

Advanced Python Programming

Taking you to the next level with software engineering in python in just 2 days

FA-0037 · 2 days

Software Development

Introduction

In the fast-evolving landscape of software development, Python has emerged as a versatile and powerful language that caters to a variety of programming needs—from web development to data science and artificial intelligence. As technology progresses and projects grow in complexity, the need for advanced programming skills becomes increasingly essential.

This course is designed to delve into the sophisticated aspects of Python, focusing on design patterns, concurrency, and parallelism. These topics are crucial for writing efficient, scalable, and maintainable code, especially in large-scale applications that require robust architecture and performance optimization.

Understanding and applying design patterns in Python can significantly enhance your problem-solving skills and your ability to communicate solutions effectively. Meanwhile, mastering concurrency and parallelism is essential for optimizing program performance and leveraging the full potential of hardware resources.

This course offers an in-depth exploration of these advanced topics, equipping participants with the tools to tackle modern software challenges adeptly.

Learning outcomes

By the end of this course, participants will be able to:

- Identify and implement various design patterns in Python to solve common software design issues.
- Understand the significance of design patterns in enhancing code modularity and reusability.
- Demonstrate proficiency in managing multiple threads and processes using Python's concurrency libraries.

- Design and develop Python applications that efficiently handle parallel execution to optimize performance.
- Debug and optimize Python code involving complex concurrency issues and race conditions.
- Apply best practices in concurrent and parallel programming to real-world software development projects.

Prerequisites

Participants are expected to have:

- A solid understanding of basic Python programming concepts, including data types, functions, classes, and exception handling.
- Familiarity with the core Python libraries and tools.
- Experience with basic software development principles and practices.
- Prior exposure to issues of software design and architecture, though not mandatory, will be beneficial.

Training outline

1. Design Patterns in Python

Introduction to Design Patterns

1. Importance of design patterns in software engineering
2. Categories of design patterns: Creational, Structural, Behavioral

Creational Patterns

1. Singleton: Ensure a class only has one instance
2. Factory Method: Delegate the instantiation process of an object to subclasses
3. Abstract Factory: Create families of related objects without specifying their concrete classes
4. Builder: Construct complex objects step by step
5. Prototype: Copy existing objects without making code dependent on their classes

Structural Patterns

1. Adapter: Enable objects with incompatible interfaces to collaborate
2. Decorator: Add new functionalities to objects dynamically
3. Facade: Provide a simplified interface to a complex subsystem
4. Composite: Compose objects into tree structures to represent part-whole hierarchies
5. Proxy: Provide a surrogate or placeholder for another object to control access to it

Behavioral Patterns

1. Observer: Allow objects to notify other objects about changes in their state

2. Strategy: Enable selecting an algorithm at runtime
3. Command: Turn a request into a stand-alone object that contains all information about the request
4. State: Alter an object's behavior when its state changes
5. Template Method: Define the skeleton of an algorithm in the superclass but let subclasses override specific steps

2. Concurrency and Parallelism in Python

Understanding Concurrency and Parallelism

1. Differences between concurrency and parallelism
2. Use cases and benefits of each approach

Working with Threads in Python

1. Threading module basics
2. Managing and synchronizing threads
3. Potential pitfalls: deadlocks, race conditions

Advanced Threading Techniques

1. Thread pools
2. Using queues for thread-safe data handling

Multiprocessing in Python

1. Basics of the multiprocessing module
2. Process vs. thread: choosing the right tool
3. Handling process pools

Asynchronous Programming

1. Asyncio library fundamentals
2. Writing asynchronous code with async and await
3. Managing asynchronous tasks and event loops

Performance Considerations

1. Profiling and debugging multi-threaded and multi-process applications
2. Optimizing performance in concurrent Python applications

This course provides a comprehensive approach to understanding and applying advanced Python concepts, ensuring participants are well-prepared to tackle current and future challenges in software development.