Python Set Up

This page explains how to set up Python on a machine so you can run and edit Python programs, and links to the exercise code to download. You can do this before starting the class, or you can leave it until you've gotten far enough in the class that you want to write some code. The Google Python Class uses a simple, standard Python installation, although more complex strategies are possible. Python is free and open source, available for all operating systems from python.org. In particular we want a Python install where you can do two things:

- Run an existing python program, such as hello.py
- Run the Python interpreter interactively, so you can type code right at it

Both of the above are done quite a lot in the lecture videos, and it's definitely something you need to be able to do to solve the exercises.

Download Google Python Exercises

As a first step, download the google-python-exercises.zip file and unzip it someplace where you can work on it. The resulting google-python-exercises directory contains many different python code exercises you can work on. In particular, google-python-exercises contains a simple hello.py file you can use in the next step to check that Python is working on your machine. Below are Python instructions for Windows and all other operation systems:

Python on Linux, Mac OS X, etc.

Most operating systems other than Windows already have Python installed by default. To check that Python is installed, open a command line (typically by running the "Terminal" program), and cd to the google-python-exercises directory. Try the following to run the hello.py program (what you type is shown in bold):

```
~/google-python-exercises$ python hello.py
Hello World
~/google-python-exercises$ python hello.py Alice
Hello Alice
```

If python is not installed, see the Python.org download page. To run the Python interpreter

interactively, just type "python" in the terminal:

```
~/google-python-exercises$ python
Python 2.5.2 (r252:60911, Feb 22 2008, 07:57:53)
[GCC 4.0.1 (Apple Computer, Inc. build 5363)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 1
2
>>> you can type expressions here .. use ctrl-d to exit
```

For Google's Python Class, you want a python version that is 2.4 or later, and avoiding the 3.x versions for now is probably best.

Execute Bit (optional)

The commands above are the simplest way to run python programs. If the "execute bit" is set on a .py file, it can be run by name without having to type "python" first. Set the execute bit with the "chmod" command like this:

```
~/google-python-exercises$ chmod +x hello.py
~/google-python-exercises$ ./hello.py   ## now can run it as ./hello.py
Hello World
```

Python on Windows

Doing a basic Python install on Windows is easy:

- Go to the python.org download page, select a version such as 2.6. Google's Python Class should work with any version 2.4 or later, and avoiding the 3.x versions for now is probably best.
- Run the Python installer, taking all the defaults. This will install Python in the root directory and set up some file associations.
- With Python installed, open a command prompt (Accessories > Command Prompt, or type 'cmd' into the run dialog). Cd to the google-python-exercises directory (from unzipping google-python-exercises.zip). You should be able to run the hello.py python program by typing "python" followed by "hello.py" (what you type is shown in bold):

```
C:\google-python-exercises> python hello.py
```

```
Hello World
C:\google-python-exercises> python hello.py Alice
Hello Alice
```

If this works, Python is installed. Otherwise, see Python Windows FAQ for help.

To run the Python interpreter interactively, select the Run... command from the Start menu, and type "python" -- this will launch Python interactively in its own window. On Windows, use Ctrl-Z to exit (on all other operating systems it's Ctrl-D to exit).

In the lecture videos, we generally run the Python programs with commands like "./hello.py". On Windows, it's simplest to use the "python hello.py" form.

Editing Python (all operating systems)

A Python program is just a text file that you edit directly. As above, you should have a command line open, where you can type "python hello.py Alice" to run whatever exercise you are working on. At the command line prompt, just hit the up-arrow key to recall previously typed commands, so it's easy to run previous commands without retyping them.

You want a text editor with a little understanding of code and indentation. There are many good free ones:

- Windows -- **do not use Notepad or Wordpad**. Try the free and open source Notepad++ or the free and open source JEdit
- Mac -- The built in TextEdit works, but not very well. Try the free TextWrangler or the free and open source JEdit
- Linux -- any unix text editor is fine, or try the above JEdit.

Editor Settings

To edit Python, we advocate the strategy that when you hit the tab key, your editor inserts spaces rather than a real tab character. All our files use 2-spaces as the indent, and 4-spaces is another popular choice. It's also handy if the editor will "auto indent" so when you hit return, the new line starts with the same indentation as the previous line. We also recommend saving your files with the unix line-ending convention, since that's how the various starter files are set up. If running hello.py gives the error "Unknown option: -", the file may have the wrong line-ending. Here are the preferences to set for common editors to treat tabs and line-endings correctly for Python:

- Windows Notepad++ -- Tabs: Settings > Preferences > Edit Components > Tab settings, and

Settings > Preferences > MISC for auto-indent. Line endings: Format > Convert, set to Unix.

- JEdit (any OS) -- Line endings: Little 'U' 'W' 'M' on status bar, set it to 'U' (i.e. Unix line-endings)
- Windows Notepad or Wordpad -- do not use
- Mac TextWrangler -- Tabs: Preference button at the top of the window, check Auto Expand Tabs. Can set the default in Defaults > Auto-Expand Tabs and Auto-indent. Line endings: little control at the bottom of each window, set it to Unix
- Mac TextEdit -- do not use
- Unix pico -- Tabs: Esc-q toggles tab mode, Esc-i to turns on auto-indent mode
- Unix emacs -- Tabs: manually set tabs-inserts-spaces mode: `M-x set-variable(return) indent-tabs-mode(return) nil`

Editing Check

To try out your editor, edit the the hello.py program. Change the word "Hello" in the code to the word "Howdy" (you don't need to understand all the other Python code in there ... we'll explain it all in class). Save your edits and run the program to see its new output. Try adding a "print 'yay!'" just below the existing print and with the same indentation. Try running the program, to see that your edits work correctly. For class we want an edit/run workflow that allows you to switch between editing and running easily.

Quick Python Style

One of the advantages of Python is that it makes it easy to type a little code and quickly see what it does. In class, we want a work setup that matches that .. a text text editor working on the current file.py, and a separate command line window where you can just hit the up-arrow key to run file.py and see what it does. (Teaching philosophy aside: the interpreter is great for little experiments, as shown throughout the lectures. However, the exercises are structured as Python files that students edit. Since being able to write Python programs is the ultimate goal, I felt it was best to be in that mode the whole time, using the interpreter just for little experiments.)

Last updated December 13, 2012.