Python Bootcamp Documentation

Release 1.0.0

septiembre 04, 2020

List of Tables

The goal of this bootcamp is to provide students with a condensed introduction to computation in the context of scientific data analysis. Early exposure to computation is vital to ensuring that future scientists are ready to thrive in our increasingly data-rich world. We'll use the programming language, Python, which is widely-used and beginner friendly.

This bootcamp has three main sections, outlined below:

Chapter 1

Getting Started

Here students are guided through

- downloading Python
- opening and navigating the command line
- writing a simple program and running it

Enter *here for Mac users* or *here for Windows users*.

Chapter 2

Learning The Basics Of Python

Here students are provided with

- the fundamental topics one should understand in order to learn a programming language
- free Python learning resources
- avenue quizzes to test knowledge

Enter *here*.

Chapter 3

Python As A Tool For Science

Here students are provided with

- what makes Python so good for scientific computing
- simple Python script templates to perform common data analysis tasks

Enter *here*.

3.1 Getting Started On Mac

3.1.1 Downloading Python

First, we need to download the software for the programming language that we want to code in, the language we'll use is called Python.

Download from the following link: https://www.python.org/downloads/mac-osx/

3.1.2 Opening Your Command Line Interface

In order to use a computer, we need some way of interacting with the computer's functions and services. Luckily, every operating system (Mac, Windows, etc.) comes with user-interfaces. The one that you are probably most familiar with is the graphical user interface (GUI), which allows users to interact with files, folders, programs, etc. using graphical icons (e.g. Finder on Mac and My Computer on Windows). The other kind of user interface is called a command line interface (CLI). This differs from a GUI in that it is text-based, the user enters commands and executes them. The CLI is what programmers use to navigate their computer and run programs, so it is important to become familiar with it. Each operating system's (OS") CLI differs in name and some syntax, but that's about it.

Mac's CLI is called Terminal, in order to open it, press Command+Space, then type "Terminal" and press Enter.

3.1.3 Basic Navigation Commands

Data on a computer is kept in files which are organized into directories (aka folders). These files and directories are accessed and manipulated by repeatedly typing commands into the CLI and pressing Enter. Every directory in your computer is defined by a path. The path is just the set of all directories that leads to the specific directory that you are currently in.

Each OS has unique basic commands which you type into the CLI and then press enter to execute. Some important commands are given in the table below:

Table 3.1. Navigation Commands

Command	Function
pwd	Displays the path to your current directory. For example, if it displays, \User\Desktop\School, then you are currently in the School directory, which is in the "Desktop" directory, which is itself in the "User" directory.
ls	Displays the files and directories that are stored in your current directory.
cd <directory>	Moves you to a different directory. If the directory that you wish to go to is in your current directory, then simply type cd NameOfDirectory. Otherwise, you will need to specify the path to the directory by typing something like cd /User/Desktop/School/NameOfDirectory.
open <fileName>.extension	Used to open files. If you wanted to open a file named "data.csv", you would type, "open data.csv".
python <programName>.py	Executes a program that is written in the Python programming language. Python files are designated with the extension, .py. All programming languages have files with their own extensions.

3.1.4 Writing Your First Program

There are many ways to program/write code.

We will provide information on two of the primary ways, using an online notebook and using a text editor.

3.1.5 Online Notebook

For students with minimal experience, we recommend starting off with a notebook. A notebook allows users to input a chunk of code, and then run that chunk. This is great because it allows you to quickly find out whether or not your code works. As opposed to writing an entire program and then finding out it doesn't work. We'll use JupyterHub, which is an open-source web application for programming. All McMaster students can access the phys-ugrad (McMaster's physics server) JupyterHub by:

- downloading Cisco AnyConnect VPN and connecting to sslvpn.mcmaster.ca (Instructions found)
- navigating to the following link, which will open up the McMaster phys-ugrad notebook

Now you can create a new Python file by selecting "New" and then "Python 3". Now type the following line:

print(«Hello World!»)

Then hit Run to execute that line of code. If «Hello World» appears below the line of code, then you have successfully run the line of code! If nothing appears, then you probably made a mistake, which is the best way to become a better programmer. If you can't figure out your error, feel free to post it on the Python Bootcamp course Discussion board on Avenue, or look it up on Google. Programmers spend a lot of time googling issues, it is a natural part of the process, and shouldn't be looked down upon. So Google away!

3.1.6 Text Editor

Once we become comfortable with our coding skills, we will want to write more complicated programs that can be packaged and shared with others. The best way to do this is by writing a program in what is called a text editor. After writing, we save the program as a file with the .py extension, and then run the program in the command line. Lets download a text editor, Sublime Text is free and widely used. It can be dowloaded from the following link (make sure to choose the correct download for your OS):

Let's write and run one of the simplest programs ever, just to get an idea of what the programming process looks like. Once downloaded, open Sublime Text and create a new file. Since we want to code in Python, make sure that the file has the .py extension after the name. Save it to any directory you want, but I recommend the Desktop for easy accessibility. In your empty file, type the following:

print("Hello, World!")

Then save the file. Now open your CLI and navigate to the directory that contains your file. Once there, use the appropriate command from the previous slide to execute the program. If "Hello, World" appears in your CLI window, then you have successfully run the program!

3.1.7 Next Step

Now that you can write and run programs, feel free to go to:

- *Python Basics*: to learn Python syntax and test your knowledge with Avenue quizzes
- *Python For Science*: to explore Python templates that can be used for scientific data analysis

3.2 Getting Started On Windows

3.2.1 Downloading Python

First, we need to download the software for the programming language that we want to code in, the language we'll use is called Python.

Download from the following link: https://www.python.org/downloads/windows/

3.2.2 Opening Your Command Line Interface

In order to use a computer, we need some way of interacting with the computer's functions and services. Luckily, every operating system (Mac, Windows, etc.) comes with user-interfaces. The one that you are probably most familiar with is the graphical user interface (GUI), which allows users to interact with files, folders, programs, etc. using graphical icons (e.g. Finder on Mac and My Computer on Windows). The other kind of user interface is called a command line interface (CLI). This differs from a GUI in that it is text-based, the user enters commands and executes them. The CLI is what programmers use to navigate their computer and run programs, so it is important to become familiar with it. Each operating system's (OS") CLI differs in name and some syntax, but that's about it.

Window's CLI is called Command Prompt, in order to open it, click Start, type "cmd" in the search bar, and press Enter.

3.2.3 Basic Navigation Commands

Data on a computer is kept in files which are organized into directories (aka folders) These files and directories are accessed and manipulated by repeatedly typing commands into the CLI and pressing Enter. Every directory in your computer is defined by a path. The path is just the set of all directories that leads to the specific directory that you are currently in.

Each OS has unique basic commands which you type into the CLI and then press enter to execute. Some important commands are given in the table below:

Table 3.2. Navigation Commands

Command	Function
cd	Displays the path to your current directory. For example, if it displays, \User\Desktop\School, then you are currently in the School directory, which is in the ”Desktop” directory, which is itself in the ”User” directory.
dir	Displays the files and directories that are stored in your current directory.
cd <directory>	Moves you to a different directory. If the directory that you wish to go to is in your current directory, then simply type cd NameOfDirectory. Otherwise, you will need to specify the path to the directory by typing something like cd /User/Desktop/School/NameOfDirectory.
<fileName>.extension	Used to open files. If you wanted to open a file named ”data.csv”, you would type, ”data.csv”.
<programName>.py	Executes a program that is written in the Python programming language. Python files are designated with the extension, .py. All programming languages have files with their own extensions.

3.2.4 Writing Your First Program

There are many ways to program/write code.

We will provide information on two of the primary ways, using a notebook and using a text editor.

3.2.5 Online Notebook

For students with minimal experience, we recommend starting off with a notebook. A notebook allows users to input a chunk of code, and then run that chunk. This is great because it allows you to quickly find out whether or not your code works. As opposed to writing an entire program and then finding out it doesn’t work. We’ll use JupyterHub, which is an open-source web application for programming. All McMaster students can access the phys-ugrad (McMaster’s physics server) JupyterHub by:

- downloading Cisco AnyConnect VPN and connecting to sslvpn.mcmaster.ca (Instructions found)
- navigating to the following link, which will open up the McMaster phys-ugrad notebook

Now you can create a new Python file by selecting “New” and then “Python 3”. Now type the following line:

print(«Hello World!»)

Then hit Run to execute that line of code. If «Hello World» appears below the line of code, then you have successfully run the line of code! If nothing appears, then you probably made a mistake, which is the best way to become a better programmer. If you can’t figure out your error, feel free to post it on the Python Bootcamp course Discussion board on Avenue, or look it up on Google. Programmers spend a lot of time googling issues, it is a natural part of the process, and shouldn’t be looked down upon. So Google away!

3.2.6 Text Editor

Once we become comfortable with our coding skills, we will want to write more complicated programs that can be packaged and shared with others. The best way to do this is by writing a program in what is called a text editor. After writing, we save the program as a file with the .py extension, and then run the program in the command line. Lets download a text editor, Sublime Text is free and widely used. It can be dowloaded from the following link (make sure to choose the correct download for your OS):

Let's write and run one of the simplest programs ever, just to get an idea of what the programming process looks like. Once downloaded, open Sublime Text and create a new file. Since we want to code in Python, make sure that the file has the .py extension after the name. Save it to any directory you want, but I recommend the Desktop for easy accessibility. In your empty file, type the following:

print("Hello, World!")

Then save the file. Now open your CLI and navigate to the directory that contains your file. Once there, use the appropriate command from the previous slide to execute the program. If "Hello, World" appears in your CLI window, then you have successfully run the program!

3.2.7 Next Step

Now that you can write and run programs, feel free to go to:

- *Python Basics*: to learn Python syntax and test your knowledge with Avenue quizzes
- *Python For Science*: to explore Python templates that can be used for scientific data analysis

3.3 Python Basics

At it's most simple, a programming language is a set of instructions that a computer can read, and there are many to choose from. But regardless of the language (C, C++, Python, Java, C, JavaScript, Ruby, Haskell etc.), there are (mostly) always the same fundamental components present, and these are:

1. Data Types and Variables
2. Operators
3. Input/Output
4. Conditionals and Loops

All that learning a programming language comes down to is learning the unique syntax for each of these components. There is a plethora of online material devoted to teaching Python. I have listed some free resources for you to check out below.

3.3.1 Online Literature

- An extensive, but somewhat lengthier option. Chapters 1-8 cover the fundamentals with an average of 6000 words per chapter. Chapter 9-20 go on to cover more intermedite topics. Each chapter includes a set of practice questions.

- A very stripped back and concise option. There are 7 sheets that aim to fit the primary syntax rules for the fundamental Python concepts. Best used as a quick aid while working on practice problem and/or actively programming.

- A moderate length option in a module format. There are 5 modules, although the basics of Python syntax are taught in module 2. The other modules go on to provide practice problems and teach intermediate concepts. Also, it has a focus on STEM applications.

- An outdated but still useful option. The only free content offered is a 12 lesson, interactive course on Python 2, which is a legacy version of Python. This is a good resource if one doesn't mind the constant teasing of PRO material, and the fact that learning Python 2 will eventually require the student to update their knowledge to Python 3 (There isn't a huge difference, but still noticeable).

3.3.2 Online Videos

- An in-depth option with lots of videos. Ep 1-4 cover setting up Python on Windows. Ep 5-33 cover Python fundamentals, which an average episode length of 6 minutes. Later episodes go on to cover intermediate topics. There are two sequel series to this one, so if one enjoys the first series, they have much more similar content to look forward to.
- This is a slightly more streamlined version of the Microsoft videos. Eps 2-9 cover Python fundamentals with videos at an average length of 20 minutes. There are 90+ other videos which explore intermediate topics.
- Another great option. Consists of 60 episodes covered fundamental Python concepts and have an average video length of 10 minutes.
- This website offers a large number of Python courses for beginners, ranging from 4 hours of lectures to 30 minutes. There are also shorter videos offered which go through and answer frequently asked questions regarding Python

3.3.3 Quizzes

After becoming familiar with the Python content, feel free to test your knowledge with Python quizzes on the course Avenue page for the Python bootcamp.

3.3.4 Next Step

After testing your knowledge with the quizzes, feel free to explore why Python is a great tool for science in the last section, *Python For Science*.

3.4 Code Templates

Python is widely-used in the sciences. It has a wide array of external libraries which we can access by importing them into our code (via an import statement). A library is just a package of code written by somebody else, which contains usable functions for us to use. It is these libraries that give Python it's notable versatility. It is with these libraries that we will explore some science-driven computational projects. In this bootcamp, we focus on the libraries:

- **NumPy**
  Provides various numerical computing tools (random number generators, mathematical functions, etc.)
- **Matplotlib**
  Provides various interactive, data visualization tools (histograms, errorbars, scatterplots, curve fitting, etc.)

- **SciPy**
  Provides varies resources for math, science, and engineering

On the Avenue course page for the Python bootcamp, there is a downloadable collection of code templates which guide students through common data analysis tasks, including:

- importing/exporting data from/to a file
- generating random data and plotting it using a histogram
- generating random data and fitting with a polynomial curve
- importing data from a file and fitting with any function
- importing data (with +/- errors) from a file and plotting with errorbars