***Lab 1: Using the interpreter***

1.      *Interaction*. Using a system command-line, IDLE, or other, start the Python interactive command line (>>> prompt), and type the expression: "Hello World!" (including the quotes). The string should be echoed back to you. The purpose of this exercise is to get your environment configured to run Python. In some scenarios, you may need to first run a cd shell command, type the full path to the python executable, or add its path to your PATH envi­ronment variable. Set it in your .cshrc or .kshrc file to make Python perma­nently available on Unix systems; use a setup.bat, autoexec.bat, or the environment variable GUI on Win­dows.

2.      *Programs*. With the text editor of your choice, write a simple module file—a file contain­ing the single statement: print 'Hello module world!'. Store this statement in a file named module1.py. Now, run this file by using any launch option you like: running it in IDLE, clicking on its file icon, passing it to the Python interpreter pro­gram on the system shell’s command line, and so on. In fact, experiment by running your file with as many of the launch techniques we’ve seen in this unit as you can. Which technique seems easiest (there is no right answer to this one, of course)?

3.      *Modules*. Next, start the Python interactive command line (>>> prompt) and import the module you wrote in the prior exercise. Does your PYTHONPATH setting need to include the directory where the file is stored? Try moving the file to a different directory and importing it again from its original directory; what happens? (Hint: is there still a file named module1.pyc in the original direc­tory?)

4.      *Scripts*. If your platform supports it, add the #! line to the top of your module1.py mod­ule, give the file executable privileges, and run it directly as an executable. What does the first line need to contain? Skip that if you are working on a Windows machine (#! usually only has meaning on Unix and Linux); instead try running your file by listing just its name in a DOS console window (this works on recent flavors of Windows), or the “Start/Run…” dialog box.

5.      *Errors*. Experiment with typing mathematical expressions and assignments at the Python interactive command line. First type the expression: 1 / 0; what happens? Next, type a variable name you haven’t assigned a value to yet; what happens this time?

You may not know it yet, but you’re doing exception processing, a topic we’ll explore in depth in a later unit. As we’ll learn then, you are technically triggering what’s known as the *default exception handler*—logic that prints a standard error message.

For full-blown source-code *debugging* chores, IDLE includes a GUI debugging interface (select Debug before running your script), and a Python standard library module named pdb provides a command-line debugging interface (more on pdb later). When first starting out, though, Python’s default error messages will probably be as much error handling as you need—they give the cause of the error, as well as the lines in your code were active when the error occurred.

6.      *Breaks*. At the Python command line, type:

L = [1, 2]

L.append(L)

L

What happens? If you’re using a Python newer than release 1.5, you’ll probably see a strange output. If you’re using a Python version older than 1.5.1 (now ancient history!), a Ctrl-C key com­bination will probably help on most platforms. Why do you think this occurs? What does Python report when you type the Ctrl-C key combination? Warning: if you do have a Python older than release 1.5.1, make sure your machine can stop a program with a break-key combination of some sort before running this test, or you may be waiting a long time.

7.      *Documentation*. Spend at least 6 minutes browsing the Python library and language manuals before moving on, to get a feel for the available tools in the standard library, and the structure of the documentation set. It takes at least this long to become familiar with the location of major topics in the manual set; once you do, though, it’s easy to find what you need. You can find this manual in the Python “Start” button entry on Windows, in the “Help” pulldown menu in IDLE, or online at [www.python.org](http://www.python.org/). We’ll also have a few more words to say about the manuals, and other documentation sources available (including PyDoc and the help function), in the statements unit. If you still have time to kill, go explore the Python website ([www.python.org](http://www.python.org/)), and the Vaults of Parnassus site.  Especially check out the python.org documentation and search pages; they can be crucial resources in practice.