

PHYSICS 113 – Recitation Assignment 1

Name

Recitation Assignment # 1 Sep. 23, 2009

You should consider this a trial by fire. If you've never done any programming before, don't worry. We're here to walk you through it. However, you should always have a general idea of what your program is doing. Also, if you've never used linux before, don't worry. You'll get the hang of it.

You may complete this in class. However, if you are unable to do so, it is expected that you complete this for recitation next time.

When a box appears, call over Coleman to check over your progress. When the sheet is complete, you will hand it in. Also, you are expected to email your final programs to Coleman.

If you have any questions, please ask.

1 Getting Started With Linux

Begin by sitting down in front of one of the terminals in Disque 704. Get comfortable. Note that you do not need to sit at the same terminal each class. Your files are shared across the system.

1. The login screen will ask for your username and password. We will give you both.
2. If you've never used linux before, the screen will look a bit unfamiliar, but it's menu driven, just like Macs and Windows. Go to:

Applications → Internet → Firefox Web Browser.

This should be pretty familiar to you, but as a starting exercise, please bookmark the webpage:

http://www.physics.drexel.edu/liki/index.php/Main_Page

I think you'll find it comes in handy.

3. Next, you're going to open up a "shell." A lot of the commands that we're going to run aren't going to be convenient from menus. Instead, go to:

Applications → Accessories → terminal

and you'll get a little box to type commands into.

4. Try the following:

```
xphy9: ~> date
```

Note that the bit in the front is known as the "prompt". You don't type that. You should find that the terminal writes out the date.

5. It is also important for you to change your password to something secure that you'll remember. To do this:

```
xphy9:~> yppasswd
```

and then simply follow the directions.

6. While we can't give you a complete list of all possible commands, there are a few which will come in handy. The format is

“command name” [optional things]

Note, you don't actually include the brackets when you type the command. To use the optional “-l” in the first example, simply type “ls -l”

- ls [-l]
List all of the files in the directory. -l gives more details.
- mkdir progs
Create a new directory called “progs.” (You can, of course, name it whatever you want). Try running “ls” after you create the directory.
- cd progs
Change to directory “progs.”
- pwd
Print out what directory you're currently in.
- cd
Change to your “home” directory.
- cd ..
Move “up” one directory level.
- more filename
Read the contents of a file as plain text. To figure out which files in your home directory can be read, run the “ls” command.

There are lots more, and we'll give you a detailed list of additional commands shortly.

7. Finally, the last thing you should be able to do is log out. This is also menu driven. Search around and then log out. Make sure you log out each time.

2 Simple Python

1. Log into a workstation, pull up firefox, make yourself comfortable, etc.
 2. In firefox, go to the webpage <http://vpython.org/>, which will serve as a handy manual for programming in visual python (the language that we will be using throughout this course). We would recommend that you bookmark this webpage in firefox.
- 3. Open up a “terminal” and go into your progs directory.

4. We're now ready to start running python. At the prompt, type "python". Something like this should appear:

```
Python 2.5.2 (r252:60911, Jun 12 2008, 12:29:49)
[GCC 4.0.1 (Apple Inc. build 5465)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

At the prompt you can type commands.

5. Let's start simple:

```
>>> 5+3
```

Does that give you what you expect? How about:

```
>>> 9*2
```

```
?
```

One more:

```
>>> 7/3
```

Wha? That's not right! And that's your first lesson. If you put in integers, the answer it will give you will be an integer. To get the answer you like, put in:

```
>>> 7./3.
```

6. Python can do more than be a calculator. You can also do algebra. For example:

```
>>> x=4
>>> y=3
>>> z=x**y
>>> print z
```

In this case, we used the power function ($x ** y$), as well as the print command.

There's lots more, but since you know addition, subtraction, multiplication, and division, you should be good for now.

7. In this course, the main thing we're going to do is use the "visual" library. This will allow us to use commands to make and move objects. Let's start simple:

```
>>> from visual import *
>>> a=sphere()
```

The second command should be obvious, but the first might look a bit confusing. It simply tells python to load up all the "visual" commands.

8. Our sphere is lonely. Let's make another one:

```
>>> b=sphere(pos=(5.0,0,0),color=color.blue)
```

Note a couple of things. First, since you already loaded up the visual library, you don't need to do so again. Second, we can describe other things about our sphere. "pos", for example, tells the computer where to put the sphere. "color.blue" paints it blue.

In general, you'll see a lot of commands which will have this formula: attribute=value. Get used to it.

Of course, if I want to know something about my sphere, I can do the following:

```
>>> print a.pos
>>> print b.pos
>>> newpos=(a.pos+b.pos)/2.
>>> print newpos
```

If you follow what we did line by line, you won't be surprised to see that the vector returned by the print statement is just the average of the other two.

9. If we wanted, we could create a third sphere:

```
>>> c=sphere(pos=newpos,color=color.red,radius=0.5)
```

Notice that I added the "radius" attribute to make the red sphere smaller than the other two. Also note that we put the new sphere between the other two.

- 10. As a final trick, we might want to move the red sphere. Try the following:

```
>>> c.pos=c.pos+(1,0,0)
```

What do you know? It jumped to the right.

11. That's all of the commands we want to give you for now. The "VPython" website has lots of other shapes that you can make.

Before we finish up this section, take your mouse, and experiment holding buttons and moving the mouse within the window.

12. Finally:

```
>>> exit()
```

Should close up your python session

3 A Python Program

Of course, typing the commands again and again is a pain. It is much better for us to write a program, so that we can just run all of the commands at once.

To do this, we're going to use an editing program called "emacs". Emacs will allow to write nice python code, has groovy shortcuts, and also will color and space the text to help you.

While you don't need to know all of the shortcuts, you may want to know a few (they'll really help everything go fast):

- ctrl-x ctrl-s – Saves a file
- ctrl-x ctrl-c – Closes up the emacs window
- ctrl-k – Deletes an entire line

Beyond that, just use your intuition.

Let's get started by example:

1. On the command line, run:

```
xphy9:~> emacs prog1.py &
```

This will pull up the emacs file editor. (The “&” allows you to keep using the shell while emacs is running).

2. Write the following simple program (inside the emacs window) and save it:

```
from visual import *
sun=sphere(radius=2,pos=(0,0,0),color=color.yellow)
```

This simple program has two commands. The first tells the python interpreter to include all of the visual libraries (the commands that draw on the screen).

The second command tells it to create a sphere called “sun” with a radius of 2.

- 3. After saving your program, you will run it by typing:

```
xphy9:~> python prog1.py
```

- 4. I also want you to create another sphere, a blue one, about half the size of the “sun”, and put at position (10,0,0). You should call it “earth.”

Put the appropriate command in your program, save it, run it, and then call over Coleman to take a look.

5. In your program, after creating your “earth” you want to give it a velocity, and set a few other variables:

```
# Prof. Goldberg
earth.v=vector(0,4,0)
t=0
dt=0.01
scene.autoscale=0
```

If you run this program, what happens?

Nothing.

The first line is just a comment. You should put comments in your codes as a reminder of how things work. The computer doesn't care about those. Replace my name with your own.

Even though in the second line you've set an attribute called "v", the earth just sits there. The computer doesn't know what that means. You have to *tell it*

t=0, and dt=0.1 don't really do anything. They just set two variables to two different values. We'll see how they are used in a moment.

The command "scene.autoscale=0" tells the computer that from now on, don't automatically resize the screen.

- 6. Add the following "loop" in your program after you've set everything at the top.

```
while (t < 5):  
    rate(50)  
  
    t=t+dt  
    print t
```

Save the program and run it.

This is an example of a "while loop". Basically, it runs everything underneath the loop again and again until whatever is in the parentheses is no longer true.

Note: The indentations are important!

The "rate" command says that the loop should run at most 50 times per second. You'll want something like this so that the code doesn't run too fast.

At every step, you add t=t+dt, which means that after the first step, it's 0.01, and then 0.02, and so on.

- 7. Here's where we make a movie. Basically, at time through the loop, you want to move the "earth" ever so slightly. So you'll want to add the line (under the loop):

```
earth.pos=earth.pos+dt*earth.v
```

Again, save and run this.

Mathematically, this is nothing more than:

$$\begin{aligned}\vec{v} &= \frac{\Delta\vec{r}}{\Delta t} \\ \Delta\vec{r} &= \vec{v}\Delta t \\ \vec{r}_{new} &= \vec{r}_{old} + \vec{v}\Delta t\end{aligned}$$

- 8. As the final part of the assignment, change the numbers so that the "earth" moves with a speed of 5, and at an angle 120 degrees from the positive x-axis.

Make sure you test it and email your final program to Coleman at ckraw@physics.drexel.edu. The subject line should be "Physics 113 yourname prog1.py"