

**Object-Oriented Python Programming**

**Course Number:** PYTH-270
**Duration:** 3 days

**Overview**

This Python Programming training course teaches attendees with Python experience how to leverage object-oriented programming (OOP) using the Python language. The class starts with a quick review of Python classes and then dives into the core principles and practices of OOP, including design patterns.

**Prerequisites**

All students must be able to comfortably write Python scripts using basic data types, program structures, and the standard Python library.

**Materials**

All Python training students receive comprehensive courseware.

**Software Needed on Each Student PC**

* Any Windows, Linux, or Mac OS X operating system
* Standard Python 3.x or Anaconda Python 3.x
* Visual Studio Code (other editors may be used)

**Objectives**

* Understand the fundamental concepts of Object-Oriented Programming (OOP)
* Review Python basics on class definitions
* Implement OOP principles and practices in Python
* Explore the principles of SOLID and explore how they impact Python program design
* Apply numerous object-oriented design patterns

**Outline**

* Introduction
* Development Environment (Very Quick Overview)
	+ Configure VS Code for Python development
	+ Code Reformatting with Black
	+ Debugging Python Scripts with VS Code
*
* Getter/Setter Properties
* Quick Class Review
	+ Defining a Class
	+ Instance and Class Members
	+ Inheritance
	+ Multiple Inheritance
* 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
* Overview of Creational Design Patterns
	+ Abstract Factory
	+ Factory
	+ Builder
	+ Prototype
	+ Singleton
* Overview of Behavioral Design Patterns
	+ Chain of Responsibility
	+ Command
	+ Interpreter
	+ Iterator
	+ Mediator
	+ Observer
	+ Strategy
	+ Memento
	+ State
	+ Template Method
	+ Visitor
* Overview of Structural Design Patterns
	+ Adapter
	+ Bridge
	+ Composite
	+ Decorator
	+ Façade
	+ Flyweight
	+ Proxy
* Conclusion