Advanced Python: Best Practices and Design Patterns

Category: Software Development & Programming | **Vendor:** Various

Duration: 32.00 hours (4 days) **26.0 CPD Hours** **Rating:** ⭐ 4.6 (5,878 reviews)

Course Information

Delivery Format: Instructor Led - Online

Course Overview

Expand upon your fundamental Python programming skills to build reliable and stable applications. In this training course, you learn to implement Gang of Four (GoF) design patterns in order to solve commonly recurring, real-world software design programs, thereby avoiding pitfalls and greatly improving the effectiveness of your programming efforts.

About This Course

Expand upon your fundamental Python programming skills to build reliable and stable applications. In this training course, you learn to implement Gang of Four (GoF) design patterns in order to solve commonly recurring, real-world software design programs, thereby avoiding pitfalls and greatly improving the effectiveness of your programming efforts.

Prerequisites & Entry Requirements

General Prerequisites:

Working knowledge of Python programming to the level of:

- Python Programming Introduction, or at least three to six months of Python programming experience

Learning Outcomes

Upon successful completion of this course, participants will be able to:

- Employ design patterns and best practices in Python applications
- Unit test, debug, and instal Python programs and modules
- Profile program execution and improve performance
- Apply advanced Python programming features for efficient, reliable, maintainable programs

Detailed Course Outline

Object-Oriented Programming in Python

- Extending classes to define subclasses
- Inheriting from multiple superclasses and mix-in classes
- Adding properties to a class
- Defining abstract base classes

Exploring Python Features

Writing "Pythonic" code

- Customising iteration and indexing with "magic" methods
- Modifying code dynamically with monkey patching

Handling Exceptions

- Raising user-defined exceptions

- Reducing code complexity with context managers and the "with" statement

Verifying Code and Unit Testing

Testing best practices

- Developing and running Python unit tests

- Simplifying automated testing with the Nose package

Verifying code behaviour

- Mocking dependent objects with the Mock package

- Asserting correct code behaviour with MagicMock

Detecting Errors and Debugging Techniques

Identifying errors

- Logging messages forauditing and debugging

- Checking your code for potential bugs with Pylint

Debugging Python code

- Extracting error information from exceptions

- Tracing program execution with the PyCharm IDE

Implementing Python Design Patterns

Structural patterns

- Implementing the Decorator pattern using @decorator
- Controlling access to an object with the Proxy pattern

Behavioural patterns

- Utilising the Iterator pattern with Python generators
- Laying out a skeleton algorithm in the Template Method pattern
- Enabling loose coupling between classes with the Observer pattern
-

Interfacing with REST Web Services and Clients

Python REST web services

- Building a REST service
- Generating JSON responses to support Ajax clients

Python REST clients

- Sending REST requests from a Python client

- Consuming JSON and XML response data

Measuring and Improving Application Performance

Measuring Application Execution

- Timing execution of functions with the "timeit" module
- Profiling program execution using "cProfile"
- Manipulating an execution profile interactively with "pstats"

Employing Python language features for performance

- Efficiently applying data structures, including lists, dictionaries and tuples
- Mapping and filtering data sets using comprehensions
- Replacing the standard Python interpreter with PyPy

Installing and Distributing Modules

Managing module versions

- Installing modules from the PyPi repository using "pip"
- Porting code between Python versions

Packaging Python modules and applications

- Establishing isolated Python environments with "virtualenv"

- Building a distribution package with "setuptools"

- Uploading your Python modules to a local repository

Concurrent Execution

Lightweight threads

- Creating and managing multiple threads of control with the Thread class

- Synchronising threads using locks

Heavy-weight processes

- Launching operating system commands as subprocesses

- Synchronising processes with queues

- Parallelising execution using process pools and Executors

Skills You Will Learn:

- Employ design patterns and best practices in Python applications
- Unit test, debug, and instal Python programs and modules
- Profile program execution and improve performance
- Apply advanced Python programming features for efficient, reliable, maintainable programs

Frequently Asked Questions

Q: What delivery options are available for Advanced Python: Best Practices and Design Patterns?

We offer multiple delivery formats:

- Live Instructor-Led Classroom Online (Virtual/Live Online)
- Traditional Instructor-Led Classroom Training (ILT)
- On-site delivery at your offices anywhere in United Kingdom
- Private dedicated courses customized for your team

Q: How many CPD hours does this course provide?

The 4-day Advanced Python: Best Practices and Design Patterns course provides up to 26.0 CPD hours of structured learning. CPD certificates can be provided upon request.

Q: What is the duration of the Advanced Python: Best Practices and Design Patterns training?

The training takes place over 4 day(s), with each day lasting approximately 32.00 hours including breaks for lunch and refreshments.

Q: Do you provide corporate training for Advanced Python: Best Practices and Design Patterns?

Yes, we provide corporate training, dedicated training, and closed classes for Advanced Python: Best Practices and Design Patterns. Training can take place anywhere in United Kingdom including London, Manchester, Birmingham, Edinburgh, or live online allowing teams from across United Kingdom or internationally to attend.

Q: What related terms do people search for?

Popular related searches include: Advanced Python: Best Practices and Design Patterns, Programming, APBPDP

Q: Why choose Nexus Human for Advanced Python: Best Practices and Design Patterns?

Nexus Human is recognized as one of the leading training providers. Our trainers have won multiple awards including:

- Small Firms Best Trainer Award
- National Training Partner of the Year (Ireland) - Multiple Years
- Global Top 30 Instructor Awards (2012, 2019, 2021)
- Tech Excellence Award Nominations
- Learning Performance Institute (LPI) External Training Provider Sponsor 2024

Q: Are there any discount codes available?

Yes! Use discount code **PENPAL5** when booking your Advanced Python: Best Practices and Design Patterns training. Please note that only one discount code can be used per booking and cannot be combined with other special offers.

Nexus Human

Professional Training & Development

Email: info@nexushuman.com

Website: www.nexushuman.com

Phone: +353 1 889 8700 (Ireland) | +44 20 8142 8961 (UK)

© 2026 Nexus Human. All rights reserved. This brochure was generated on 14/11/2026.