Advanced Python

Duration: 5 Days Course Code: GK821512 Delivery Method: Virtual Learning

Overview:

This course will help you gain an understanding of Python's capabilities beyond basic syntax with a focus on widely accepted Pythonic constructs and procedures that will enable you to write reliable, optimized, and modular applications. This very hands-on course includes a deep dive into Pythonic data structures, exception handling, meta programming, regular expression, advanced file-handling, asynchronous programming, and more. At the completion of the course, you will also gain an understanding of unit testing in Python with lab-based practices designed to help you create and run unit test cases.

Virtual Learning

This interactive training can be taken from any location, your office or home and is delivered by a trainer. This training does not have any delegates in the class with the instructor, since all delegates are virtually connected. Virtual delegates do not travel to this course, Global Knowledge will send you all the information needed before the start of the course and you can test the logins.

Target Audience:

This course is designed for students with Python programming literacy who want to learn about advanced Python features and how to automate and simplify tasks.

Objectives:

- This course has 50% hands-on labs to 50% lecture ratio with engaging instruction, demos, group discussions, labs, and project work in which youll learn:
- Enhancements to classes
- Advanced Python metaprogramming concepts
- Writing robust code using exception handling
- Working with different data structures supported in Python
- Search and replace text with regular expressions
- Easy-to-use and easy-to-maintain modules and packages
- Creating multithreaded and multi-process applications
- Implementing and execute unit tests

Prerequisites:

Students should have experience writing Python scripts, as well as a user-level knowledge of Unix/Linux, Mac, or Windows.

Content:

Day 1

- OOP conventions
- Class/static data and methods
- Parse information to create classes using a dictionary
- Super() method
- Metaclasses
- Abstract base classes
- Implementing protocols (context, iterator, etc.) with special methods
- Implicit properties
- Globals() and locals()
- Working with object attributes
- The inspect module
- Callable classes

Python refresher

- Built-in data types
- Lists and tuples
- Dictionaries and sets
- Program structure
- Files and console I/O
- If statement

for and while loops

Data Structures and Algorithms

- Linked list
- Stack
- Queue
- Trees
- Graphs
- Sorting algorithms

Day 2

- OOP conventions
- Class/static data and methods
- Parse information to create classes using a dictionary
- Super() method
- Metaclasses
- Abstract base classes
- Implementing protocols (context, iterator, etc.) with special methods
- Implicit properties
- Globals() and locals()
- Working with object attributes
- The inspect module
- Callable classes

Errors and Exception Handling

Advanced Functional Features of Python

- Advanced unpacking
- List Comprehension
- Anonymous functions
- Lambda expressions
- Generator Expression
- Decorator
- Closure
- Single/multi dispatch
- Relative imports
- Using __init__ effectively
- Documentation best practices

Day 3

- OOP conventions
- Class/static data and methods
- Parse information to create classes using a dictionary
- Super() method
- Metaclasses
- Abstract base classes
- Implementing protocols (context, iterator, etc.) with special methods
- Implicit properties
- Globals() and locals()
- Working with object attributes
- The inspect module
- Callable classes

Metaprogramming

- OOP conventions
- Class/static data and methods
- Parse information to create classes using a dictionary
- Super() method
- Metaclasses
- Abstract base classes
- Implementing protocols (context, iterator, etc.) with special methods
- Implicit properties
- Globals() and locals()
- Working with object attributes
- The inspect module
- Callable classes

Monkey patching

Advanced file handling

- Paths, directories, and filenames
- Checking for existence
- Permissions and other file attributes
- Walking directory trees

Multiprogramming

- Concurrent programming
- Multithreading
- The threading module
- Sharing variables
- The queue module
- The multiprocessing module
- Creating pools
- Coroutines

About async programming

Python Design Patterns

- Need for design patterns and types
- Creational
- Structural
- Behavioral
- Best coding practices

Day 5

- OOP conventions
- Class/static data and methods
- Parse information to create classes using a dictionary
- Super() method
- Metaclasses
- Abstract base classes
- Implementing protocols (context, iterator, etc.) with special methods
- Implicit properties
- Globals() and locals()
- Working with object attributes
- The inspect module
- Callable classes

Developer Tools

- Analyzing programs with pylint
- Using the debugger
- Profiling code

Testing speed with benchmarking

Unit testing with PyTest

- What is a unit test
- Testing with Unit-test framework
- Testing with PyTest
- Testing with doctest
- Writing tests
- Working with fixtures
- Test runners

Mocking resources

- Syntax errors
- Exceptions
- Using try/catch/else/finally
- Handling multiple exceptions

Ignoring exceptions

Implementing Regular Expressions

- RE Objects
- Searching and matching
- Using Regular Expression to search data sets
- Searching for data in Wireshark Traces (Python and *.pcaps)
- Compilation flags
- Groups and special groups
- Replacing text

Splitting strings

- Creating filters with fileinput
- Using shutil for file operations

Day 4

- OOP conventions
- Class/static data and methods
- Parse information to create classes using a dictionary
- Super() method
- Metaclasses
- Abstract base classes
- Implementing protocols (context, iterator, etc.) with special methods
- Implicit properties
- Globals() and locals()
- Working with object attributes
- The inspect module
- Callable classes

Advanced Data Structure features in Python

- Use defaultdict, Counter, and namedtuple
- Create data classes
- Store data offline with pickle
- Pretty printing data structures
- Compressed archives (zip, gzip, tar, etc.)

Persistent data

Writing real-life applications

- Build the classic minesweeper game in the command line
- Build a program that can go into any folder on your computer and rename all of the files based on the conditions set in your Python code
- Implement the binary search algorithm
- Build a random password generator
- Build a countdown timer using the time Python module.

Further Information:

For More information, or to book your course, please call us on 0800/84.009

info@globalknowledge.be

www.globalknowledge.com/en-be/