Python for Beginners

Prerequisites: Knowledge of any programming language.

Module 1: Core Python Data Types

- Introducing Python Object Types
- Conceptual hierarchy and built-in object types.
- Core data types: int, str, float, bool
- Mutability and immutability.

Module 2. Data Structures

- List
- Tuple
- Sets
- Dictionary

Module 3. The Dynamic Typing Interlude

- Python's dynamic typing mechanism.
- Shared references and garbage collection.

Module 4. File Handling

- File Opening.
- File Closing.
- Different modes of file handling
- Context Manager

Hands-On Labs:

- Implement basic file handling operations.
- Explore dynamic typing with custom examples.

Module 5: String Manipulation and Structured Data

- String Fundamentals
- String methods, slicing, and formatting techniques.
- Advanced string operations (templates and Unicode).

Module 6: Lists and Dictionaries and Tuples

- List comprehensions, nesting, and operations.
- Dictionary methods, comprehensions, and optimizations.
- Tuples vs. lists.

Module 7: Data Storage and Transmission

- Data serialization with `pickle`
- Data serialization with `json`.
- Data deserialization

Hands-On Labs:

- Perform serialization and Deserialization of data

Module 8: Control Flow and Iterations

- Python Statements
- Understanding Python’s indentation-based syntax.
- Interactive loops and error handling basics.

Module 9: Conditionals, Loops, and Comprehensions

- Writing complex conditional logic with `if`, `elif`, `else`.
- Loop constructs: `while`, `for`, and `else`.
- Iteration protocols and advanced comprehensions.

Hands-On Labs:

- Develop a multi-condition data filter.
- Use list comprehensions to transform nested data.

Module 10: Functions, Scopes, and Generators

- Functions and Arguments
- Function basics: defining and calling.
- Scope and closure concepts.
- Argument-passing techniques, including `*args` and `**kwargs`.

Module 11: Advanced Functions

- Recursive functions and functional programming tools (`map`, `filter`).
- `lambda` expressions and introspection.

Module 12: Generators and Comprehensions

- Generator functions with `yield`.
- Generator expressions and comparisons to list comprehensions.
- Advanced use cases for generators in pipelines.

Hands-On Labs:

- Create a recursive function for hierarchical data traversal.
- Implement a generator pipeline for processing live data streams.

Module 13: Benchmarking and Practical Applications

- Benchmarking Techniques
- Timing operations with `timeit`
- Performance analysis using iteration

Module 14: Practical Applications and Best Practices

- Review of covered concepts applied to real-world scenarios.
- Writing efficient, maintainable Python code.

Hands-On Labs:

- Build a custom benchmarking script for comparing sorting algorithms.

Module 15: Project Work:

- Data aggregator with file handling and JSON output.
- Custom benchmarking utility for Python scripts.