To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Object-Oriented Programming with Python and AI Tools

Class Duration

28 hours of live online training delivered over 4-5 days

Student Prerequisites

- Experience with Python programming
- Some experience with object-oriented programming in Python or another language is recommended

Target Audience

Python developers, backend engineers, data and QA automation coders, DevOps toolsmiths, architects, and team leads seeking sharper OOP: master classes and typed interfaces, apply SOLID and patterns, refactor with tests and DI, pass ruff/mypy, debug in VS Code, and use AI to ship cleaner APIs and scalable features.

Description

The Object-Oriented Programming with Python training course is for students with Python experience desiring to learn more about object-oriented programming (OOP) using the Python language. The class starts with coverage of Python classes and their many features. Then proceeds into the core principles and practices of OOP. Then, SOLID and Object-Oriented design patterns are explained, discussed, and applied. Throughout the class, AI tools will be used to learn OOP programming and how to apply to software programming. Finally, students will transition from being programmers to software engineers and use AI tools to do the programming for them, while they plan the coding changes using OOP patterns.

Learning Objectives

- Model a domain with Python classes; validate with unit tests.
- Refactor using encapsulation, inheritance, composition, and polymorphism.
- Define typed interfaces with dataclasses and ABCs/Protocols; pass ruff and mypy.

- Debug object collaborations in VS Code to find and fix defects.
- Spot SOLID violations and add abstractions without breaking tests.
- Implement appropriate design patterns and justify the choice.
- Convert legacy code to OO with dependency injection to reduce coupling.
- Write unit tests and test doubles that pin behavior during refactors.
- Use AI tools to draft code, tests, and refactor plans; critique outputs.
- Deliver a small feature using two patterns and SOLID, with a brief design and test report.

Training Materials

Comprehensive courseware is distributed online at the start of class. Courseware is distributed via GitHub in the form of documentation and extensive code samples. Students practice the topics covered through challenging hands-on lab exercises. All students receive a downloadable MP4 recording of the training.

Software Requirements

Students will need a free, personal GitHub account to access the courseware. Students will need permission to install Python and Visual Studio Code on their computers. Also, students will need permission to install Python Packages and Visual Studio Extensions. If students are unable to configure a local environment, a cloud-based environment can be provided.

Training Topics

Introduction

- What is Object-Oriented Programming?
- What are Patterns?
- What is SOLID?
- How is Python an Object-Oriented Programming Language?

Development Environment

- Configure VS Code for Python development
- Code Reformatting with Ruff
- Linting with Ruff and MyPy
- Debugging Python Scripts with VS Code

Python Classes

- Defining a Class
- Instance and Class Members
- Inheritance
- Multiple Inheritance
- Getter/Setter Properties

Principles and Practical Object-Oriented Programming

- Encapsulation
- Polymorphism
- Inheritance
- Composition
- Shared Variable Context for Functions

SOLID Programming

- Single Responsibility Principle
- Open-Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle

Component Design

- Component Cohesion
- Component Coupling

Creational Design Patterns

- Abstract Factory
- Factory
- Builder
- Prototype
- Singleton

Behavioral Design Patterns

- Chain of Responsibility
- Command
- Interpreter
- Iterator
- Mediator

- Observer
- Strategy
- Memento
- State
- Template Method
- Visitor

Structural Design Patterns

- Adapter
- Bridge
- Composite
- Decorator
- Façade
- Flyweight
- Proxy

AI Tools and OOP Programming

- AI Tools and Programming Patterns
- Refactor Legacy Code to use Patterns
- Transform from One Pattern to Another
- Discover Patterns in Code
- Discuss Pattern Options
- Plan Software Changes with AI Tools and Patterns