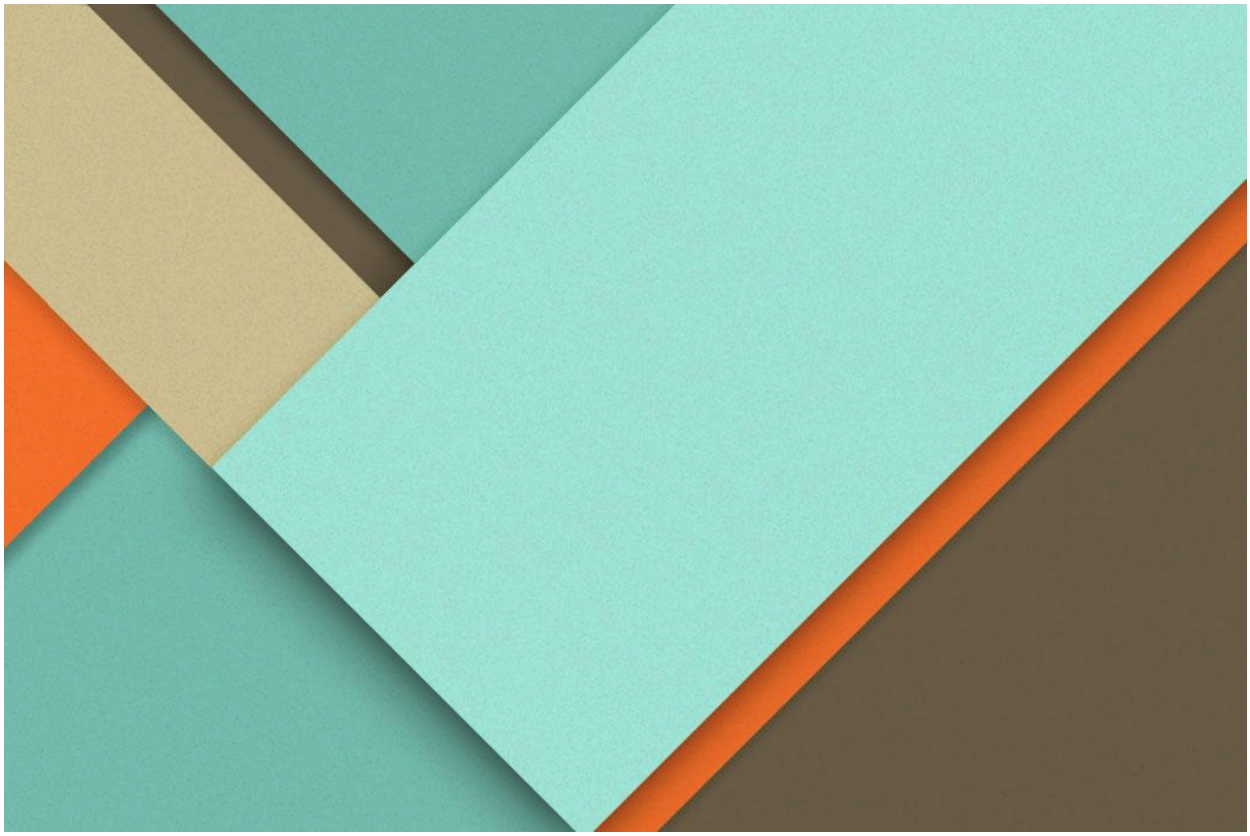

Data Structures and Algorithms in Python

05.21.2024

—

Himanshu Bhadra
BayAreaLa8s | www.bayareala8s.com | https://github.com/bayareala8s
186 Cameo Drive,
Livermore, CA 94550

Course Overview

This course provides a comprehensive understanding of data structures and algorithms using Python. It covers fundamental concepts, implementation techniques, and problem-solving strategies, preparing students to tackle complex computational problems efficiently.

Course Objectives

- Understand and implement core data structures in Python.
- Analyze the performance of algorithms.
- Develop problem-solving skills using algorithms and data structures.
- Apply data structures and algorithms in practical scenarios.

Prerequisites

- Basic knowledge of Python programming.
- Understanding of basic programming concepts and constructs.

Course Duration

Total: 24 hours (1440 minutes)

Module 1: Introduction to Data Structures and Algorithms

Duration: 60 minutes

- Lecture Topics:
  - Course overview and objectives (10 mins)
  - Importance of data structures and algorithms (15 mins)
  - Python refresher: syntax, functions, and classes (20 mins)
  - Complexity analysis: Big O notation (15 mins)

Module 2: Arrays and Linked Lists

Duration: 120 minutes

- Lecture Topics:
  - Arrays: definition, operations, and applications (30 mins)

- Linked Lists: singly, doubly, and circular linked lists (45 mins)
- Implementation and complexity analysis (45 mins)

Lab Activities:
- Implementing arrays and basic operations (30 mins)
- Creating linked lists and performing insertion, deletion, and traversal (60 mins)

Module 3: Stacks and Queues

Duration: 120 minutes
- Lecture Topics:
  - Stacks: LIFO principle, operations, and applications (30 mins)
  - Queues: FIFO principle, operations, and applications (30 mins)
  - Variants: circular queue, priority queue, and deque (30 mins)

Lab Activities:
- Implementing stacks and queues using lists and linked lists (30 mins)
- Solving problems using stacks and queues (e.g., balanced parentheses) (60 mins)

Module 4: Recursion and Backtracking

Duration: 120 minutes
- Lecture Topics:
  - Understanding recursion: base case and recursive case (30 mins)
  - Examples: factorial, Fibonacci series, and permutations (30 mins)
  - Backtracking: concept and examples (30 mins)

Lab Activities:
- Writing recursive functions (30 mins)
- Solving problems using recursion and backtracking (e.g., N-Queens problem) (60 mins)

Module 5: Trees - Part 1

Duration: 120 minutes
- Lecture Topics:

- Introduction to trees: terminology and properties (30 mins)
- Binary Trees: structure, traversal (pre-order, in-order, post-order) (45 mins)
- Binary Search Trees (BST): operations and applications (45 mins)

Lab Activities:

- Implementing binary trees and traversal algorithms (60 mins)
- Building and manipulating binary search trees (60 mins)

Module 6: Trees - Part 2

Duration: 120 minutes

- Lecture Topics:
  - Balanced Trees: AVL trees and rotations (30 mins)
  - Heaps: binary heap, heap operations, and heap sort (45 mins)
  - Trie: structure and applications (45 mins)

Lab Activities:

- Implementing AVL trees and performing rotations (60 mins)
- Creating heaps and implementing heap sort (30 mins)
- Building and using a trie for string operations (30 mins)

Module 7: Graphs - Part 1

Duration: 120 minutes

- Lecture Topics:
  - Graph terminology and representations (adjacency matrix, adjacency list) (30 mins)
  - Graph traversal algorithms: BFS and DFS (45 mins)
  - Applications of graph traversal (45 mins)

Lab Activities:

- Representing graphs using different methods (30 mins)
- Implementing BFS and DFS (90 mins)

Module 8: Graphs - Part 2

Duration: 120 minutes
- Lecture Topics:
  - Shortest path algorithms: Dijkstra's and Bellman-Ford (45 mins)
  - Minimum spanning tree: Kruskal's and Prim's algorithms (45 mins)
  - Topological sorting and applications (30 mins)

Lab Activities:
  - Implementing shortest path algorithms (45 mins)
  - Creating minimum spanning trees (45 mins)
  - Performing topological sorting (30 mins)

Module 9: Sorting and Searching Algorithms

Duration: 120 minutes
- Lecture Topics:
  - Sorting algorithms: bubble sort, selection sort, insertion sort, merge sort, quick sort (60 mins)
  - Searching algorithms: linear search, binary search (30 mins)
  - Algorithm analysis and comparison (30 mins)

Lab Activities:
  - Implementing and comparing different sorting algorithms (60 mins)
  - Implementing searching algorithms and analyzing their performance (60 mins)

Module 10: Hashing

Duration: 120 minutes
- Lecture Topics:
  - Hash tables: concept, hashing functions, and collision resolution (45 mins)
  - Applications of hashing (45 mins)
  - Performance analysis (30 mins)

Lab Activities:
  - Implementing hash tables with collision resolution techniques (60 mins)
  - Solving problems using hash tables (60 mins)

Module 11: Advanced Topics

Duration: 120 minutes

- Lecture Topics:
  - Dynamic programming: principles and examples (e.g., knapsack problem, longest common subsequence) (60 mins)
  - Greedy algorithms: principles and examples (e.g., activity selection, Huffman coding) (60 mins)

Lab Activities:

- Solving problems using dynamic programming (60 mins)
- Implementing greedy algorithms for various problems (60 mins)

Module 12: Review and Final Project

Duration: 180 minutes

- Lecture Topics:
  - Review of key concepts and techniques (30 mins)
  - Discussion of potential project ideas and real-world applications (30 mins)
  - Final project guidelines and expectations (30 mins)

Lab Activities:

- Work on final projects: applying data structures and algorithms to solve a comprehensive problem (90 mins)
- Presentations and peer reviews of final projects (90 mins)

Assessments

- Weekly quizzes and assignments.
- Mid-term exam covering Weeks 1-6.
- Final exam covering Weeks 7-11.
- Final project presentation and report.

This course structure provides a balanced mix of theoretical understanding and practical implementation, equipping students with the skills needed to excel in computer science and software development roles.