Python Projects for Beginners

A Ten-Week Bootcamp Approach to Python Programming

Connor P. Milliken

Apress®

Python Projects for Beginners

A Ten-Week Bootcamp Approach to Python Programming

Connor P. Milliken

Apress®

Python Projects for Beginners

Connor P. Milliken
Derry, NH, USA

ISBN-13 (pbk): 978-1-4842-5354-0 ISBN-13 (electronic): 978-1-4842-5355-7
https://doi.org/10.1007/978-1-4842-5355-7

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Nikhil Karkal
Development Editor: Rita Fernando
Coordinating Editor: Divya Modi

Cover designed by eStudioCalamar

Cover image designed by Pixabay

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-5354-0. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

This book is dedicated to my girlfriend Jess.

Ever since we first met, you changed my life forever.

There's so much that I wish to tell you each day,
like how beautiful you are, how you inspire me, or how I would
give anything just to be with you every second of the day.

Your smile lights up my whole world and you make me so
unbelievably happy.

Anytime I have a bad day, I know you'll always be there for me.

I thought that I would only find you in my dreams, but here you are,
standing in front of me, looking beautiful as ever.

From the day I met you, I knew I wanted to give you everything.

You're smart, motivated, beautiful, and resemble all that is
right with this world.

If I only do one thing right in life, I'd like it to be you.

I promise to always push you to be better, always support
you in times of need, and always be there with a Werther's
candy to help you study.

Your dreams have become my dreams, and whatever you want in life,

I want to be there to celebrate and help guide you.

I will always love you, past forever, with all my heart and soul.

So I have only one question left for you…

(turn the page)

Will You Marry Me?

Table of Contents

About the Author

Connor P. Milliken Focused on helping others achieve their goals through education and technology, **Connor P. Milliken** brings a wealth of programming and business experience to his classes.

He graduated with a computer science degree from Daniel Webster College and is pursuing a master's in computer science with a focus in interactive intelligence from Georgia Tech.

Before becoming an instructor at Coding Temple, he was designing simulators in the video game industry for several years. During that time, he took on a vast number of roles from business to programming that he used to release a total of 11 different titles on PC and co-created an award-winning football card game called "Masters of the Gridiron."

Connor has experience in more than seven different languages and three frameworks. He focuses primarily in web development and data analytics using Python. When this book was written, he taught for a coding bootcamp in Boston, MA, where students can learn Python, web development, and data analytics over a 10-week full-time course. He is now a software engineer at Hubspot, Inc. in Cambridge, MA.

Github: *Connor-SM*

About the Technical Reviewer

Bharath Thiruveedula currently works for a major telco service provider. He is core reviewer and key contributor to various OpenStack/ONAP projects. Bharath is passionate about open source technologies and is an evangelist who is focused on making his mark in the Cloud/Container domains. He has been working on distributed systems and machine learning for a significant amount of time.

Acknowledgments

I would like to thank the following people for their generosity and help:

Jessica Boucher, who has been my rock this whole time. Your love and support have continued to help me in all my endeavors. I'm truly blessed to have you in my life.

My family, who have supported and believed in me all my life. Without your guidance, none of this would be possible. To have parents and siblings like you all is nothing short of a miracle and I wouldn't have it any other way.

Clay and Dee Dreslough, who gave me an opportunity and mentored me. This book would not be possible without your guidance over the years. It was at Sports Mogul that I had realized my passion of computer programming, thanks to you both.

Derek Hawkins, who mentored and taught me a lot about teaching, programming, Python, and Ping Pong.

Kirsten Arnold, who created all the art within this book. The work you were able to create from my poor drawing skills was exactly what I had imagined.

Ripal Patel, who helped with the interview portion of Week 9. Your expertise in the hiring and interview process has been wonderful for not only me but the students.

My friends, who over the years have been there for me through it all. Whether it was watching my dog, going on adventures, or just hanging out… thank you. I will always make the drive for you all.

My coaches, who taught me about perseverance, hard work, commitment, and teamwork. Whether it was 6 AM practices or triple sessions in the middle of summer, you've played a big part in my life and for that I'm grateful.

The Coding Temple team, who gave me the opportunity and entrusted me to educate those wanting to pursue a career in tech.

The Apress team, who have helped me throughout this entire process with writing, formatting, reviewing, and more.

My students, who helped to show me why teaching is so rewarding.

CHAPTER 1

Getting Started

Hello there! Welcome to your first step toward becoming a Python developer. Exciting isn't it? Whether you're just beginning to learn how to program, or have experience in other languages, the lessons taught in this book will help to accelerate your goals. As a Python instructor, I can guarantee you that it's not about where you start, it's about how hard you're willing to work.

At the time of writing this book, my daily job is a coding bootcamp instructor where I teach students how to go from zero programming experience to professional developers in just ten weeks. This book was designed with the intent to bring a bootcamp-based approach to text. This book aims to help you learn subjects that are valuable to becoming a professional developer with Python.

Each subsequent chapter will have an overview and a brief description of what we'll cover that week. This week we'll be covering all the necessary basics to get us jump started. Following the age old saying, "*You must learn to walk before you can run,*" we must understand what our tools are and how to use them before we can begin coding.

Overview

- Understanding why and how this book works
- Installing Python and Anaconda
- Understanding how to use these new tools
- Understanding how to use the terminal
- Writing your first Python program

Without further ado, let's get started, shall we?

C. P. Milliken, *Python Projects for Beginners*, https://doi.org/10.1007/978-1-4842-5355-7_1

Monday: Introduction

Almost every programmer remembers that "Aha!" moment, when everything clicked for them. For me that was when I picked up Python. After years of computer science education, one of the best methods I found to learn was by building applications and applying the knowledge you learn. That's why this book will have you coding along rather than reading about the theory behind programming. Python makes it simple to pick up concepts otherwise difficult in other languages. This makes it a great language for breaking into the development industry!

You may have already noticed that the structure of this book is different than most. Instead of chapters, we have each topic separated by weeks or days. Notice the current header for the section. This is part of the bootcamp-based approach, so that you may set goals for each day. There will be two ways to follow along this book:

1. Over the course of ten weeks
2. Over the course of ten days

If you'd like to follow the 10-week approach, then think of each chapter as a weekly goal. All chapters are broken up further into daily segments Monday to Friday. The first four days, Monday through Thursday, will introduce new concepts to understand. Friday, or better known as Project Day, is where we will create a program together based on the lessons learned throughout the week. The focus is that you set aside 30–60 minutes each day to complete each daily task.

If you're eager enough to try the bootcamp style, where you learn all the material in ten days, then think of each chapter as a single day. Granted, you must know that in order to complete this book in ten days, you will need to dedicate around 8 hours per day, which is a typical day for coding bootcamp students. In bootcamps (*like the one I taught*), we go over several concepts daily, and each subsequent day we reiterate the topics learned from previous lessons. This helps to accelerate the process of learning each concept.

What Is Python?

Python is an **interpreted**, **high-level**, **general-purpose** programming language. To understand what each of these descriptions mean, let's make a few comparisons:

- **Low Level vs. High Level**: Refers to whether we program using instructions and data objects at the level of the machine or whether we program using more abstract operations that have been provided by the language designer. Low-level languages (like C, C++) require you to allocate and manage memory, whereas Python manages memory for us.
- **General Purpose vs. Targeted**: Refers to whether the operations of the programming language are widely applicable or are fine-tuned to a domain. For example, SQL is a targeted language that is designed to facilitate extracting information from relational databases, but you wouldn't want to use it to build an operating system.
- **Interpreted vs. Compiled**: Refers to whether the sequence of instructions written by the programmer, called "*source code*," is executed directly (*by an interpreter*) or whether it is first converted (*by a compiler*) into a sequence of machine-level primitive operations. Most applications designed with Python are run through the interpreter, so errors are found at runtime.

Python also emphasizes code readability and uses whitespace to separate snippets of code. We'll learn more about how whitespace in Python works as we get into our lessons, but for now just know that Python is a great first language to break into the computer science industry.

Why Python?

I could go on about why Python is so amazing, but a simple Google search would do that for me. Python is one of the easier languages to learn. Notice I said "*easier*" and not "*easy*"... that's because programming is still difficult, but Python reads closer to the English language than most other languages. This is one of the benefits of learning Python, because concepts that you learn from this book are still applicable to other languages. Python is also one of the most sought-after skills in the technology industry today, used by companies such as Google, Facebook, IBM, etc. It's been used to build applications like Instagram, Pinterest, Dropbox, and much more!

It's also one of the fastest growing languages in 2019, climbing to the top 3 languages to learn for the future.[1] How well does it pay though? According to Indeed.com, the average salary in 2018 was around **$117,000 USD**![2] That's a lot of monopoly money!

One of the biggest reasons for learning Python, though, must be the use of the language itself. It's used in several different industries: front-end development, back-end development, full-stack, testing, data analytics, data science, web design, etc., which makes it a useful language.

Why This Book?

Let's start with the main reason for wanting to read this book. The material taught throughout this book has a proven track record. I've personally used this exact organization approach to help get my students well-paying positions across a variety of industries. The structure of this curriculum has been repeatedly improved over the years to stick with current industry trends.

One of the next great strengths of this book vs. its competitors is how the concepts are taught. I won't bore you with details; instead we'll build small- and large-scale applications together throughout the course of this book. The best way to learn is often by doing! Especially when it comes to programming, one of the lessons I often tell students is to just try writing the code, and if it breaks, fix it. You won't be able to learn if you don't try to break things!

Lastly, this book will not only teach you how to program but how to think like a programmer. At the beginning of each week, I'll challenge you, and by the end of the lesson, you'll be able to understand the approach you need to take. You can always tell the difference between those who are only able to program and those that are proven developers.

Who This Book Is For?

It's always good to understand what you're getting into before you start reading the book. To want to read a book, you first must realize if the book itself is designed for you. If you can answer yes to any of the following questions, then this book is for you:

[1] www.tiobe.com/tiobe-index/

[2] www.indeed.com/salaries/Python-Developer-Salaries

- Do you have experience in other programming languages but want to pick up a high-level language?
- Have you never programmed before but are eager to learn?
- Did you take computer science courses previously, but they just didn't help you learn how to create applications?
- Do you want to make a career change?
- Have you tried to learn languages previously but couldn't because of the difficulty of the language?
- Have you programmed in Python before but want to improve your abilities and learn new tools?

This book is designed for a wide array of readers, no matter your background. The real question is on you, "**How hard are you willing to work**?" The concepts taught in this book can benefit anyone willing to learn. Even if you've programmed in Python before, this book can still help you become a stronger developer.

What You'll Learn

This book was created to be used for bootcamp classes designed in teaching Python. You can expect to cover necessary information that would be required of you on the job as a Python developer. These concepts will give you the ability to go forward with your education in programming. At the end of each chapter, we'll use the concepts covered to create a variety of real-world applications. After all, we're not just focused on Python here, we're trying to build you up to become a better developer.

Tomorrow, we'll find out how to install the necessary software that this book uses. If you already have Anaconda and Python on your machine, you can skip to Wednesday's lesson.

Tuesday: Setting Up Anaconda and Python

Today, we're going to get our software setup. Throughout this book we'll be using a software platform called **Anaconda**, an **integrated development environment (IDE)** called **Jupyter Notebook**, and the language of Python itself. This book will strictly cover Python 3; however, at times you may see me mention subtle differences between versions 2 and 3. Let's go ahead and download and install these first, then I'll get into what each of them are.

Cross-Platform Development

Python runs on all major operating systems, making it a cross-platform language. This means that you can write code on one operating system and work with someone that uses a completely different machine than you. If both machines have Python installed, they should both be able to run the program.

Installing Anaconda and Python for Windows

Most OS X and Linux operating systems already come with Python installed; however, you still need to download Anaconda. For Windows users, Python usually isn't included, but it gets installed with Anaconda. Use the following steps to install Anaconda properly:

1. Open your browser and type `www.anaconda.com/distribution/`.
2. Click the download button in the header (see Figure 1-1).

Figure 1-1. *Anaconda Download Page*

3. Once you are on the next page, make sure you select the proper operating system on the header at the top. Click that button (see Figure 1-2).

Figure 1-2. *Selecting an operating system*

4. Next, click the download button for the Python 3.7 (*or greater*) section (see Figure 1-3).

Figure 1-3. *Downloading Python 3.x version*

5. **This step is strictly for Windows users**... Once the installer fully downloads, go ahead and run it. Use all defaults except for one option. When you get to the page in Figure 1-4, make sure you click the "**add to path**" option. This will let us access Anaconda through our terminal.

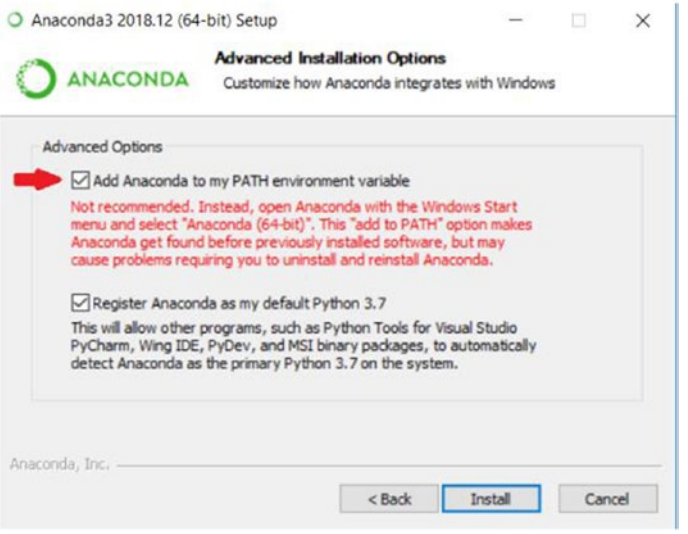

Figure 1-4. *Add to Path*

6. For all options (*besides step 5 for Windows users*), use default settings. Then go ahead and click the "Install" button and let Anaconda finish installing.

What Is Anaconda?

Anaconda is a Python and R distribution software. It aims to provide everything you need for Python "*out of the box*." Its primary use is for data analytics and data science; however, it's a superb tool for learning as well. Upon downloading, it includes

- The core Python language and libraries
- Jupyter Notebook
- Anaconda's own package manager

These are just a few features out of the many that Anaconda comes with; however, these are the ones we'll be using throughout the book. The first feature in this list is the Python language and included packages that Python has access to. Libraries are pre-written code by another developer that you can use for your own benefit. The second feature is talked about in the next section. Lastly, Anaconda has a way of managing environments for us. This is a complex topic that we'll get into in later weeks.

What Is Jupyter Notebook?

It is an open-source **integrated development environment (IDE)** that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. For us, it's essentially our notebook, where we will code along together. If you're not familiar with IDEs, they are simply a tool for developers to code in. Think of them as a canvas for artists. It also allows you to write snippets of code without needing to know a lot about Python. We'll get more into Jupyter Notebook for Thursday's lesson.

In today's lesson, we installed Anaconda, Python, and Jupyter Notebook. Tomorrow, we'll learn why and how to use the terminal.

Wednesday: How to Use the Terminal

Depending on your operating system, you're going to be using the **Command Prompt** (*Windows*) or the **Terminal** (*Linux and OS X*). From this point forward, I'm going to refer to it as the "*terminal*," so just keep that in mind if you're on Windows. The terminal is a tool for users to be able to issue commands to the computer through basic text. For most of this book, we will use the terminal to either test our Python code or run Jupyter Notebook. Today we'll be learning basic commands and how to use the Python shell. To get started, let's open the terminal. As each operating system will look different, terminal sessions will be defined in code by the "**$**". Any text you see after that symbol will be what you need to write into the terminal yourself.

Changing Directories

While inside the terminal, you'll often want to move around from folder to folder. This gives you the power to navigate around your computer. It's important to understand how to do this, as it's always going to be what we do to start up Jupyter Notebook. In order to change directories, you need to type in "*cd*" followed by the folder name you wish to go to.

```
$ cd desktop
```

If you need to go backward, out of a folder, then you'll want to use two dots (".."):

```
$ cd ..
```

Often, throughout this book, you'll need to traverse through several directories to get into a project folder. When you use the "cd" command, you can go as far forward or backward as you select, you just need to specify the correct path to the folder you wish to go to. Take the following code, for instance…

```
$ cd desktop/../desktop
```

We're going into the desktop directory, but then going back out, only to go back into it. There's nothing wrong with this; however, this is just an example that the computer will follow the path that you specify. Normally we would just cd into the desktop and be done.

Checking the Directory

To check the directory that you're currently in, just look to the left of where you can write these lines of text. For Windows users, the directory you're currently in will be the ending URL that you're on, as marked in bold as follows:

```
C:\Users\name\desktop>
```

The last folder name is the "*desktop*," which means that I'm currently in the directory for my desktop. If I were to create any files or folders, they would be created directly on there. To check which directory you're in for Linux, it will be the name just to the left of the "*$*":

```
user@user:~/Desktop$
```

For OS X users, it'll be to the left of your username (*who you're logged in as*):

```
User-Macbook-Pro:Desktop Name$
```

Making Directories

Though it's certainly okay to go into your file explorer, right-click, and select "*create new folder*," it's good to know how to create a new folder through the terminal session itself. Make sure that you're in the "*desktop*" directory that we "*cd*" into previously. Then write the following line:

```
$ mkdir python_bootcamp
```

This will create a new folder called "**python_bootcamp**" on your desktop. We'll be using this folder from here on out to store our lessons so that we stay organized.

Creating Files

Again, it's easier to create files by going into your file explorer. However, sometimes we need to create files in terminal depending on the file type. Before we create a new file, however, let's "*cd*" into our "*python_bootcamp*" folder that we created:

```
$ cd python_bootcamp
```

Now, for **Windows** users, we'll need to type the following:

```
$ echo.>example.txt
```

Or if you're on **Linux/OSX**:

```
$ touch example.txt
```

You should now be able to see the sample.txt file in file explorer.

Note If you don't see the "*.txt*" extension, it's because you don't have "**extensions**" checked in your preferences within file explorer.

Checking a Version Number

The terminal is always a great way to check version numbers of certain software that we download. Since we already downloaded and installed Python, let's run the following code:

```
$ python --version
```

Clearing the Terminal Output

Sometimes the terminal gets full of useless output or just becomes tough to read. When you want to clear the output, you need to write the following line (*for Windows*):

```
$ cls
```

For Linux/OSX users, you'll need to type in the following:

```
$ clear
```