Object-Oriented Programming with Python

Duration

3 days

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 a quick review of Python classes and then proceeds into the core principles and practices of OOP. Then, SOLID and Object-Oriented design patterns are explained, discussed, and applied.

Objectives

- Understand the fundamental concepts of Object-Oriented Programming (OOP)
- Review Python basics on class definitions
- Learn how to 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.

Prerequisites

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

Training Materials

All students receive comprehensive courseware covering all topics in the course. 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.

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.
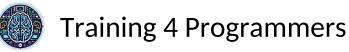
Outline

- Introduction

- Development Environment (Very Quick Overview)
    - Configure VS Code for Python development
    - Code Reformatting with Black
    - Debugging Python Scripts with VS Code
- Quick Class Review
    - 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
- 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