Advanced Python Programming / Next-Level Python

Duration: *4 Day(s)*

Course Overview

Advanced Python Programming is a four-day, expert-led course designed for professionals with basic Python experience who want to move beyond simple scripts and develop real-world programming expertise. Python is a powerful and versatile language, but to fully harness its potential, you need to understand how to write more efficient, scalable, and maintainable code. This course helps bridge that gap by focusing on practical skills that make your Python development faster, more organized, and better suited for professional applications.

Guided by our expert instructor, you will gain hands-on experience with Python's advanced features, learning how to work with OS services, automate tasks, manage files and directories, and apply best practices in structuring and optimizing your code. These next-level skills are essential for anyone looking to build robust applications, streamline workflows, or handle large-scale data processing. You will learn how to debug and profile code effectively, implement unit tests using PyTest, and interact with databases using SQL and ORM tools. You will also explore network programming, work with APIs, and leverage concurrency techniques like threading, multiprocessing, and asynchronous programming to improve performance. For those interested in application development, the course includes practical instruction on GUI development with PyQt, as well as working with structured data formats like JSON, XML, and YAML.

With 50% of the course dedicated to hands-on exercises, you will not just learn these skills—you will apply them in real scenarios, giving you the confidence and experience to use Python effectively in professional settings.

Review this course online at https://www.alta3.com/courses/TTPS4850

Objectives

- Design efficient and maintainable Python applications by leveraging advanced data structures and best practices.
- Implement effective automation and system operations using Python scripting for tasks including file management and OS services.
- Understand and utilize databases, APIs, and web services integration through SQL and ORM tools in Python.
- Improve performance and scalability in your programs using concurrency techniques like threading and asynchronous programming.

Who Should Attend

- Software Developers
- Data Analysts
- System Administrators
- Engineers

Prerequisites

To get the most out of this expert-led, hands-on course, you should have a basic understanding of Python and some experience writing simple scripts. This course builds on foundational Python skills, so you will be ready to take on more advanced programming concepts.

Here are a few key prerequisites:

Basic Python Syntax and Data Structures – You should be comfortable with variables, loops, conditionals, functions, and common data types like lists, dictionaries, and tuples.

Writing and Running Simple Python Scripts – Experience creating and executing Python scripts, whether for basic automation, simple data processing, or small projects.

Some Exposure to Modules and File Handling – A general understanding of how to import and use Python modules, as well as read and write to files, will be helpful.

Take Before: Advanced Python programming requires incoming basic experience with Python. Each course below teaches basic Python experience, so each course serve as a standalone, solid prerequisite to attend the advanced class. Any ONE of the classes below would apply.

Course Outline

Python refresher

1. Builtin data types
2. Lists and tuples
3. Dictionaries and sets
4. Program structure
5. Files and console I/O
6. If statement
7. for and while loops

OS Services

8. The os and os.path modules
9. Environment variables
10. Launching external commands with subprocess
11. Walking directory trees
12. Paths, directories, and filenames
13. Working with file systems

Dates and Times

14. Basic date and time classes
15. Different time formats
16. Converting between formats
17. Formatting dates and times
18. Parsing date/time information

Binary Data

19. What is Binary Data?
20. Binary vs text
21. Using the Struct module

Pythonic Programming

22. The Zen of Python
23. Tuples
24. Advanced unpacking
25. Sorting
26. Lambda functions
27. List comprehensions
28. Generator expressions
29. String formatting

Functions, modules, and packages

30. Four types of function parameters
31. Four levels of name scoping
32. Single/multi dispatch
33. Relative imports
34. Using init effectively
35. Documentation best practices

Intermediate classes

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

Metaprogramming

40. Implicit properties

41. globals() and locals()
42. Working with object attributes
43. The inspect module
44. Callable classes
45. Decorators
46. Monkey patching

Developer Tools

47. Analyzing programs with pylint
48. Using the debugger
49. Profiling code
50. Testing speed with benchmarking

Unit testing with PyTest

51. What is a unit test?
52. Writing tests
53. Working with fixtures
54. Test runners
55. Mocking resources

Database access

56. The DB API
57. Available Interfaces
58. Connecting to a server
59. Creating and executing a cursor
60. Fetching data
61. Parameterized statements
62. Using Metadata
63. Transaction control
64. ORMs and NoSQL overview

PyQt

65. Overview
66. Qt Architecture
67. Using designer
68. Standard widgets
69. Event handling
70. Extras

Network Programming

71. Builtin classes
72. Using requests
73. Grabbing web pages
74. Sending email
75. Working with binary data
76. Consuming RESTful services
77. Remote access (SSH)

Multiprogramming

78. The threading module
79. Sharing variables
80. The queue module
81. The multiprocessing module
82. Creating pools
83. About async programming

Scripting for System Administration

84. Running external programs
85. Parsing arguments
86. Creating filters to read text files
87. Implementing logging

Serializing data

88. Parsing XML with ElementTree
89. Updating the XML tree
90. Creating new XML documents
91. Reading/Writing JSON data
92. Reading/writing CSV data
93. YAML, other formats

Virtual Environments

94. Use case for
95. Creating an environment
96. Activating and deactivating
97. Replicating an environment
98. Useful tools

Type hinting

99. Annotate variables
100. Learn what type hinting does NOT do
101. Use the typing module for detailed type hints
102. Understand union and optional types
103. Write stub interfaces

Advanced data handling

104. Discover the collections module
105. Use defaultdict, Counter, and namedtuple
106. Create dataclasses
107. Store data offline with pickle