Advanced Python – 40 Hours

Course Overview:

Python's power lies in its simplicity - but true mastery comes from understanding its depth. This advanced, hands-on course dives into the language's most sophisticated features, exploring how to leverage them to write cleaner, faster, and more maintainable code. Participants will uncover Python's hidden capabilities, from metaprogramming and decorators to advanced data structures, class inner working, asynchronous programming, and design patterns that take full advantage of Python's dynamic nature.

What Participants Gain?

- Using advanced Excel formulas and conditional logic for efficient data analysis
- Building and customizing Pivot Tables, Pivot Charts, and interactive dashboards
- Performing What-If analysis (Goal Seek, Scenarios, Data Tables)
- Automating repetitive tasks with macros
- Building data models with Power Pivot and writing basic DAX measures
- Cleaning, transforming, and automating data preparation with Power Query

Target Audience

- Data Engineers
- Backend Developers
- BI Developers working with semi-structured data
- DevOps engineers involved in data pipelines
- Technical professionals working with modern data platforms

Prerequisites

- Solid understanding of Python fundamentals, including functions, classes, and object-oriented programming (OOP).
- At least 6–12 months of practical Python experience is recommended.

Course Syllabus:

- Python refresher
  - Comprehensions
  - Functions: variadic, lambda
  - Classes
- Type Hints
  - Python's approach to type safety
  - Basic types
  - Collections
  - Type aliasing
  - Callables
  - Classes
  - Generics
  - Constants and literals
  - Structured dictionaries
  - *args and **kwargs
  - Using Mypy
  - Annotating existing code bases
- More on functions
  - Forcing keyword / positional arguments
  - Closures
  - Functions as objects
  - Standard Library heroes
- Decorators
  - Function based
  - Class based
  - Variadic
  - Nesting
  - Decorating library functions
  - Decorating classes
  - Some useful decorators
- Classes and OOP deep dive
  - Behind the scenes of a class
  - Method bindings
  - Attributes scope
  - Inspecting objects
  - Magic methods
  - Instance, class and static methods
  - Inheritance and multiple-inheritance
    - ❏ The super() function
    - ❏ Method Resolution Order (MRO)
    - ❏ Common pitfalls
  - Duck typing and Polymorphism
  - Abstract classes and Protocols
  - Container protocols

        - ❏ Iterator
        - ❏ Sequence
        - ❏ Mapping
        - ❏ Custom Containers
    - Useful design patterns
        - ❏ Singleton
        - ❏ Prototype
        - ❏ Proxy
        - ❏ Abstract Factory
        - ❏ Decorator
- Data Classes
    - Data Classes vs. plain classes
    - Default values
    - Post-init processing
    - Adding methods
    - Helper functions
- Descriptors basics
    - Descriptors protocol
    - Data and non-data descriptors
    - Common use-cases
- Introduction to meta classes
    - Classes as objects
    - Defining meta classes
    - Common use-cases
- Generators and Coroutines
    - Generator protocol
    - Generator functions
    - Generator classes
    - Generator expressions
    - Generating from other generators
    - Interacting with a generator
    - Data pipelines with generators
- Introduction to Async programming
    - Synchronous vs. asynchronous programming
    - Async, Await and the Event Loop
    - Coroutines and Tasks
    - Error Handling in Async Code