**Introduction to Python Development**

# Environment Setup

## Installing Python3.7 on CentOS7

**Download and Install Python 3 from Source**

$ sudo -i

$ yum groupinstall -y "development tools"

$ yum install -y \

libffi-devel \

zlib-devel \

bzip2-devel \

openssl-devel \

ncurses-devel \

sqlite-devel \

readline-devel \

tk-devel \

gdbm-devel \

db4-devel \

libpcap-devel \

xz-devel \

expat-devel \

postgresql-devel

$ cd /usr/src

$ wget http://python.org/ftp/python/3.7.2/Python-3.7.2.tar.xz

$ tar xf Python-3.7.2.tar.xz

$ cd Python-3.7.2

$ ./configure --enable-optimizations

$ make altinstall

$ exit

*Important:* make altinstall causes it to not replace the built in python executable.

Using **sudo nano /etc/sudoers** (or your preferred text editor), ensure that secure\_path in /etc/sudoers file includes /usr/local/bin. The line should look something like this:

Defaults secure\_path = /sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin

**Upgrade Pip (might not be necesary)**

The version of pip that we have might be up-to-date, but it's a good practice to try to update it after the installation. We need to use the pip3.7 executable because we're working with Python 3, and we use sudo so that we can write files under the /usr/local directory.

$ sudo pip3.7 install --upgrade pip

## Installing Python3.7 on Debian/Ubuntu

**Download and Install Python 3 from Source**

sudo -i

$ apt update -y

$ apt install -y \

wget \

build-essential \

libffi-dev \

libgdbm-dev \

libc6-dev \

libssl-dev \

zlib1g-dev \

libbz2-dev \

libreadline-dev \

libsqlite3-dev \

libncurses5-dev \

libncursesw5-dev \

xz-utils \

tk-dev

$ cd /usr/src

$ wget http://python.org/ftp/python/3.7.2/Python-3.7.2.tar.xz

$ tar xf Python-3.7.2.tar.xz

$ cd Python-3.7.2

$ ./configure --enable-optimizations

$ make altinstall

*Note:* make altinstall causes it to not replace the built in python executable.

Ensure that secure\_path in /etc/sudoers file includes /usr/local/bin

**Upgrade Pip (might not be necesary)**

The version of pip that we have might be up-to-date, but it's a good practice to try to update it after the installation. We need to use the pip3.7 executable because we're working with Python 3, and we use sudo so that we can write files under the /usr/local directory.

$ pip3.7 install --upgrade pip

## Picking a Text Editor or IDE

**Terminal based editors**

There are two terminal based editors I would consider to be extremely popular:

* [Vim](https://www.vim.org/) - Modal editor, extremely customizable.
* [Emacs](https://www.gnu.org/software/emacs/) - Unbelievably customizable, not modal (at least not by default).

Both of these tools are either pre-installed or readily available on all major Linux distros.

**GUI Based Editors**

GUI based editors can be extremely powerful and more aestetically pleasing than terminal based editors. This list is comprised of classic "text editors", but most of them can be enhanced using plugins that add additional functionality. I'm going to divide them into two camps: native applications and Electron applications (built using JavaScript). This seems like a weird distinction, but plenty of people don't like the resource overhead that running Electron based applications requires.

Native:

* [SublimeText](https://www.sublimetext.com/) - Multi-platform. Very performant and extended using Python 3.
* [Notepad++](https://notepad-plus-plus.org/) - Windows only. Not as powerful as the others, but a good starter text editor that won't get in your way.

Electron Based:

* [Atom](https://atom.io/) - The original Electron based editor. Aestetically pleasing and very extendable through plugins.
* [VS Code](https://code.visualstudio.com/) - Arguably the most popular GUI based editor. Vast ecosystem of plugins and built-in debugger.

**IDEs**

The primary IDE that is used by people in the Python community would be [PyCharm](https://www.jetbrains.com/pycharm/). There is a free community edition and there is also a paid edition. To connect to a remote server to do your editing, you'll need to have the paid version.

## Setting up a Vim Development Environment

A mettre plus tard si nécessaire

# Running Python

## Using the REPL (Read, Evaluate, Print, Loop)

Pour lancer le python installé il suffit de taper python3.7

Tu quittes l’interface en tapant exit()

## Creating and Running Python Files

**Creating Our First Python Script**

Tu crées un script python avec vim puis pour le lancer tu as différents choix :

Soit à l’endroit où il y a ton fichier tu tapes : python3.7 NomDuFichier.py

**Setting a Shebang**

You'll most likely want your scripts to be:

1. Executable from anywhere (in our $PATH)
2. Executable without explicitly using the python3.7 CLI

Thankfully, we can set the process to interpret our scripts by setting a shebang at the top of the file:

*hello.py*

#!/usr/bin/env python3.7

print("Hello, World")

Then we need to make the file executable using chmod:

$ chmod u+x hello.py

Then run the script now by using **./hello.py**

**Adding Scripts to Our $PATH**

Now we need to make sure that we can put this in our $PATH. For this course, we'll be using a bin directory in our $HOME folder to store our custom scripts, but scripts can go into any directory that is in your $PATH.

Let's create a bin directory and move our script:

$ mkdir ~/bin

$ mv hello.py ~/bin/hello

Here's how we add this directory to the $PATH in our .bashrc (the .bashrc for this course already contains this):

$ export PATH=$PATH:$HOME/bin

Finally, let's run the hello script from our $PATH:

$ hello

Hello, World!

Using Comments

**Single Line Comment**

We can comment out a whole line:

# This is a full line comment

**What About Block Comments?**

Python does not have the concept of block commenting that you may have encountered in other languages. Many people mistake a triple-quoted string as being a comment, but it is not — it's a multi-line string. That being said, multi-line strings can functionally work like comments, but they will still be allocated into memory.

"""

This is not a block comment,

but it will still work when you really need for some lines of code to not execute.

"""