Syllabus For Powerful Python Bootcamp

"You will simply not find another Python class that presents such advanced material in such an easy-to-grasp manner. An absolute master class." - *Eric Kramer, Software Architect*

https://powerfulpython.com/teams/

Overview

Powerful Python Bootcamp is advanced training for software professionals.

It is designed for professional engineering teams who wish to "level up" their speed and skill writing real-world, high-quality software systems in Python.

Powerful Python Bootcamp includes

- Live group mentoring sessions with an experienced Pythonista instructor,
- A student Slack channel for networking and peer learning,
- Extensive realistic coding exercises to quickly ingrain your skills, and
- A proven curriculum which quickly teaches "Top 1%" Python skills.

"Powerful Python's first principle approach really attracted me to it. The idea of mastering is something that was very attractive and something that I wanted. And so it's been very valuable. I really encourage you to consider it." - *Erik Engstrom, Embedded Systems Developer*

About The Curriculum

The Powerful Python curriculum follows the "95/5" rule. Teaching the 5% of advanced Python principles and best practices which give 95% of the results.

The reason: *time*.

For busy professionals, time and energy is at a premium. Powerful Python Bootcamp is designed to fit in your life, with only 5-10 hours per week, and on your schedule.

Bootcamp topics include:

- Pythonic Object-Oriented Programming

- Test-Driven Development and Automated Testing
- Writing scalable Python code, to gracefully handle larger data
- Higher-order Python, leveraging higher-order abstractions
- Code "walkthroughs" of real-world production software
- Module organization and evolution
- Patterns for exceptions and errors
- Logging for application monitoring and rapid debugging
- Dependency management
- Writing command-line programs

Pythonic OOP

Covers the timeless, universal principles of object-oriented programming… and how they're expressed, Pythonically.

- Review: Python's Syntax for Classes
- The Single-Responsibility Principle
- Inheritance
- Refactoring For Inheritance
- Interfaces
- Methods And Inheritance
- Access Control
- Properties
- Properties And Refactoring
- Factories, Class Methods And Static Methods
- The Pythonic Observer Pattern
- Python's Data Classes

Test-Driven Python

Writing automated tests one of the key skills that separates average developers, from the very best engineers on the face of the earth. This is a superpower; do not underestimate the impact this will have on your career.

- The Foundations
- Richer Unit Testing
- Test-Driven Development
- More Assertion Types
- Fixtures

- Unit Vs. Integration Tests
- Subtests
- Mock Objects
- Patching With Mocks
- Mocking Strategies, Pros and Cons

Scaling Python With Generators

In this course, you learn about Python's memory model. And how to write code that is scalable - code that can handle increasing magnitudes of data… gracefully, reliably, robustly.

- Foundations of Generators
- Generator Patterns
- Iteration
- List Comprehensions
- Generator Comprehensions (And Other Comprehensions)
- Passing In Data (Coroutines)
- Overview of AsyncIO

Higher-Order Python

This course came out of digging into the source code of the most important and popular libraries in the Python ecosystem. Libraries like Django, Flask, SQLAlchemy, Pytest, Pandas, Twisted and more.

The result is one of the most profound and thought-provoking Python courses ever created. It's been described by students as "seeing into the Matrix of Python". Where you start being able to sense "below the surface" of Python code, and see what 99% of other Python developers never understand.

- Variable Arguments and Argument Unpacking
- Functions As Objects
- Decorators
- Stateful Decorators and More
- Higher-Order Decorators
- Class-Based Decorators

Code Walkthroughs

A collection of in-depth code review videos… each shining a spotlight on code pulled from REAL professional software, solving complex, real-world, production-grade problems…

Going through line by line, narrating in careful detail not just how it works, but WHY it was

designed that way, and how it fits in the larger application.

- DateInterval class
- Program: lookupemails.py
- DownloadDir class
- EmailAnonymizer class
- A Special Method
- The "Christmas Birthday" Video
- Q&A Session Recordings

Module Organization

The larger your application, the more important organizing its code becomes. And understanding Python's module system comes to the rescue!

Includes a thorough treatment of how modules evolve over time, as requirements change, and as you gain greater clarity during the development process. A critically important topic, that is almost NEVER talked about.

- The structure of simple modules, and how they naturally emerge from your application's development cycle
- Advanced considerations for importing components
- More complex modules, and sub-module structure
- The best ways to evolve your module organization over time

Python Dependency Management

Real Python programs depend on open-source libraries and tools. How do you manage those dependencies for your application, and track their changes over time? How do you keep an unexpected upgrade from breaking your code? Learn the different options, their pros and cons, and how to improve what you do today.

- A full understanding of Python's built-in toolset for dependency management
- Effective, proven strategies for conflicting requirements in different contexts
- Reproducible builds and reliable deployments
- Alternative toolsets: what they add over the built-in tooling, and where they fall short

Command-Line Programs

Want to know how to increase the value of ANY program you write?

Simple: make it controllable from the command line.

This makes your program more automatable. By allowing its input sources and outputs to be

specified… options about its configuration and behavior… and more.

When you do this, your program can be combined by control code to inter-operate with other programs and software systems, in powerful ways you never imagined. This course teaches you how to increase the value of potentially every program you write, from now on.

- The strategy and philosophy of how to maximize the impact of the CLI
- The easiest, most readable and maintainable ways to support required inputs and basic options
- Advanced input strategies for option conversion and validation
- Alternate tools for CLI parsing and processing
- Logging in Python

Talk To A Powerful Python Coach

To find out if Powerful Python is right for your team, book a time to talk with us here:

https://powerfulpython.com/teams/