Data Structures and Algorithms in Python

Course Description: This course offers a comprehensive, definitive introduction to data structures and algorithms, including their design, analysis, and implementation in Python. Utilizing a consistent object-oriented viewpoint throughout the book, it provides detailed algorithmic strategies for producing efficient realizations of common data structures such as arrays, stacks, queues, linked lists, trees, maps, hash tables, search trees, and graphs. This course also provides an in-depth analysis of algorithmic performance that helps readers to recognize common trade-offs between competing strategies. This course incorporates a host of pedagogical features, including illustrations, code fragments, and end-of-Module exercises.

Duration: 5 Days

What you will learn from this course

- **Module 1 Introduction to Python**
  - Python Overview
  - Objects in Python
  - Expressions, Operators, and Precedence
  - Control Flow
  - Functions
  - Simple Input and Output
  - Exception Handling
  - Iterators and Generators
  - Additional Python Conveniences
  - Scopes and Namespaces
  - Modules and the Import Statement

- **Module 2 Object-Oriented Programming**
  - Goals, Principles, and Patterns
  - Software Development
  - Class Definitions
  - Inheritance
  - Namespaces and Object-Orientation
  - Shallow and Deep Copying

- **Module 3 Introduction to Data Structures and Algorithms**
  - Data Structures
  - Experimental Studies
  - The Seven Functions Used in this Book
  - Analysis of Algorithms
  - Simple Justification Techniques

- **Module 4 Recursion**
  - Examples Illustrating Recursion
  - Analysing Recursive Algorithms

- Further Examples of Recursion
- Designing Recursive Algorithms
- Eliminating Tail Recursion

- **Module 5 Array-Based Sequences**
  - Python's Sequence Types
  - Low-Level Arrays
  - Dynamic Arrays and Amortization
  - Efficiency of Python's Sequence Types
  - Using Array-Based Sequences
  - Multidimensional Data Sets

- **Module 6 Stacks**
  - Stacks
  - The Stack Abstract Data Type
  - Simple Array-Based Stack Implementation

- **Module 7 Queues**
  - Queues
  - The Queue Abstract Data Type
  - Array-Based Queue Implementation
  - Double-Ended Queues
  - Circular Queues

- **Module 8 Linked Lists**
  - Singly Linked Lists
  - Circularly Linked Lists
  - Doubly Linked Lists
  - The Positional List ADT
  - Sorting a Positional List
  - Link-Based vs. Array-Based Sequences

- **Module 9 Trees**
  - General Trees
  - Binary Trees
  - Implementing Trees
  - Tree Traversal Algorithms
  - Case Study: An Expression Tree

- **Module 10 Priority Queues**
  - The Priority Queue Abstract Data Type
  - Implementing a Priority Queue
  - Heaps
  - Adaptable Priority Queues

- **Module 11 Maps, Hash Tables, and Sets**

- Maps and Dictionaries
- Hash Tables
- Sorted Maps
- Sets, Multisets, and Multimaps

- **Module 12 Search Trees**
  - Binary Search Trees
  - Balanced Search Trees
  - AVL Trees
  - Splay Trees
  - (2, 4) Trees
  - Red-Black Trees

- **Module 13 Sorting Algorithms**
  - Why Study Sorting Algorithms?
  - Merge-Sort
  - Quick-Sort
  - Studying Sorting through an Algorithmic Lens
  - Sorting with a Priority Queue
  - Comparing Sorting Algorithms
  - Python's Built-In Sorting Functions

- **Module 14 Graph Algorithms**
  - Graphs
  - Data Structures for Graphs
  - Graph Traversals
  - Transitive Closure
  - Directed Acyclic Graphs
  - Shortest Paths
  - Minimum Spanning Trees

- **Module 15 Text Processing**
  - Abundance of Digitized Text
  - Pattern-Matching Algorithms
  - Dynamic Programming
  - Text Compression and the Greedy Method
  - Tries

- **Module 16 Memory Management and B-Trees**
  - Memory Management
  - Memory Hierarchies and Caching
  - External Searching and B-Trees
  - External-Memory Sorting