from .001 to 2. Find a value that can be easily seen (ie on the same scale) as the first plot.

Plot this with your position graph in part A so there are two lines on one plot. Do the same with velocity. In the end when your script is executed, there should be two plots that pop up: position and velocity. The plots should make sense to you. Here's what is necessary for a good grade:

-Distance must decrease in time
-Velocity is negative
-With drag force, the velocity should become constant in time and the corresponding distance should become a linear plot.
-Distance and velocity must stop at zero