Lab 01

Learning Objectives

I. **What characters do I type next?** *How do I get Python to compute things for me?* This lab will provide an opportunity to practice integrating and applying the basic syntax of the Python programming language towards interesting calculations typical of science research.

II. **How do I plan a project and execute my plan?** *How do I write a Python program from start to finish?* This lab will provide an opportunity to practice careful construction of a multilayered program.

III. **What broke and how do I fix it?** *How do I figure out where the code is breaking?* This lab will provide an opportunity to practice print debugging to locate your error.

A. **Python is a physical system. Experiment!** *If you don't know a bit of syntax, see if you can guess it.* This lab will provide an opportunity to practice one method for learning how to learn new syntax.

Today we'll draw on one of the major applications of scientific computing: **computer modeling**. In future classes we'll also delve into the other major application, **data analysis**.

While You Work will give structure and motivation to practicing experimentation in Python.

Parts 1 and 2 give you the opportunity to write programs that calculate and display approximations to two common mathematical constants, φ (the golden ratio) and π.

In **part 3**, you will submit your code for the day.

In **part 4**, if you have time, you will learn how to SSH into corn and set up a GitHub account.

While You Work: Look for New Syntax

Any programming language has a lot of little syntactical details. We've covered the most important ones in lecture, but there will probably be times while you're working that you stray beyond the material in lecture. This is your opportunity to practice guessing that syntax.

If you figure out a bit of new syntax, share it with the rest of the class! Call over one of the instructors and we'll put it on the board.

Part 0: Today's Python Interpreter

So that you can get started with Python right away, today we'll be using an online interpreter. Navigate to

`http://repl.it/languages/Python`

and wait for it to load. **Before doing anything else, click the "Save" button at the top to create a customized URL**, and make sure to save regularly while you work. You'll use this URL at the end of class to submit your work for **part 1**. For **part 2**, open a new browser tab and get a different URL. You'll use the pane on the left-hand side to write code. Anything you print will appear on the right-hand pane.

If that's not working for you, let the instructors know. They might suggest you try this interpreter instead:

`http://www.tutorialspoint.com/execute_python_online.php`.

Part 1: The Fibonacci Sequence and Calculating the Golden Ratio

The Fibonacci sequence is defined by $f_{n+1} = f_n + f_{n-1}$, giving the sequence:

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 …

The ratio of successive elements converges to

$$\varphi = (1 + \sqrt{5})/2 \approx 1.61803,$$

the famous *golden ratio*. **Your task is to approximate** φ.

Here we make some suggestions about how to approach this task. They will give you opportunities for practice, so we recommend following along. However, if you're feeling confident, feel free to try to pave your own path.

Your program should implement two functions:

- `fib` should calculate the n th element in the Fibonacci sequence. It should have one argument, `n`. Its return value should be the n th element.
- `phi_approx` should:
  - calculate the approximate value of φ obtained by the ratio of the n th and $(n-1)$ st elements, $f_n/f_{n-1} \approx \varphi$; and
  - print n, f_n, f_{n-1}, and φ.

  It should have one argument, `n`. Its return value should be the approximate value of φ. This function should probably call `fib`.

After defining those two functions, pick a few values of n, call `phi_approx` with them, and see how close your approximation gets!

Part 2: Calculating Pi

Now you'll learn how to calculate π using a *Monte Carlo method*. Instead of calculating π analytically from the limit of a series or an integral, a Monte Carlo method finds a fast approximation using random numbers and, in this case, the area of a circle. Here's how we'll do it:

1. Take a large number of coordinate pairs (x, y) in the unit square: $x, y \in [0, 1]$.
2. Count the number of points that fall inside the unit circle in this quadrant, i.e. the points where $x^2 + y^2 \leq 1$.
3. Divide this by the total number of points to approximate the area of this segment = $\pi/4$.

Your task is to use this method to approximate π. In this part, we'll give you less guidance—see what you can do on your own. Once you are confident in your implementation, see how close you can get!

But you will need one tip. To generate a random number between 0 and 1, do two things:

- put the statement `import random` at the top of your program; and
- when needed, call e.g. `x = random.random()`.

We'll talk more about how this works in upcoming classes.

Part 3: Submitting Your Code

Today you'll be submitting your code via Google form. Navigate to
http://goo.gl/forms/oFOmW6krtK
and paste in the relevant URLs. Then hit Submit.

Part 4: (If You Have Time) Setting up an SSH mechanism and a GitHub account

Starting on Thursday, we'll be using Stanford's "corn" machines remotely to run Python. The following Tuesday, we'll start learning how to use git and GitHub. In preparation, please use your remaining time to get set up.

The procedure for logging into corn depends on your operating system. On Windows, you'll need to install some open-source software first. Navigate to
http://web.stanford.edu/class/physics91SI/cgi-bin/?page_id=947,
open the appropriate file, and follow the directions.

To open a GitHub account, navigate to
https://github.com/
and click "Sign up." Pick a username you can remember, since you'll need it next week.