Installing Python on Windows

We provide two strategies for setting up a Python environment on windows.

1. Bash on Ubuntu on Windows
2. Using a Virtual Machine

We recommend Strategy 1, as it will give you a development environment that is similar enough to the macOS and Linux instructions that you will be able to act almost as if you have a Linux computer.

We are fine with Strategy 2, although it is essentially just avoiding the problem by throwing computational resources at it.

If Strategy fails horribly, Strategy 2 will definitely work, but is fairly resource intensive.

So, we recommend trying out Strategy 1 first. If that fails, decide whether you are willing to allocate a nontrivial amount of computing resources to this course. If so, proceed with Strategy 3.

(1) Bash on Ubuntu on Windows

Requires: 64-bit Windows 10, updated to the 2016 Anniversary build or later. *If you regularly download updates, you'll be fine.*

Windows 10 has added a Ubuntu subsystem which we will use for development in this class. In particular, we'll use the Ubuntu system to download Python and to create a virtual environment for this course.

First, follow these instructions from HowToGeek in order to activate the "Windows Subsystem for Linux," get Ubuntu from the Microsoft Store, and launch a `bash` shell on Ubuntu.

One quick note here: Your Windows file system is located at `/mnt/c` in the Bash shell environment.

Once you're in the `bash` shell, you can follow the Linux guide, which has more details. If you're just getting started, you may also need to run the following commands:

```
$ sudo add-apt-repository ppa:deadsnakes/ppa
$ sudo apt-get update
$ sudo apt-get install python3.8
$ sudo apt-get install python3.8-venv
```

You can check which version of Python you've installed by running:

```
$ python3 --version
Python 3.8.0
```

In broad strokes, you will need to run the following commands to set up a virtual environment. The Linux and macOS handouts contain more detailed information.

```
$ python3 -m venv "${HOME}/py-env"
$ source "${HOME}/py-env/bin/activate"
(py-env)$ pip install "prompt-toolkit==2.0.10" "ipython[all]" jupyter
jupyterlab numpy scipy matplotlib nltk scikit-learn requests flask pyc
odestyle autopep8 Pillow
(py-env)$ deactivate
```

The command `source "${HOME}/py-env/bin/activate"` is especially important. **Every time you open a new bash shell, you will need to run `source "${HOME}/py-env/bin/activate"` in order to activate your virtual environment.** The `deactivate` command deactivates an active virtual environment.

You can tell if a virtual environment is active by looking for the parenthesized `(py-env)` prefix.

Working with Windows is complicated enough, so we're going to omit the instructions for how to use `virtualenvwrapper` to set up managed virtual environments. If you're really interested, you can follow the macOS instructions for `virtualenvwrapper`.

(2) Use a Virtual Machine

It can be very hard to properly set up development environments on Windows. We're going to give up on Windows, and instead we'll use VirtualBox to run an entire Linux operating system on your Windows computer. First, download VirtualBox 6.1.0. Make sure you know where you have downloaded this file.

Great! We're halfway there.

Next, you'll need to download a Unix OS. We recommend using Ubuntu 18.04. Now,

1. Launch VirtualBox by double-clicking on the downloaded executable.

2. Create a new VM instance and point the prompt to the Ubuntu ISO you just downloaded.
3. VirtualBox will prompt you to configure lots of settings for your new virtual machine. You can use the defaults, or you can adjust the settings if you know what you are doing. Roughly speaking, the more resources you give to your VM, the fewer your normal non-VM computer has. a. You can name your virtual machine something like `cs41-vm`
4. Click through the on-screen instructions to finish setting up Ubuntu.

Ubuntu is a Linux distribution, so from here you should follow the Linux guide that has already been posted.

If you'd like to be oriented to VirtualBox itself, they have posted a manual (warning: it's pretty long) that covers First Steps with VirtualBox. If you only read one section, we recommend "Section 1.9: Running Your Virtual Machine."

Credit

Much of this guide was based on a similar handout written by Sam Redmond.