MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.081—Introduction to EECS I
Spring Semester, 2007

Work for Week 1

This handout contains:

- Software Lab for Tuesday, February 6
- Pre-Lab exercises due Thursday, February 8 at 2PM; you should come and do them in lab on Wednesday, February 7. **This week: Everyone should come and bring their laptop, if they have one. We'll have a software installation party.**
- Robot Lab for Thursday, February 8
- Post-lab exercises due Tuesday, February 13 at 2PM

All pre- and post-lab exercises are to be handed in individually. Please see the collaboration policy on the web site for more information.

Getting started

Schedule In general, the weekly schedule for 6.081 will be like this:

- Tuesday lecture from 2–3:30, followed by a software lab from 3:30 - 5:00. We will also hand out the homework for the coming week. The homework will include a part that is due on Thursday before lab and a part that is due the following Tuesday before lecture. The homework will have parts to write up and turn in, as well as parts to be done with the online tutor.
- Wednesday evening homework help session from 6–10. Attendance at these sessions is strongly recommended. You are welcome to do the homework for Thursday on your own if you prefer, or in a self-organized study group. But we suggest that your time will be much better spent if you do your pre-lab homework in the staffed Wednesday sessions. When programming, especially, it's easy to get lost in rat-holes and spend enormous amounts of time digging yourself out. Doing your work when there are staff members around can make things much easier.
- Thursday afternoon lab. Each lab will end with a short quiz based on the pre-lab assignment as well as the material covered during the lab.
- Post-lab homework, due before the following Tuesday lecture.

Reading This is such a cool new way of looking at the material of intro EECS, there is no textbook. We will be producing draft notes each week, and those will be your primary reading resource. In addition, for Python, we recommend finding a basic language reference that you like and also reading *How to Think Like a Computer Scientist*.

- Official Python tutorial: `http://http://docs.python.org/tut/tut.html`

- *How to Think Like a Computer Scientist: Learning with Python*, by Allen Downey, Jeffrey Elkner, Chris Meyers. This book assumes no programming experience. It is not a Python reference manual; it is a computer science text that uses Python as an example. Highly recommended. `http://www.greenteapress.com/thinkpython/thinkCSpy.pdf`
- *Learning Python*, by David Ascher and Mark Lutz. This book also assumes very little/no programming experience, and is longer (not so great for learning Python in a hurry, but covers topics in great detail, so is good as a reference guide if you know what you're looking for). `http://proquest.safaribooksonline.com.libproxy.mit.edu/0596002815?tocview=true` (You need an MIT Certificate to view this one)
- *Learn Python in 10 Minutes*, by Poromenos. If you have very little time and a fair bit of programming experience in another language, this tutorial covers Python's syntax quickly. `http://www.poromenos.org/tutorials/python`
- Look at some of the other reference material on the class web site under "Resource Material" (under the "General Information" menu at the top of the web page).[1]

Software Lab

Starting and using IDLE

IDLE is a simple way to edit and run Python programs. To start it on one of the lab laptops, go to a shell and type

```
> idle-python2.4
```

Python Shell The Python shell acts a little bit like a calculator. You type in expressions, Python evaluates them, and then prints the result. So, if you type

```
>>> 4 + 4
```

It will print out `8`. Play with the shell a little bit.

Edit a file You'll want to use the shell to test things out, but not to write your programs. If you want to start defining procedures, you should open a new file (choose **`New Window`** from the **`File`** menu) and write your definitions in there. So, start by making a file containing this definition:

```
a = 'hi'
b = 7
def f(x):
    return x + 1
```

Now, choose **`Run Module`** from IDLE's **`Run`** menu. It will act as if you have typed the text of your file into the shell. If there was something obviously syntactically wrong with your file (parentheses not closed, for example), IDLE will tell you about it and highlight the point in your file where the problem is. Otherwise, the shell will print out something like

[1] If you use Windows with Internet Explorer as your web browser, you won't see the drop-down menu under "General Information". There's a "Resource Material" link at the bottom of the home page that you can follow instead.

```
>>> ================================ RESTART ================================
>>>
```

And now you can ask it to evaluate expressions, including things you've defined in your file.

Question 1. Use the Python shell to compute `f(f(f(b)))`.

Question 2. What happens if you do `f(a)`?

To do the following exercises, use the shell for experimentation, and write new procedure definitions in your file. Whenever you change your file, you'll need to do `Run Module` again.

Archive your file When you do work on a lab laptop, you should always remember to mail or FTP your files back to yourselves. We don't guarantee that you'll always get the same laptop or that any files that you leave on it will remain there from week to week.

An easy way to keep your files safe is to use a webmail service. You may have a service such as Gmail or Hotmail before. We'll explain the process using MIT webmail.

To use MIT webmail, first, you go to: `http:\\webmail.mit.edu`. To log in, type your Athena account user name and its password. You can compose an email to your email address (`user@mit.edu`, where `user` is your account name), and attach your files.

Another way to protect your files is to copy the files from a lab laptop to your Athena account. You can do this by using the `scp` command. General usage of `scp` (when copying the file to the remote computer) is as follows.

```
file user@machine:file
```

For example, if you want to copy the file `lab1.py` to the directory `Courses/6.081` on your Athena account `user1`, you type:

```
scp lab1.py user1@athena.dialup.mit.edu:Courses/6.081/lab1.py
```

Note that the directory specified in the destination (`Courses/6.081`) must have been created before copying a file into it. Also note that `scp` command automatically overwrites the file you specified as the destination, without asking your permission. So, you have to be careful not to overwrite your important files using `scp`.

When you want to copy the file from Athena to a lab laptop, you can simply switch the destination and the source for `scp`:

```
scp user1@athena.dialup.mit.edu:Courses/6.081/lab1.py lab1.py
```

Exercises

Start by reading the sections of the Lecture 1 notes describing the new features in Python.