Python for Programmers

You already know how to code, but you need to get up to speed - in a practical sense - with the Python programming language

Python for Programmers

Get up to speed - in a practical sense - with the Python programming language

Mark McDonnell

This book is for sale at http://leanpub.com/pythonforprogrammers

This version was published on 2016-09-15

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Also By Mark McDonnell

Programming in Clojure

Contents

Introduction

Welcome to Python for Programmers.

Background

I'm a programmer, have been for many many years now.

I've worked for small agencies, large public sector organisations and commercial companies. I've written apps, services and scripts in lots of different languages:

- Go
- Ruby
- Python
- Clojure
- Node
- Bash

Brief History of Python

Python was first conceived around 1987; development of the language commenced around 1989. The name came about because the author (and still current maintainer) Guido van Rossum was a big fan of the British comedy "Monty Python".

Python is a multi-pardigm programming language, and by that I mean you'll be able to write classical style object-oriented code or implement (although limited) a functional programming style. There's also aspect-oriented programming (achieved via meta-programming) and standard structured programming concepts available too.

The community has seemingly settled on object-oriented programming (OOP) as the natural design choice for the majority of project types, but this is really down to you as the developer to decide.

Python is a general purpose programming language. It wasn't designed for the web or any particular platform. It's primarily used (at the time of writing) for data analysis but, as a general purpose language, is successfully utilised in multiple fields of interest such as building OS command line applications, web services and highly concurrent web applications; pretty much whatever you can think of, you can achieve with Python.

One of the key design choices about Python was to be openly explicit and consistent when writing programming logic (unlike, let's say Ruby, which aims to allow you do things a million different ways and as cryptically as you like).

At first glance Python might almost seem quite verbose if you're coming from a lisp language such as Clojure or Scheme, or another OOP language such as Ruby. But you'll find, like with most programmers who have moved from other said languages, you'll soon come to appreciate this explicitness and consistency.

The Go programming language (designed by Google, also big users of the Python programming language) is a spiritual evolution of Python for the C programming language crowd. Its key benefits are performance and concurrency, but it shares many of the principles of Python, and I believe a lot of inspiration comes from the fact that Google is a big user of Python and so the transition over to Go (and vice versa - which is what I did; coming from Go to Python) is made very easy.

I mention Go, because as a 'dynamic' programming language, Python (like *all* dynamic languages, including Ruby) is considered slow compared to the lower-level languages such as C, C++ or even something like Java. In my opinion Python isn't slow. There are lots of things you can do to ensure your Python programme is performant (and we'll cover those ideas in a later chapter). But if you find yourself in need of a low level 'C-like' language with comparable performance to C, then I highly recommend checking out Go as your transition will be smoother when coming from a language such as Python.

One final aspect of Python that's worth mentioning, and should deter you from jumping ship to Go immediately, is that it's quick to build with; a pleasure to write (developer enjoyment is key); easy to understand; and has a wealth of very high quality open-source packages and libraries (for just about anything) that makes it so much more enjoyable to write software with in comparison to other languages.

I also recommend you check out the Zen of Python[1] which is a list of principles used when writing Python code.

The Pythonic Way

You'll hear a lot of Python developers say "this is not the Pythonic way".

What this means is that the code they're referring to isn't idiomatic Python. That's all. Nothing magical.

To help get you to write code in a 'Pythonic' way, you just need to follow the PEP 8 guidelines: https://www.python.org/dev/peps/pep-0008/[2] which also has additional documentation here[3].

[1] https://en.wikipedia.org/wiki/Zen_of_Python

[2] https://www.python.org/dev/peps/pep-0008/

[3] http://pep8.readthedocs.io/en/latest/index.html

There are also plugins for many different code editors to help notify you when you're writing non-Pythonic code (in the next chapter we'll cover this in more detail by discussing Pylint[4] and Flake8[5], which are tools that most of these code editor plugins utilise behind-the-scenes).

This Book

I make the assumption that you, the reader, are indeed a programmer and that I don't need to explain how to write code. Nor do I have to explain what a class is, or trade-offs between inheritance and composition etc. This isn't that book. This book is for established programmers looking to pick up Python for the first time and just need a point in the right (idiomatic) direction.

If you're a programmer, then it doesn't matter what language it is, you'll know that when it comes time to pick up a new language (whether it be for fun or because you're starting a new job), it's the getting down to the practical information as quickly as possible that is what you want; and that is what I aim to do here in this book.

> Note: this isn't to say a junior programmer won't find lots of benefit too, I just haven't worried myself too much about explaining every little detail

I also don't make any bones about why Python is any better or worse than any other language. I might make some comparisons here and there to other languages when it's appropriate to do so. But it's not my job to fight the good fight for Python or any other programming language. I don't care for flame wars. Certain languages are better suited to certain problems, and in those situations where the better language is unclear, then most of the time people just choose what they prefer.

Hence this book is:

- Short †
- Concise π
- Cheap

I do not intend for it to be anything more.

> † some chapter/sections are literally one page π there are lots of ways to do things, I focus on only one

[4] https://www.pylint.org/
[5] http://flake8.pycqa.org/en/latest/

Prerequisites

I'm using a Mac with OS X 'El Capitan'.

I'll be discussing Python 3.5+

I also use Vim.

Let's get started…