Python

A Project-based learning approach

After all, tomorrow can only grow out of today—and if one seeks success tomorrow, the factors of success need to be prepared today. --Errico Malatesta, 1892.

Hello and welcome to our introductory Python class. Here's how we suggest you use this document:

1. This document serves as a guide covering most of the material in class. Use it for your reference if you are stuck anywhere during your tasks.
2. The majority of your learning will be done via doing, rather than memorizing. The more you code, the better you will become!
3. The notes in here will definitely not be comprehensive! It's impossible to squeeze the entirety of programming in Python into a single book, let alone a training document like this. You are more than welcomed to use the Internet to look up more tricks and techniques.
4. Remember to have fun!

Font Usage

Words written in `monospace` are commands to be typed, or references to specific objects like script names or environments. For example:

```
def hello()
```

Terminal output will be `grey out monospace`. For example:

```
ls
```

```
bin  boot  dev  etc  home  keybase  klauncherXM3425.1.slave-
socket  lib  lib64  mnt  opt  proc  read-write  root  run
sbin  selinux  srv  sys  tmp  usr  var
```

Setting up your development environment

Before we start programming, it is very important to set up your development environment. Like having a good workshop, having a proper development environment setup will enable you to program efficiently, and most often correctly. Here, we are going to show you how to set up Anaconda along with your programming IDEs.

We will be using the latest stable version of Python (3 in this case) unless otherwise stated in this course. Python, like all other programming languages, is being continuously modified and improved in relation to enhancements to the language as implemented by the community, or in response to identified problems with a certain way the coding was done that can only be fixed by changing the syntax of the scripting language itself.

Unfortunately, when this happens, breakages may occur as packages written in Python2 fail to work without the code being rewritten to fit the Python3 syntax. Other times, you may have no choice but

to use Python2 for your projects, which is where setting up Anaconda for a Python2 environment will make coexisting projects less of a headache.

Anaconda, not the Snake!

Anaconda was developed to solve a very specific problem, which is to manage Python packages for various Python-based projects, often times to do with data science. Packages may depend on other packages, which are referred to as dependencies. Sometimes, there may be conflicting dependencies, which can result in a very untidy development environment, as your programs randomly fail to compile.

With Anaconda, we can create specific environments targetting a particular project, with it's own particular dependencies. We will cover how to install and use Anaconda on Ubuntu, a very popular Linux distro, and Windows 10.

We will know you are ready to do Python programming once you can run this hello_world.py code

```
     Code 1: Your first Python script!
def hello():
   """Print "Hello World" and return None."""
   print("Hello World")


hello()
```

Installing Anaconda

You should always consult the official installation guide rather than relying on guides like ours for the most up to date installation methods for your operating system. We will however, endeavour to provide the most basic summary to get you up and running as soon as possible.

Keep the official guide open in your browser Window at all times and keep track of what stage of the installation process you are at.

Installing on Ubuntu (and other Linux distros)

The official installation guide for Linux recommends installing some software dependencies first. If you are using Ubuntu, follow the guide under Debian, as Ubuntu is a Debian-derived Linux distro. Run the following in your terminal:

```
sudo apt-get install libgl1-mesa-glx libegl1-mesa libxrandr2
libxrandr2 libxss1 libxcursor1 libxcomposite1 libasound2
libxi6 libxtst6
```

You will notice that we have included `sudo` at the beginning of the command. This is because we need root privileges to install these dependencies system-wide. Make sure you have admin privileges to your account before you proceed.

Once your software dependencies have been installed, download the Anaconda shell script installer. When you follow the download link, make sure you are getting the 64-bit Individual Edition, which is freely distributable under an open source license. Download the installer to your Downloads folder so when you run the installation command from the terminal, it will locate and run the installer.

Note that the version of Anaconda that you have downloaded may be different from the command below. If so, change the filename accordingly.

```
bash ~/Downloads/Anaconda3-2020.02-Linux-x86_64.sh
```

You will first be asked to read through the license agreement. Press the Spacebar to quickly scroll to the bottom. You need to type `yes` to proceed.

Thereafter, we recommend accepting all default options in the installer unless you know exactly what you desire to customise. You should be able to launch Anaconda Navigator now by running `anaconda-navigator` in the terminal, or through Ubuntu's app launcher by typing Anaconda and clicking on the Anaconda icon.

Installing Anaconda in Windows 10

Just like with Ubuntu, it is important to follow the official Windows installer guide for getting Anaconda set up right. We will summarise the steps that you need to follow, but urge you to read and follow the official instructions which may change by the time you read this document.

First, download and run the Windows installer. Make sure that you are getting the 64-bit edition of Anaconda Individual Edition. You may choose to accept all the default settings for the installation.

Creating your first Anaconda environment

Let's prepare to write our first Python program using the Anaconda paradigm of development, by setting up an environment first for your project.

1. Launch Anaconda Navigator
2. Click on the Environment tab, and click on the Create button.
3. Name our environment hello_world, and make sure the box for Python is checked. Apply your settings and wait for the environment to be generated.
4. In the terminal, run:

   ```
   conda info –envs
   ```

   To check and see hello_world is listed as one of the available base environments.

Note that if you followed the recommended default installation options, the base environment will already be created and selected for you by Anaconda.

Installing an IDE

An integrated development environment (IDE) is a sophisticated program that makes your life easier while programming by including all the basic tools to compile and run your programs. There are several IDEs we can use depending on the type of project you wish to run.

All IDEs have there own individual quirks, which are geared towards different workflows. In comparing between the three IDEs we will be installing here, VS Code is the most generalist of the IDEs, as it allows one to pivot to other programming languages if needed. Jupyter is heavily-oriented towards statistical programming in Python and R with line-by-line code execution being it's speciality. Spyder occupies the ground of a general purpose Python IDE, which means you can use it to write Python apps, or use it for statistical programming.

Different instructors may have different preferences too for IDEs. For now, it is good enough to know these are the options you have, and that you can switch between them, and other IDEs you may find online until you find one that fits your idea of the perfect development workflow.

Installing Visual Studio Code

Visual Studio Code (VS Code) is a versatile IDE that can be used with many different programming languages, based upon the extensions that are available for you to install. We recommend using this IDE largely because it can be used beyond Python if you decide to learn another programming language.

As is the case with Anaconda, we recommend referencing the official guides (Linux or Windows) for installing VS Code for your operating system. If you are using a Linux distro that is not listed in the official guides, check out the package repositories for VS Code to see if it has already been packaged for your distribution.

For Ubuntu, and other distros that support the Snap store, you can install VS Code by running the following in the terminal:

```
sudo snap install --classic code # or code-insiders
```

You can also install VS Code by downloading and running the deb installer.

For Windows, you can download the installer from the official site and run it to set up VS Code. We recommend the User Setup unless you need to have VS Code available to all users on your computer.

Installing Spyder and Jupyter

Spyder and Jupyter are IDEs that are designed specifically for statistical programming. You should install them if your project is heavily-oriented towards using Python for data analysis. Because Anaconda itself originated as a tool to make data analysis as painless as possible, you can easily install these two IDEs from Anaconda.

Installing Spyder and Jupyter is as easy as opening the Home tab in Anaconda Navigator, finding them, and clicking the Install button. In fact, they may even be installed by default!

hello_world with Visual Studio Code

1. Click on VS Code in the Home tab of Anaconda Navigator.
2. Once VS Code launches, it will offer to download and install Python-based extensions as it detects the hooks from Anaconda. If it does not, saving your file as `hello_world.py` will also trigger the recommendation for Python extensions.

3. In VS Code, take a look at the bottom left-hand corner of the IDE. You will see that it is pointed towards the base Python environment from Anaconda. Click on it and select the hello_world environment instead.
4. Click on File > New File, and type in the script in Code 1.
5. Click on File > Save As, and save your file `hello_world.py`.
6. If you have installed the Python extensions as recommended in step 2 above, you should see an arrow pointed towards the right in your scripting tab. Click on it and you will see a bunch of output in the terminal and most importantly:

   ```
   Hello World
   ```

Congratulations! You have successfully run your first Python program in VS Code.

hello_world with Spyder

1. Click on Spyder in the Home tab of Anaconda Navigator.
2. In the scripting pane, write down the hello_world code from Code 1.
3. Press F5 or click on the green arrow on the menubar that says “Run file” in the hover text.
4. You should see a bunch of output from the console and the output:

   ```
   Hello World
   ```

Congratulations! You have executed your first Python script in Spyder!

hello_world with Jupyter Notebook

1. Click on Jupyter Notebook in the Home tab of Anaconda Navigator. Jupyter will open in a browser tab.
2. Select New > Notebook > Python 3
3. In the input cell, enter the hello_world script from Code 1.
4. Click on the Run button. You should see the output with your coding cell showing:

   ```
   Hello World
   ```

Congratulations! You have executed your first Python script in Jupyter!

Updating Anaconda for a specific environment

The convenience of using Anaconda is you can choose to keep separate environments running with their own distinct set of Python packages with specific dependencies intact. Let’s test it out by updating packages in the `hello_world` environment and compare the version with that in the `base` environment.

1. In the Environment tab, select hello_world
2. To filter out only updatable packages, select Updatable from the pull down menu from the default Installed filter setting. Take note of the version number of the program to be updated.

3. Click on the green Apply button and in the verification page, click on Apply again.
4. Once the installation is complete, take a look at the version of the updated package. Change back to the base environment. You should see that the package in the base environment remains the same.