Community Experience Distilled

Python 3
Object Oriented Programming

Harness the power of Python 3 objects

Dusty Phillips

[PACKT] PUBLISHING open source* community experience distilled

Python 3 Object Oriented Programming

Harness the power of Python 3 objects

Dusty Phillips

[PACKT] open source
community experience distilled
PUBLISHING

BIRMINGHAM - MUMBAI

Python 3 Object Oriented Programming

First published: July 2010

Production Reference: 1160710

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 978-1-849511-26-1

www.packtpub.com

Cover Image by Asher Wishkerman (a.wishkerman@mpic.de)

Credits

Author
Dusty Phillips

Reviewers
Jason Chu
Michael Driscoll
Dan McGee
Lawrence Oluyede

Acquisition Editor
Steven Wilding

Development Editor
Mayuri Kokate

Technical Editor
Vanjeet D'souza

Indexer
Hemangini Bari

Editorial Team Leader
Mithun Sehgal

Project Team Leader
Lata Basantani

Project Coordinator
Jovita Pinto

Proofreader
Chris Smith

Graphics
Geetanjali Sawant

Production Coordinator
Shantanu Zagade

Cover Work
Shantanu Zagade

About the Author

Dusty Phillips is a Canadian freelance software developer, teacher, martial artist, and open source aficionado. He is closely affiliated with the Arch Linux community and other open source projects. He maintains the Arch Linux storefronts, and compiled the popular Arch Linux Handbook. Dusty holds a Master's degree in Computer Science specializing in Human-Computer Interaction. He currently has six different Python interpreters installed on his computer.

I would like to thank my editors, Steven Wilding and Mayuri Kokate for well-timed encouragement and feedback. Many thanks to friend and mentor Jason Chu for getting me started in Python and for patiently answering numerous questions on Python, GIT, and life over the years. Thanks to my father, C. C. Phillips, for inspiring me to write while editing his terrific works of fiction. Finally, thanks to every person who has said they can't wait to buy my book; your enthusiasm has been a huge motivational force.

About the Reviewers

Jason Chu is the CTO and part founder of Oprius Software Inc. He's developed software professionally for over 8 years. Chu started using Python in 2003 with version 2.2. When not developing personal or professional software, he spends his time teaching karate, playing go, and having fun in his hometown: Victoria, BC, Canada. You'll often find him out drinking the Back Hand of God Stout at Christie's Carriage House.

Michael Driscoll has been programming Python for almost 4 years and has dabbled in other languages since the late nineties. He graduated from university with a Bachelor's degree in Science, majoring in Management Information Systems. Michael enjoys programming for fun and profit. His hobbies include biblical apologetics, blogging about Python at `http://www.blog.pythonlibrary.org/`, and learning photography. Michael currently works for the local government where he programs with Python as much as possible. This is his first book as a technical reviewer.

I would like to thank my mom without whom I never would have grown to love learning as much as I do. I would also like to thank Scott Williams for forcing me to learn Python as, without him, I wouldn't have even known that the language existed. Most of all, I want to thank Jesus for saving me from myself.

Dan McGee is a software developer currently living in Chicago, Illinois. He has several years of experience working full-time in the Chicago area doing primarily Java web development; however, he has also been spotted working in a variety of other languages. Dan has also worked on a handful of freelance projects. In 2007, Dan became a developer for the Arch Linux distribution and has been doing various projects related to that since, including hacking on the package manager code, being a part-time system admin, and helping maintain and improve the website.

Lawrence Oluyede is a 26 years old software development expert in Python and web programming. He's glad that programming is going parallel and functional languages are becoming mainstream. He has been a co-author and reviewer for the first Ruby book in Italian (*Ruby per applicazioni web*) published by *Apogeo*. He has also contributed to other books in the past like the *Python Cookbook* (`http://www.amazon.com/Python-Cookbook-Alex-Martelli/dp/0596007973/`) and *The Definitive Guide to Django* (`http://www.amazon.com/Definitive-Guide-Django-Development-Right/dp/1590597257`).

Table of Contents

Preface

This book will introduce you to the terminology of the object-oriented paradigm, focusing on object-oriented design with step-by-step examples. It will take you from simple inheritance, one of the most useful tools in the object-oriented programmer's toolbox, all the way through to cooperative inheritance, one of the most complicated. You will be able to raise, handle, define, and manipulate exceptions.

You will be able to integrate the object-oriented and not-so-object-oriented aspects of Python. You will also be able to create maintainable applications by studying higher-level design patterns. You'll learn the complexities of string and file manipulation and how Python distinguishes between binary and textual data. Not one, but two very powerful automated testing systems will be introduced to you. You'll understand the joy of unit testing and just how easy unit tests are to create. You'll even study higher-level libraries such as database connectors and GUI toolkits and how they apply object-oriented principles.

What this book covers

Chapter 1, *Object-oriented Design* covers important object-oriented concepts. It deals mainly with abstraction, classes, encapsulation, and inheritance. We also briefly look into UML to model our classes and objects.

Chapter 2, *Objects in Python* discusses classes and objects and how they are used in Python. We will learn about attributes and behaviors in Python objects, and also the organization of classes into packages and modules. And lastly we shall see how to protect our data.

Chapter 3, *When Objects are Alike* gives us a more in-depth look into inheritance. It covers multiple inheritance and shows us how to inherit from built-ins. This chapter also covers polymorphism and duck typing.

Chapter 4, *Expecting the Unexpected* looks into exceptions and exception handling. We shall learn how to create our own exceptions. It also deals with the use of exceptions for program flow control.

Chapter 5, *When to Use Object-oriented Programming* deals with objects; when to create and use them. We will see how to wrap data using properties, and restricting data access. This chapter also discusses the DRY principle and how not to repeat code.

Chapter 6, *Python Data Structures* covers object-oriented features of data structures. This chapter mainly deals with tuples, dictionaries, lists, and sets. We will also see how to extend built-in objects.

Chapter 7, *Python Object-oriented Shortcuts* as the name suggests, deals with little time-savers in Python. We shall look at many useful built-in functions, then move on to using comprehensions in lists, sets, and dictionaries. We will learn about generators, method overloading, and default arguments. We shall also see how to use functions as objects.

Chapter 8, *Python Design Patterns I* first introduces us to Python design patterns. We shall then see the decorator pattern, observer pattern, strategy pattern, state pattern, singleton pattern, and template pattern. These patterns are discussed with suitable examples and programs implemented in Python.

Chapter 9, *Python Design Patterns II* picks up where the previous chapter left us. We shall see the adapter pattern, facade pattern, flyweight pattern, command pattern, abstract pattern, and composite pattern with suitable examples in Python.

Chapter 10, *Files and Strings* looks at strings and string formatting. Bytes and byte arrays are also discussed. We shall also look at files, and how to write and read data to and from files. We shall look at ways to store and pickle objects, and finally the chapter discusses serializing objects.

Chapter 11, *Testing Object-oriented Programs* opens with the use of testing and why testing is so important. It focuses on test-driven development. We shall see how to use the `unittest` module, and also the `py.test` automated testing suite. Lastly we shall cover code coverage using `coverage.py`.

Chapter 12, *Common Python 3 Libraries* concentrates on libraries and their utilization in application building. We shall build databases using SQLAlchemy, and user interfaces TkInter and PyQt. The chapter goes on to discuss how to construct XML documents and we shall see how to use ElementTree and lxml. Lastly we will use CherryPy and Jinja to create a web application.

What you need for this book

In order to compile and run the examples mentioned in this book you require the following software:

- Python version 3.0 or higher
- py.test
- coverage.py
- SQLAlchemy
- pygame
- PyQt
- CherryPy
- lxml

Who this book is for

If you're new to object-oriented programming techniques, or if you have basic Python skills, and wish to learn in depth how and when to correctly apply object-oriented programming in Python, this is the book for you.

If you are an object-oriented programmer for other languages you will also find this book a useful introduction to Python, as it uses terminology you are already familiar with.

Python 2 programmers seeking a leg up in the new world of Python 3 will also find the book beneficial but you need not necessarily know Python 2.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "We can access other Python modules through the use of the `import` statement."

A block of code is set as follows:

```
class Friend(Contact):
    def __init__(self, name, email, phone):
        self.name = name
        self.email = email
        self.phone = phone
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
class Friend(Contact):
    def __init__(self, name, email, phone):
        self.name = name
        self.email = email
        self.phone = phone
```

Any command-line input or output is written as follows:

```
>>> e = EmailableContact("John Smith", "jsmith@example.net")
>>> Contact.all_contacts
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "We use this feature to update the label to a new random value every time we click the **Roll!** button".

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on `www.packtpub.com` or e-mail `suggest@packtpub.com`.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you get the most from your purchase.

Downloading the example code for this book

You can download the example code files for all Packt books you have purchased from your account at `http://www.PacktPub.com`. If you purchased this book elsewhere, you can visit `http://www.PacktPub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **let us know** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Object-oriented Design

In software development, design is often considered the step done **before** programming. This isn't true; in reality, analysis, programming, and design tend to overlap, combine, and interweave. In this chapter, we will learn:

- What object-oriented means
- The difference between object-oriented design and object-oriented programming
- The basic principles of object-oriented design
- Basic Unified Modeling Language and when it isn't evil

Object-oriented?

Everyone knows what an object is: a tangible "something" that we can sense, feel, and manipulate. The earliest objects we interact with are typically baby toys. Wooden blocks, plastic shapes, and over-sized puzzle pieces are common first objects. Babies learn quickly that certain objects do certain things. Triangles fit in triangle-shaped holes. Bells ring, buttons press, and levers pull.

The definition of an object in software development is not so very different. Objects are not typically *tangible somethings* that you can pick up, sense, or feel, but they are models of *somethings* that can do certain things and have certain things done to them. Formally, an object is a collection of **data** and associated **behaviors**.

So knowing what an object is, what does it mean to be object-oriented? Oriented simply means directed toward. So object-oriented simply means, "functionally directed toward modeling objects". It is one of many techniques used for modeling complex systems by describing a collection of interacting objects via their data and behavior.