Intermediate Python Programming

Facilitated by Kent State University

Duration:

- 10 Hours
- Synchronous Virtual

Instructor:

- Gregory S. DeLozier, PhD,
- Associate Professor
- Department of Computer Science

COURSE OBJECTIVES

This workshop provides a solid foundation in intermediate Python programming, focusing on data manipulation, object-oriented programming, functional programming, and best practices in debugging and testing. By the end of the course, participants will be able to:

- Apply Object-Oriented Programming (OOP) principles, including classes, objects, inheritance, and polymorphism.
- Write nested and conditional comprehensions to streamline data transformations.
- Utilize counters and sorting techniques effectively to ensure efficient resource management.
- Work with structured data formats using the csv and json modules.
- Implement robust error handling strategies.
- Understand class basics, including advanced instance attributes, private and protected members.
- Differentiate between inheritance and composition, use method overriding, and apply the super() function.
- Customize class behaviors through dunder (magic) methods and other techniques.
- Use functional programming tools like map, filter, reduce, and list comprehensions for data processing.
- Organize code effectively using modules and packages for maintainability.
- Follow best practices in debugging and testing, including unit tests with unittest or pytest.
- Explore concurrency and parallelism, including threading and multiprocessing.
- Work with advanced data structures and efficient file handling techniques.
- Apply generalized functional programming concepts for cleaner, more modular code.
- Utilize modules, error handling, and best practices for writing scalable and maintainable Python applications.

This course is designed for developers who have a foundational knowledge of Python and want to advance their skills for practical applications in a corporate environment.

COURSE REQUIREMENTS

- An account at GitHub
  - Codespaces permitted
- A computer with a Chrome browser

- Optional laptop software:
  - Visual Studio Code
  - Node.js and related tools
  - Python 3.x
- Setup is covered in class

COURSE OUTLINE

Here’s a structured breakdown of 10 one-hour lesson topics that efficiently group related concepts while maintaining a logical progression:

Lesson 1: Advanced Data Structures & Comprehensions

- Lists, tuples, sets, and dictionaries: when to use each
- Nested and conditional comprehensions
- Counters, sorting techniques, and efficient data manipulation

Lesson 2: Working with Structured Data (CSV & JSON)

- Reading and writing CSV files using the csv module
- Parsing and manipulating JSON data with the json module
- Best practices for handling structured data

Lesson 3: Object-Oriented Programming (OOP) Basics

- Classes and objects
- Instance attributes and methods
- Private and protected members (_ and __)

Lesson 4: Advanced OOP: Inheritance & Polymorphism

- Inheritance vs. composition: when to use each
- Method overriding and super()
- Customizing class behaviors with dunder (magic) methods

Lesson 5: Functional Programming in Python

- Introduction to functional programming
- Using map, filter, reduce, and lambda functions
- List comprehensions vs. functional tools

Lesson 6: Organizing Code with Modules & Packages

- Creating and importing modules
- Structuring projects with packages
- Understanding Python’s import system

Lesson 7: Error Handling & Debugging Best Practices

- Using try, except, finally, and else blocks

- Raising and defining custom exceptions
- Debugging tools: pdb, logging, and best practices

Lesson 8: Unit Testing and Code Quality

- Introduction to testing frameworks (unittest and pytest)
- Writing test cases and assertions
- Code quality tools (linting, formatting, and type hints)

Lesson 9: Concurrency & Parallelism

- Understanding threading and multiprocessing
- When to use concurrency vs. parallelism
- Introduction to asyncio for asynchronous programming

Lesson 10: File Handling & Resource Management

- Reading and writing files efficiently
- Using context managers (with statement)
- Managing system resources effectively