Document Generated: 11/03/2026

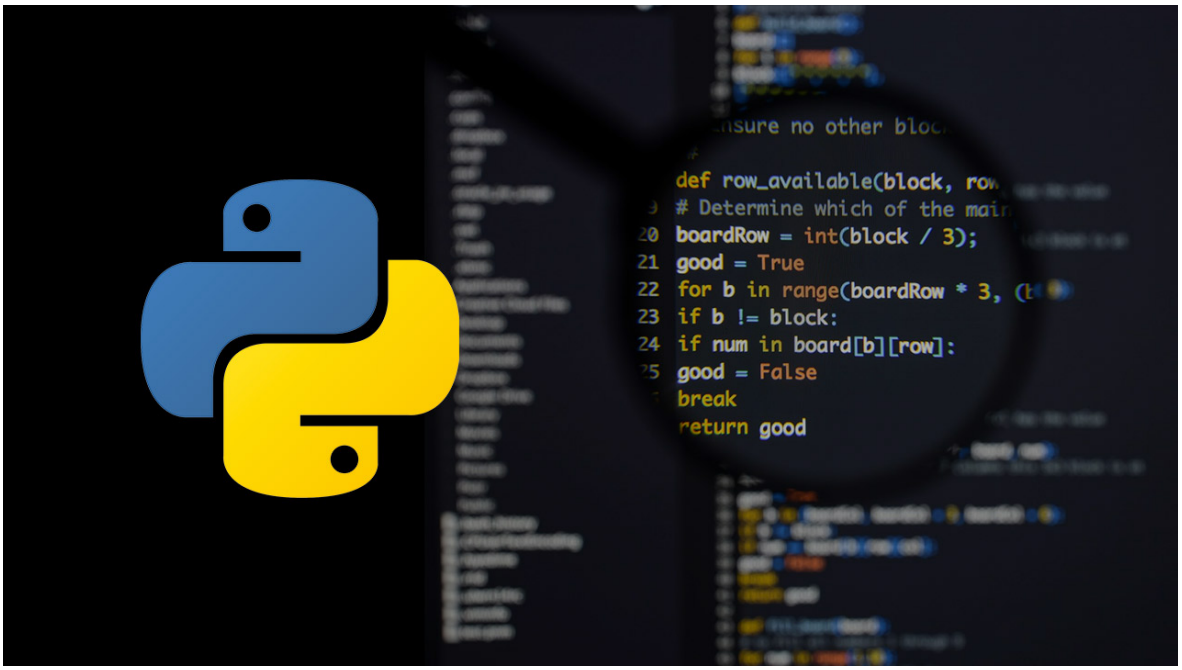
Learning Style: Virtual Classroom

Technology:

Difficulty: Advanced

Course Duration: 4 Days

Advanced Python Programming (TTPS4850)



About This Course:

Advanced Python Programming is a practical, hands-on Python training course that thoroughly explores intermediate to advanced level topics and skills, with a focus on enterprise development. Throughout the course, students will learn how to Leverage OS services, Code graphical interfaces for applications, create modules and run unit tests, define classes, interact with network services, query databases, process XML data, and much more. This comprehensive, practical course provides an in-depth exploration of working with the programming language, not an academic overview of syntax and grammar.

Course Objectives:

This course is approximately 50% hands-on, combining expert lecture, real-world demonstrations and group discussions with machine-based practical labs and exercises. Our engaging instructors and mentors are highly experienced practitioners who bring years of current "on-the-job" experience into every classroom. Create working Python scripts following best practices

- Leverage OS services
- Add enhancements to classes
- Code graphical interfaces for applications
- Understand advanced Python metaprogramming concepts
- Create easy-to-use and easy-to-maintain modules and packages
- Implement and run unit tests
- Create multithreaded and multi-process applications
- Interact with network services
- Design professional scripts
- Query databases
- Process XML, CSV, and JSON data
- Working with more data types *if time permits*
- Using type hints *if time permits*

Audience:

- This is an intermediate and beyond level Python course geared for students experienced with Python who want to use Python in web development projects or automate or simplify common tasks with the use of Python scripts

Prerequisites:

- Basic incoming practical experience working with Python is required, along with a working, user-level knowledge of Unix/Linux, Mac, or Windows. This course does not cover Python fundamentals.

Course Outline:

Python Quick Refresher

- Builtin data types
- Lists and tuples
- Dictionaries and sets
- Program structure
- Files and console I/O
- If statement
- *for* and *while* loops

OS Services

- The **os** and **os.path** modules
- Environment variables
- Launching external commands with **subprocess**
- Walking directory trees
- Paths, directories, and filenames
- Working with file systems

Dates and Times

- Basic date and time classes
- Different time formats
- Converting between formats
- Formatting dates and times
- Parsing date/time information

Binary Data

- What is Binary Data?
- Binary vs text
- Using the Struct module

Pythonic Programming

- The Zen of Python
- Tuples
- Advanced unpacking
- Sorting
- Lambda functions
- List comprehensions
- Generator expressions
- String formatting

Functions, modules, and packages

- Four types of function parameters
- Four levels of name scoping
- Single/multi dispatch

- Relative imports
- Using __init__ effectively
- Documentation best practices

Intermediate classes

- Class/static data and methods
- Inheritance (or composition)
- Abstract base classes
- Implementing protocols (context, iterator, etc.) with special methods

Metaprogramming

- Implicit properties
- globals() and locals()
- Working with object attributes
- The **inspect** module
- Callable classes
- Decorators
- Monkey patching

Developer Tools

- Analyzing programs with **pylint**
- Using the debugger
- Profiling code
- Testing speed with benchmarking

Unit testing with PyTest

- What is a unit test?
- Writing tests
- Working with fixtures
- Test runners
- Mocking resources

Database access

- The DB API
- Available Interfaces
- Connecting to a server
- Creating and executing a cursor
- Fetching data
- Parameterized statements
- Using Metadata
- Transaction control
- ORMs and NoSQL overview

PyQt

- Overview
- Qt Architecture
- Using **designer**
- Standard widgets
- Event handling
- Extras

Network Programming

- Builtin classes
- Using **requests**
- Grabbing web pages
- Sending email
- Working with binary data
- Consuming RESTful services
- Remote access (SSH)

Multiprogramming

- The **threading** module
- Sharing variables
- The **queue** module
- The **multiprocessing** module
- Creating pools
- About async programming

Scripting for System Administration

- Running external programs
- Parsing arguments
- Creating filters to read text files
- Logging

Serializing data – XML and JSON

- Working with XML
- XML modules in Python
- Getting started with **ElementTree**
- Parsing XML
- Updating an XML tree
- Creating a new document
- About JSON
- Reading JSON
- Writing JSON
- Reading/writing CSV files
- YAML, other formats as time permits

Time Permitting Sessions

Advanced data handling

- Discover the **collections** module
- Use **defaultdict**, **Counter**, and **namedtuple**
- Create dataclasses
- Store data offline with **pickle**

Type hinting

- Annotate variables
- Learn what type hinting does NOT do
- Use the **typing** module for detailed type hints
- Understand *union* and *optional* types
- Write stub interfaces

Credly Badge:



Display your Completion Badge And Get The Recognition You Deserve.

Add a completion and readiness badge to your Linkedin profile, Facebook page, or Twitter account to validate your professional and technical expertise. With badges issued and validated by Credly, you can:

- Let anyone verify your completion and achievement by clicking on the badge
- Display your hard work and validate your expertise
- Display each badge's details about specific skills you developed.

Badges are issued by QuickStart and verified through Credly.

Find Out More or See List Of Badges