Setup and Installation

Throughout the course of this module, you will need access to three different software elements: a programming environment, a terminal, and a version control system. This document outlines how to set up each of these elements on both Windows and Apple computers, using both the command line and the graphical user interface (GUI) in PyCharm.

1 Programming Environments

A *programming environment*, or *integrated development environment* (IDE), is a workspace in which programmers create, run, and debug programs. The programming language we are using throughout the module is Python, but the environment is ultimately up to you; different people have very different preferences, and we are happy to let you use whatever you're comfortable with.

That being said, if you have no preference then the one we would recommend is PyCharm, due to its integration with the other two essential software elements for this module (the terminal and version control). Again, if you already have, or can set up on your own, other solutions for these elements, you are welcome to keep using the IDE of your choice, as we also present standalone options for these elements.

If you want to install PyCharm, you'll first need to install Python. You can do this for either platform (Windows or Mac) by going to `www.python.org/downloads`. Here you can find the latest release of Python 2.7 for your operating system (at the time of writing this it is 2.7.14).

After installing Python, you can find PyCharm at `www.jetbrains.com/pycharm/download/` and again download it for your operating system. When you open PyCharm for the first time, you'll see the option to create a new project or to check out from version control. If you click on 'Create New Project' then you should see options for a location and an interpreter. Leave the interpreter as the default and feel free to set the location to wherever you want.

2 Terminal

When you open a terminal window in an operating system, you are opening a program that executes commands that you type. At first, you are greeted by a *prompt*, which is often a character like `?` or `>`. In our code snippets we use `>` to represent the prompt, although it varies across operating systems.

Due to its universality (as exhibited by its integration into later releases of Windows such as Windows 10), we will be using the *Bash* command language for the Unix shell (`en.wikipedia.org/wiki/Bash_(Unix_shell)`). On a Mac, entering these types of commands simply involves opening the Terminal application (as Mac is built on top of Unix). In Windows, your best bet is to install Git from `git-scm.com/downloads`, as it also solves the problem of providing a version control system. You can then go to Control Panel > System > Advanced Options, and select from the bottom of the Advanced tab the Environments Variables button. From there, select the path and add to it the paths related to the executables you want to use (e.g., `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Git` should work if you installed `git-bash.exe` in the default location). To open a window into which you can enter bash commands, you can just open `git-bash.exe`.

If you'd rather not use a standalone program, the terminal is also integrated into PyCharm: you can access it by going to View > Tool Windows > Terminal. In Mac, this will provide the Unix shell by default, but in Windows you may need to update it (after installing git) by going to Settings > Tools > Terminal and selecting 'git bash' as the command language.

After you have set things up, open a terminal (using whatever application you'd like) and type `ls`. If you see a list of things come up, then you're ready to start the lab.

3 Version Control

One of the most important parts of being a programmer is writing and debugging your code collaboratively, sharing it with others, and allowing others to alter it to fit their purposes. Even for yourself, it is considered good practice to use *version control*, which provides the opportunity to revisit old solutions. Believe it or not, there are better ways to achieve this than creating mass email threads with your Python files attached.

One of the most popular protocols for version control is *Git*, and one of the most popular ways to use Git is the centrally maintained service Github. (Running private instances of Git, using services like Gitlab, may be useful for a particularly sensitive project.) This is so popular that it is generally acknowledged as the only real way to maintain and distribute open-source projects; e.g., the codebase for the entire Bitcoin cryptocurrency is on Github at `www.github.com/bitcoin/bitcoin`.

If you have followed the instructions above, the chances are high that you already have Git installed. To find out, enter `git --version` into a terminal and see if you get any meaningful answer. On a Mac, this will work if you have XCode installed, and on Windows it will work if you're using the terminal in `git-bash.exe`. If it doesn't work on a Mac, you can either install XCode or install Git using the same link as for Windows provided above. Once you get a meaningful answer (e.g., `git version 2.7.0` or something similar), you're ready to start the lab.

If you would prefer to integrate Git directly into PyCharm, you're welcome to follow the instructions at `www.jetbrains.com/help/pycharm/using-git-integration.html`. In the lab handouts, however, all instructions regarding Git will be given with respect to the command line.