



Mark Summerfield

Foreword by Doug Hellmann,
Senior Developer, DreamHost

Python in Practice

Create Better Programs Using
Concurrency, Libraries, and Patterns

Developer's Library



Python in Practice

Python in Practice: Create Better Programs Using Concurrency, Libraries, and Patterns

Table of Contents

Contents

Foreword

Introduction

Acknowledgments

Chapter 1. Creational Design Patterns in Python

1.1. Abstract Factory Pattern

1.1.1. A Classic Abstract Factory

1.1.2. A More Pythonic Abstract Factory

1.2. Builder Pattern

1.3. Factory Method Pattern

1.4. Prototype Pattern

1.5. Singleton Pattern

Chapter 2. Structural Design Patterns in Python

2.1. Adapter Pattern

2.2. Bridge Pattern

2.3. Composite Pattern

2.3.1. A Classic Composite/Noncomposite Hierarchy

2.3.2. A Single Class for (Non)composites

2.4. Decorator Pattern

2.4.1. Function and Method Decorators

Table of Contents

2.4.2. Class Decorators

2.5. Façade Pattern

2.6. Flyweight Pattern

2.7. Proxy Pattern

Chapter 3. Behavioral Design Patterns in Python

3.1. Chain of Responsibility Pattern

3.1.1. A Conventional Chain

3.1.2. A Coroutine-Based Chain

3.2. Command Pattern

3.3. Interpreter Pattern

3.3.1. Expression Evaluation with eval()

3.3.2. Code Evaluation with exec()

3.3.3. Code Evaluation Using a Subprocess

3.4. Iterator Pattern

3.4.1. Sequence Protocol Iterators

3.4.2. Two-Argument iter() Function Iterators

3.4.3. Iterator Protocol Iterators

3.5. Mediator Pattern

3.5.1. A Conventional Mediator

3.5.2. A Coroutine-Based Mediator

3.6. Memento Pattern

3.7. Observer Pattern

3.8. State Pattern

3.8.1. Using State-Sensitive Methods

3.8.2. Using State-Specific Methods

3.9. Strategy Pattern

3.10. Template Method Pattern

Table of Contents

3.11. Visitor Pattern

3.12. Case Study: An Image Package

3.12.1. The Generic Image Module

3.12.2. An Overview of the Xpm Module

3.12.3. The PNG Wrapper Module

Chapter 4. High-Level Concurrency in Python

4.1. CPU-Bound Concurrency

4.1.1. Using Queues and Multiprocessing

4.1.2. Using Futures and Multiprocessing

4.2. I/O-Bound Concurrency

4.2.1. Using Queues and Threading

4.2.2. Using Futures and Threading

4.3. Case Study: A Concurrent GUI Application

4.3.1. Creating the GUI

4.3.2. The ImageScaleWorker Module

4.3.3. How the GUI Handles Progress

4.3.4. How the GUI Handles Termination

Chapter 5. Extending Python

5.1. Accessing C Libraries with ctypes

5.2. Using Cython

5.2.1. Accessing C Libraries with Cython

5.2.2. Writing Cython Modules for Greater Speed

5.3. Case Study: An Accelerated Image Package

Chapter 6. High-Level Networking in Python

6.1. Writing XML-RPC Applications

6.1.1. A Data Wrapper

6.1.2. Writing XML-RPC Servers

Table of Contents

6.1.3. Writing XML-RPC Clients

6.2. Writing RPyC Applications

6.2.1. A Thread-Safe Data Wrapper

6.2.2. Writing RPyC Servers

6.2.3. Writing RPyC Clients

Chapter 7. Graphical User Interfaces with Python and Tkinter

7.1. Introduction to Tkinter

7.2. Creating Dialogs with Tkinter

7.2.1. Creating a Dialog-Style Application

7.2.2. Creating Application Dialogs

7.3. Creating Main-Window Applications with Tkinter

7.3.1. Creating a Main Window

7.3.2. Creating Menus

7.3.3. Creating a Status Bar with Indicators

Chapter 8. OpenGL 3D Graphics in Python

8.1. A Perspective Scene

8.1.1. Creating a Cylinder with PyOpenGL

8.1.2. Creating a Cylinder with pyglet

8.2. An Orthographic Game

8.2.1. Drawing the Board Scene

8.2.2. Handling Scene Object Selection

8.2.3. Handling User Interaction

Appendix A. Epilogue

Appendix B. Selected Bibliography

Index

Table of Contents