# Python Lab 0

## July 11, 2015

The goal of this lab is to get you installing `Python` and everything else you need, and then making your first program (`Hello World`). As a bonus, there's also a part on making a `Github` account. You should really do this, it's good for getting a job.

1. Download/install `Python` (make sure it's version `3.x.x`, not version `2.x.x`)

2. Download a text editor. I like `gVim`, but there's a learning curve. `Notepad++` or `gedit` is easier, but not as useful. You can be noob and use the simple Windows `Notepad` or the equivalent for Macs, but it's not built for coding. Don't be a noob, do things properly from the start.

3. *Optional*: Download a nice terminal GUI. You'll be using the terminal/command prompt a lot to run Python. For example, the command prompt in Windows, the GUI isn't very good (you can't resize the windows, the colours are weird, etc.). I like to use `Console2` on Windows, which I can resize and customize the colours so it looks nicer (see here).

4. *For Windows only*: Download `Cygwin`, or its lightweight alternative, `GNU On Windows` (`GOW`). CS people usually use `UNIX` commands (e.g. Linux, Mac OS) for telling the command prompt/OS what to do. Windows uses a different system (`DOS`). I prefer using Windows as my native OS, but I install the `UNIX` commands so I can use them in Windows. That's what `Cygwin` does. But I find it a bit too big, and to start off you'll only need a few commands, so `GOW` is easier to use. I'd recommend that `https://github.com/bmatzelle/gow`.

   Once you have that installed, you should test it out. Open your terminal (e.g. `Console2`) and try commands like `cd` (change directory, i.e. folders), `ls` (list files in a directory), `pwd` (path to working directory, i.e. the current folder). Try to explore the different folders on your computer from the command line using these commands. Other useful commands are `mv` (move files from one directory to another), `cp` (copy files from one place to another), and `rm` (remove files... be careful you don't delete important files/folders with this one).

5. Make sure your Python is installed correctly and recognized by the OS. In your command prompt (after Cygwin or GOW is installed), type:

   ```
   >> which python
   ```

   (Don't type the >>, that symbolizes command prompt input)

   The which command is a UNIX command which returns where the OS thinks the executable (in this case, python) is installed. It should return the path to wherever you installed it. For example, I see:

   ```
   E:\Programs\Python3.3.2\python.EXE
   ```

   because I installed Python in the E:\Programs\Python3.3.2\ directory.

   Note, when you type in python in the command line, it really runs python.EXE. You could type the .EXE if you wanted to, but you don't have to.

   To make sure you have the right version, type:

   ```
   >> python --version
   ```

   This runs python with the --version flag. Basically this is a built-in function of python asking it to return what version it is. Make sure it is at least version 3.x.x.

6. Write your first Python program. Open your text editor and type in:

   ```
   print("Hello World!")
   ```

   Save this file as *hello.py*. Keep note of the path of this file (i.e. the directory). Open up your terminal/command prompt and cd to that directory. Type ls to list all the files there, your *hello.py* should be there. Once you're there, type the following:

   ```
   >> python hello.py
   ```

   You should see the output in the terminal on the following line as:

   ```
   Hello World!
   ```

   Congrats! You wrote your first Python program!

## Bonus: Create a Github Account

When you write an essay, you probably have one Word document that you make changes to and save. Maybe you save different versions of it as you go, but usually people just have that one copy. Now let's say you write a version on Monday, add a new paragraph on Tuesday and overwrite another one on Wednesday, but you realize you did something wrong and want to go back to the version you had on Monday. If you only had one saved copy of that essay, you're shit out of luck - you'll have to rewrite the version you had on Monday (if you remember what it even was).

Instead of an essay, imagine it's software code for a program. If you have a version of your code working, and you add stuff/make changes that screws it up, it'd be good to: 1) know what those changes were, and 2) be able to revert to the working version. Computer scientists and software engineers do this using `version control`. There are several different programs to do this: `svn`, `git`, and `Perforce` to name a few. `Perforce` is popular for companies like Facebook and Google (I used it at Altera too), but it costs a lot of money. For open-source, `git` is the most popular. You can ask Dad what he uses; he's worked at different companies so I'd imagine he's familiar with many different types of version control.

Working with `git` is easy. You `clone` the latest version of the repository to your local computer. You make changes to a file (or a collection of files), and save that as a version, or a `commit`. Then you upload or `push` that to the repository. You save the files into an online repository, the most popular being `Github`. This repository saves the different versions you've worked in chronologically, and what the differences between each version was (so you can see what was changed that's causing issues).

Try it. Create a `Github` account (`www.github.com`). This is your online code repository. Follow the official `Github Hello World` tutorial, it takes about 10 min. Once you do that, try to install `git` on your computer. If you don't want to use `git` from the command-line, there's a nice `GitHub` GUI which you can install and sync with your account.

Once you have it up and running, try to submit the code you just wrote. Go to my `Github` repository, `learnpython`. Clone this repo to your lo-

cal machine. You'll notice a folder called `example_student`, with a subfolder `lab00_submission`. There is a single file there, the *`hello.py`* file.

Do something similar to submit your *`hello.py`* file. At the top-level directory of the repository, create a folder with your `Github` username (instead of `example_student`). Then create the `lab00_submission` and put your program there. Commit this file and push it to the repo. Once it's done, we should be able to see it online. I will write a script to test your program and make sure it gives the correct output.

You can make your own repos and work on your own code. You can make repos public so people can see your work. It's useful for getting jobs, you can show employers what you've worked on if you want to try for internships next summer.