

Lab 1 - Playing with Python

Thursday 31 August, 2006 - Due: Thursday 7 September

The aim of this lab is to get you started with programming in Python. While programming has a strong abstract and theoretical component, it is also something which most people learn with practice and experimentation. Here, you will learn about IDLE, the environment in which you develop Python programs, standalone programs and the interactive shell.

1 Some introductory comments

Programming takes patience and practice. Writing good programs requires planning and a logical approach. For someone who has not programmed before there is usually a period of frustration. Don't worry - this is normal and will decrease with practice. Be careful to type things *exactly* as they are written in the lab – computers are very literal beasts and require all i's to be dotted and t's to be crossed before they will do your bidding. If in doubt refer to your Python book and the online documentation <http://www.vpython.org/webdoc> for Visual Python.

1.1 Python

For the purposes of this class, a program in Python is simply one or more text files. These files are collections of lines of text that give a recipe telling the computer what to do. A program can be more than one file: one of the files will be the main program and the other files will be *modules* that give the main program additional capabilities. These files are read and *interpreted* by Python as commands. These commands have meanings such as "print this" or "draw that", multiply these two numbers or do the following commands 1000 times. The Python interpreter, while reading the instructions, manages the data you create and follows your instructions to give more detailed instructions to the computer on what to draw or multiply or print. Python is called a high level language which means that you can tell the computer to do quite complicated things with relatively few, transparent instructions. The interpreter is then able to understand these instructions and translate them ultimately into binary instructions (strings of 0 and 1's) which can be fed to the CPU of the computer to execute.

1.2 IDLE

IDLE stands for Integrated DeveLopment Environment. It is the interface that you will use for writing and executing Python programs. It contains (among other things):

- An interactive *shell* - the shell executes your Python commands each time you type them in and hit **return**.

- A program editor - this allows you to put a set of commands into a program file. Once you enter and save a program file, IDLE has the capacity to **run** your program as a sequence of commands, rather than one at a time.

1.3 VPython

VPython is a special graphical extension to Python (technically a *module*) which gives a very nice simple environment to produce animations (simulations) of physical systems and record quantitative output data in the form of, for example, graphs. Typically a separate window is fired up to contain the graphical output.

1.4 Lab reports

You should record all your work and observations for each lab into a separate file (use Word, wordpad or your favorite text/document editor). You can subsequently print this out on the cluster printer to hand in.

2 Starting IDLE and the shell

1. You need to log onto your PC. Enter your NetID and password. You should save all of your work to your CMS/Unix account (the I: drive of the PC) and/or email it to yourself.
2. Open up MS word or some other editor for your lab report. Put your name on the top of the page, the lab number and the date.
3. Your report is to be a narrative of what you do. Be concise but be complete. I expect your lab report to be 2/3 pages typically.
4. Save your report to a file (on your I: drive or SUnix account) with a name like "My-name.lab1.doc". This will help organize your work from this class.
5. Now launch IDLE by first looking under the Start menu in windows. Find "SU Academic Departments", then "Physics". There will be a menu choice for "VPython". Select that item.
6. IDLE will open up a single window. There are two types of window that Python uses:
 - Shell windows. Shell windows have some text in them and contain the *prompt* which looks like this " >>>> "
 - File windows: these are for editing programs. It is usually empty at start.

Record in your report what the type of your first Python window.

- If your first window is a shell window, call up a file window for fun. Do this by going to the **File** menu at the top of your shell window and selecting **New**

window. Close it the usual way by clicking on the cross at the top right corner of the window.

- If your first window is **not** a shell window, call up a shell window. Do this by going to the **Run** menu at the top of your current window and selecting **Python shell**. Then close your window by clicking at the top right corner.

7. Now that you have a shell window, give the interpreter some commands to execute. Enter a command by typing a command after the " >>> " prompt and then hit the **Enter** key. Try the following:

```
print "Hello, there!"
print 3
print 2+7
print "2 times 3 equals", 2*3
print "a"
a=3
print "a"
print a
```

8. Note the following

- In these examples you used the **print** statement to write to the shell window. You can write **expressions** like `2*3` or a variable like `a`.
- You can use/write **string literals** - characters enclosed by quotes.
- You created a **variable** and assigned it a value with the command `a=3`.

9. Now try the following and note down your results:

```
a=3
b=5
print a+b
print a*b
c=range(1,13)
d=range(1,5)
print c
print d
print c+d
print a**b
print a/b
```

```
print b/a
```

10. These examples show that not all variables contain single numbers. The range function creates a list of integers which are subsequently assigned to the variables `c` and `d`. What does the `+` operation do in this case? What do you notice about the value of `a/b`? Be careful when dividing integers - Python will assume you want an integer result from dividing two integers. What happens if I promote `a,b` to **real** numbers by adding a decimal point?

3 Writing and running program files

Interactive mode as exemplified by the Python shell is neat for experimenting with simple commands but becomes tedious for creating real programs. To do the latter we build sequences of commands and store them in a file. This allows you to use the program many times with ease and carry out small changes without retyping everything.

1. Open up a file window (see earlier).
2. First, give this file a name and save it. Go to the **File** menu and select **Save as**. Click on **My Computer** and then the I: or K: drive. Give the program a name eg. 'MyFirstPythonprogram.py' and save it to your SUnix account.
3. Create a short program. Enter the following into your empty file window, hitting the enter key at the end of each line. Note: you can use the arrows to move around in the file and click to move the cursor.

```
a=3
b=9
c=range(1,11)
print a, "+", b,"is",a+b
print "A list of the numbers between 1 and 11 is",c
d=range(a,b)
print "A list of numbers between",a,"and",b,"is",d
```

4. Save your program as before.
5. Run your program. Under the **Run** menu select "Run program" OR just hit F5.
6. What happens. Try to fix any errors you see by editing the program and re-executing the program.
7. Modify your program. Make it more interesting. Include the modified program in your report. To do this:
 - Select the whole program by finding **Select all** on the **Edit** menu

- Copy your selection using **Copy** under **Edit**
- Pasting your selection into your report (**Paste** under the **Edit** menu in Word)

4 VPython

Now we will experiment with the graphical capabilities of Python.

1. Open the file window and type the following line

```
from visual import *
```

This accesses the VPython visual module containing many useful graphical functions. Any VPython program we will write will contain this line.
2. Add the line `ball=sphere()` to your program. This creates a variable of type `sphere` and optionally gives it the name `ball`. Compare `c=range(1,10)` we discussed earlier.
3. Now run the program. Describe what happens.
4. It might be nice change the color of this ball object. In Python this can be accomplished by specifying various **attributes** of the ball. Remove the original line creating the ball and replace it with

```
ball=sphere(color=color.green)
```
5. By default Python places such an object at the center of the screen. To place it somewhere else we need to give its location. In general VPython is a 3D graphical environment so it is necessary to give its location in 3D Cartesian coordinates (x,y,z). Try the following line to create a green sphere with name `ball` and place it at the location (2,3,4)

```
ball=sphere(color=color.green,pos=vector(2,3,4))
```
6. To move it is easy. Add (do not remove the line to create the ball) the new line

```
ball.pos=vector(-1,-2,-3)
```

You should see that the ball has shifted position. In general whenever you change an attribute of one of VPython's built in geometrical shapes, Python handles any redrawing that is necessary.
7. Experiment with modifying other attributes such as its size (`radius`) and creating other shapes, e.g., `cylinder`. You will find a nice introduction to the Python visual environment under **Help/Visual**. Add the lines to your report.
8. Finally, from the file menu **Open** go to the demos folder and look at some of the VPython sample programs there. Try `bounce` and `bounce2`. What happens when you drag the picture using the right mouse button ? What does the middle button do ? You should start to see the power of the VPython 3D environment.