Contents

Foreword

Hello, and welcome to *Python Basics: A Practical Introduction to Python 3*. I hope you're ready to learn why so many professional and hobbyist developers are drawn to Python and how you can begin using it on your own projects, small and large, right away.

This book is targeted at beginners who either know a little programming but not the Python language and ecosystem or are starting fresh with no programming experience whatsoever.

If you don't have a computer science degree, don't worry. David, Dan, Joanna, and Fletcher will guide you through the important computing concepts while teaching you the Python basics and, just as importantly, skipping the unnecessary details at first.

Python Is a Full-Spectrum Language

When learning a new programming language, you don't yet have the experience to judge how well it will serve you in the long run. If you're considering learning Python, let me assure you that this is a good choice. One key reason is that Python is a **full-spectrum** language.

What do I mean by this? Some languages are very good for beginners. They hold your hand and make programming super easy. We can go to the extreme and look at visual languages such as Scratch.

In Scratch, you get blocks that represent programming concepts like variables, loops, method calls, and so on, and you drag and drop them on a visual surface. Scratch may be easy to get started with for sim-

ple programs, but you cannot build professional applications with it. Name one Fortune 500 company that powers its core business logic with Scratch.

Come up empty? Me too, because that would be insanity.

Other languages are incredibly powerful for expert developers. The most popular one in this category is likely C++ and its close relative, C. Whichever web browser you used today was likely written in C or C++. Your operating system running that browser was very likely also built with C/C++. Your favorite first-person shooter or strategy video game? You nailed it: C/C++.

You can do amazing things with these languages, but they are wholly unwelcoming to newcomers looking for a gentle introduction.

You might not have read a lot of C++ code. It can almost make your eyes burn. Here's an example, a real albeit complex one:

```
template <typename T>
_Defer<void(*(PID<T>, void (T::*)(void)))
    (const PID<T>&, void (T::*)(void))>
defer(const PID<T>& pid, void (T::*method)(void))
{
  void (*dispatch)(const PID<T>&, void (T::*)(void)) =
    &process::template dispatch<T>;
  return std::tr1::bind(dispatch, pid, method);
}
```

Please, just no.

Both Scratch and C++ are decidedly *not* what I would call full-spectrum languages. With Scratch, it's easy to start, but you have to switch to a "real" language to build real applications. Conversely, you can build real apps with C++, but there's no gentle on-ramp. You dive headfirst into all the complexity of the language, which exists to support these rich applications.

Python, on the other hand, is special. It *is* a full-spectrum language. We often judge the simplicity of a language based on the `Hello, World` test. That is, what syntax and actions are necessary to get the language to output `Hello, World` to the user? In Python, it couldn't be simpler:

```
print("Hello, World")
```

That's it! However, I find this an unsatisfying test.

The `Hello, World` test is useful but really not enough to show the power or complexity of a language. Let's try another example. Not everything here needs to make total sense—just follow along to get the Zen of it. The book covers these concepts and more as you go through. The next example is certainly something you could write as you get near the end of the book.

Here's the new test: What would it take to write a program that accesses an external website, downloads the content to your app in memory, then displays a subsection of that content to the user? Let's try that experiment using Python 3 with the help of the `requests` package (which needs to be installed—more on that in chapter 12):

```
import requests
resp = requests.get("http://olympus.realpython.org")
html = resp.text
print(html[86:132])
```

Incredibly, that's it! When run, the program outputs something like this:

```
<h2>Please log in to access Mount Olympus:</h2>
```

This is the easy, getting-started side of the Python spectrum. A few trivial lines can unleash incredible power. Because Python has access to so many powerful but well-packaged libraries, such as `requests`, it's often described as *having batteries included*.

So there you have a simple yet powerful starter example. On the real-world side of things, many incredible applications have been written in Python as well.

YouTube, the world's most popular video streaming site, is written in Python and processes more than a million requests per second. Instagram is another example of a Python application. Closer to home, we even have realpython.com and my sites, such as talkpython.fm.

This full-spectrum aspect of Python means that you can start with the basics and adopt more advanced features as your application demands grow.

Python Is Popular

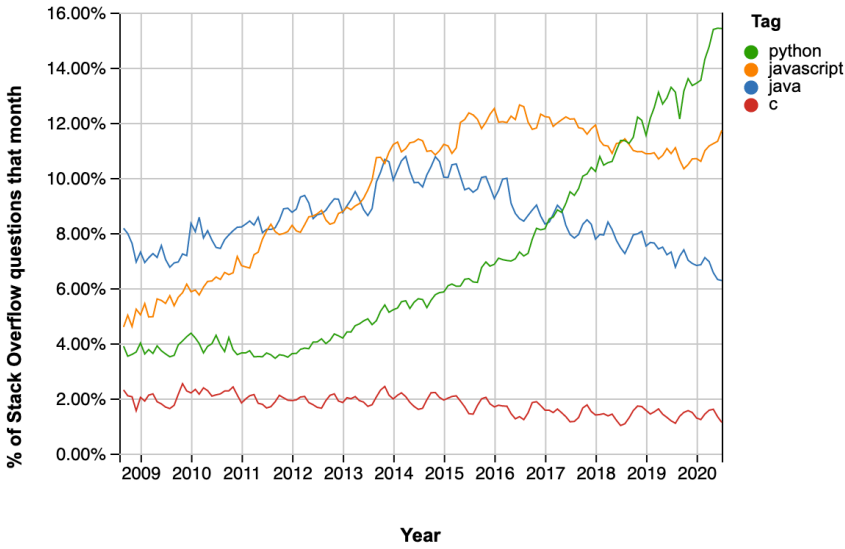
You might have heard that Python is popular. It may seem that it doesn't really matter how popular a language is so long as you can build the app you want to build with it.

But, for better or worse, the popularity of a programming language is a strong indicator of the quality of libraries you'll have available as well the number of job openings you'll find. In short, you should tend to gravitate toward more popular technologies as there will be more choices and integrations available.

So, is Python actually that popular? Yes it is. You'll find a lot of hype and hyperbole, but there are plenty of stats backing this claim. Let's look at some analytics presented by stackoverflow.com, a popular question-and-answer site for programmers.

Stack Overflow runs a site called Stack Overflow Trends where you can look at the trends for various technologies by tag. When you compare

Python to the other likely candidates you could pick to learn programming, you'll see one is unlike the others:



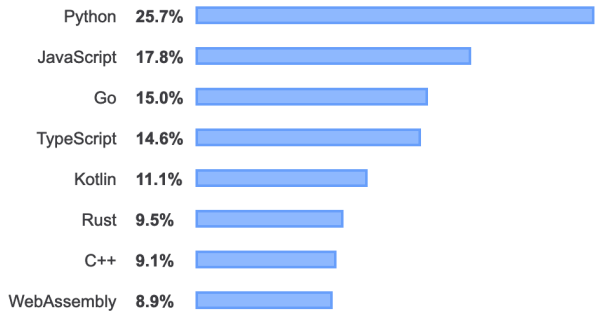
You can explore this chart and create similar charts to this one over at insights.stackoverflow.com/trends.

Notice the incredible growth of Python compared to the flat or even downward trend of the other usual candidates! If you're betting your future on the success of a given technology, which one would you choose from this list?

That's just one chart—what does it really tell us? Well, let's look at another. Stack Overflow does a yearly survey of developers. It's comprehensive and very well done. You can find the full 2020 results at insights.stackoverflow.com/survey/2020.

From that writeup, I'd like to call your attention to a section titled "Most Loved, Dreaded, and Wanted Languages." In the "Most Wanted" section, you'll find data on the share of "developers who are not developing with the language or technology but have expressed interest in developing with it."

Again, in the graph below, you'll see that Python is topping the charts and is well above even second place:



If you agree with me that the relative popularity of a programming language matters, then Python is clearly a good choice.

We Don't Need You to Be a Computer Scientist

One other point that I want to emphasize as you start your Python learning journey is that we don't need you to be a computer scientist. If that's your goal, then great. Learning Python is a powerful step in that direction. But the invitation to learn programming is often framed as "We have all these developer jobs going unfilled! We need software developers!"

That may or may not be true. But, more importantly, programming (even a little programming) can be a personal superpower for you.

To illustrate this idea, suppose you are a biologist. Should you drop out of biology and get a job as a front-end web developer? Probably not. But skills such as the one I opened this foreword with, using requests to get data from the Web, can be incredibly powerful for you as a biologist.

Rather than manually exporting and scraping data from the Web or from spreadsheets, you can use Python to scrape thousands of data sources or spreadsheets in the time it takes you to do just one man-

ually. Python skills can take your *biology power* and amplify it well beyond your colleagues' to make it your *superpower*.

Dan and Real Python

Finally, let me leave you with a comment on your authors. Dan Bader and the other *Real Python* authors work day in and day out to bring clear and powerful explanations of Python concepts to all of us via realpython.com.

They have a unique view into the Python ecosystem and are keyed into what beginners need to know.

I'm confident leaving you in their hands on this Python journey. Go forth and learn this amazing language using this great book. Most importantly, remember to have fun!

— **Michael Kennedy**, Founder of Talk Python (@mkennedy)