**ASSIGNMENT-11.1**

**SK JAVID AMAN**

**2503A52L17**

**Lab 11-Data Structures with AI: Implementing Fundamental Structures.**

**Lab Objectives:**

**• Use AI to assist in designing and implementing fundamental data**

**structures in Python.**

**• Learn how to prompt AI for structure creation, optimization, and**

**documentation.**

**• Improve understanding of Lists, Stacks, Queues, Linked Lists, Trees,**

**Graphs, and Hash Tables.**

**• Enhance code quality with AI-generated comments and performance**

**suggestions.**

**Task Description #1 – Stack Implementation**

**Task: Use AI to generate a Stack class with push, pop, peek, and is\_empty**

**methods.**

**Sample Input Code:**

**class Stack:**

**pass**

A computer screen shot of a program code

AI-generated content may be incorrect.

**Expected Output:**

**• A functional stack implementation with all required methods and**

**docstrings.**

A screenshot of a computer screen

AI-generated content may be incorrect.

**Task Description #2 – Queue Implementation**

**Task: Use AI to implement a Queue using Python lists.**

**Sample Input Code:**

**class Queue:**

**pass**

A computer screen shot of a program

AI-generated content may be incorrect.

**Expected Output:**

**• FIFO-based queue class with enqueue, dequeue, peek, and size**

**methods.**

A screenshot of a computer

AI-generated content may be incorrect.

**Task Description #3 – Linked List**

**Task: Use AI to generate a Singly Linked List with insert and display methods.**

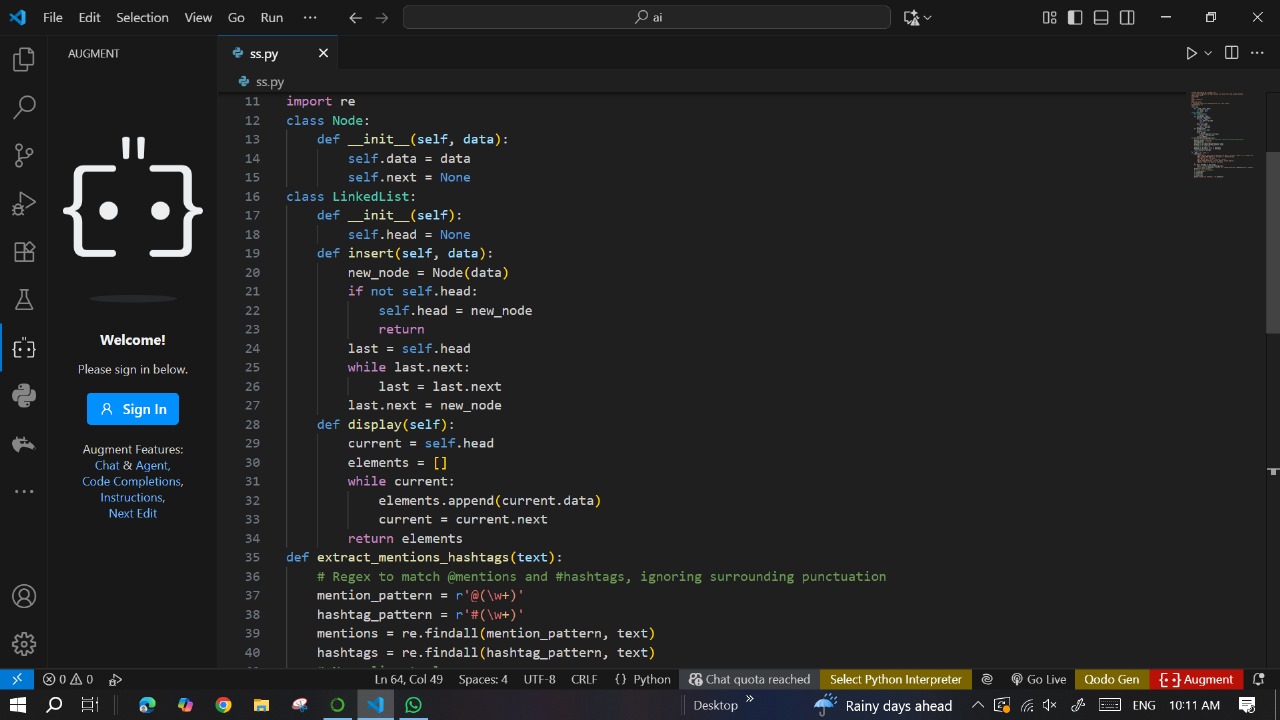
**Sample Input Code:**

**class Node:**

**pass**

**class LinkedList:**

**pass**



A screen shot of a computer

AI-generated content may be incorrect.

**Expected Output:**

**• A working linked list implementation with clear method**

**documentation.**

A screenshot of a computer

AI-generated content may be incorrect.

**Task Description #4 – Binary Search Tree (BST)**

**Task: Use AI to create a BST with insert and in-order traversal methods.**

**Sample Input Code:**

**class BST:**

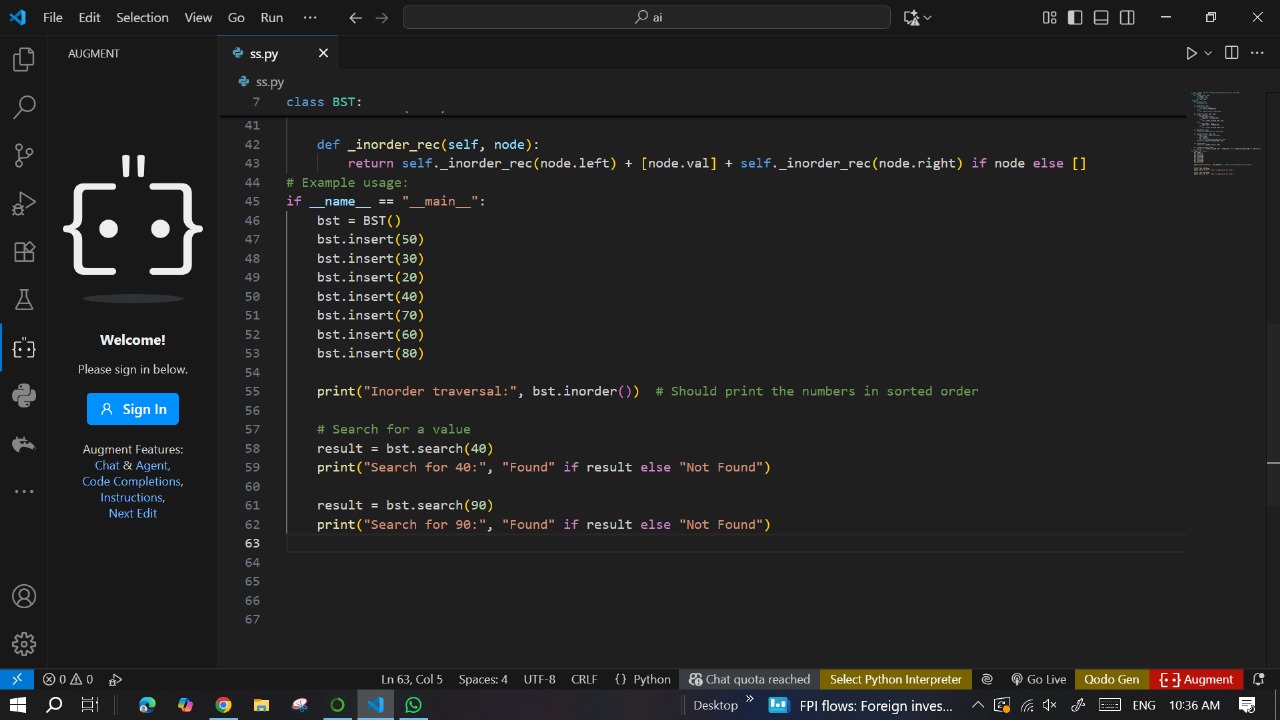
**pass**

A computer screen with text on it

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.



**Expected Output:**

**• BST implementation with recursive insert and traversal methods.**

A screenshot of a computer

AI-generated content may be incorrect.

**Task Description #5 – Hash Table**

**Task: Use AI to implement a hash table with basic insert, search, and delete**

**methods.**

**Sample Input Code:**

**class HashTable:**

**pass**

A computer screen shot of a program code

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

**Expected Output:**

**• Collision handling using chaining, with well-commented methods.**

A screenshot of a computer

AI-generated content may be incorrect.

**Task Description #6 – Graph Representation**

**Task: Use AI to implement a graph using an adjacency list.**

**Sample Input Code:**

**class Graph:**

**pass**

A computer screen with text on it

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Expected Output:**

**• Graph with methods to add vertices, add edges, and display**

**connections.**

A screenshot of a computer

AI-generated content may be incorrect.

**Task Description #7 – Priority Queue**

**Task: Use AI to implement a priority queue using Python’s heapq module.**

**Sample Input Code:**

**class PriorityQueue:**

**pass**

A computer screen shot of a program

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

**Expected Output:**

**• Implementation with enqueue (priority), dequeue (highest priority), and**

**display methods.**

A screenshot of a computer

AI-generated content may be incorrect.

**Task Description #8 – Deque**

**Task: Use AI to implement a double-ended queue using collections.deque.**

**Sample Input Code:**

**class DequeDS:**

**pass**

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

**Expected Output:**

**• Insert and remove from both ends with docstrings.**

A screenshot of a computer

AI-generated content may be incorrect.

**Task Description #9 – AI-Generated Data Structure Comparisons**

**Task: Use AI to generate a comparison table of different data structures (stack,**

**queue, linked list, etc.) including time complexities.**

**Sample Input Code:**

**# No code, prompt AI for a data structure comparison table.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Expected Output:**

**• A markdown table with structure names, operations, and complexities.**

**A computer screen shot of a computer program

AI-generated content may be incorrect.**

**Task Description #10 Real-Time Application Challenge – Choose the**

**Right Data Structure**

**Scenario:**

**Your college wants to develop a Campus Resource Management System that**

**handles:**

**1. Student Attendance Tracking – Daily log of students entering/exiting**

**the campus.**

**2. Event Registration System – Manage participants in events with quick**

**search and removal.**

**3. Library Book Borrowing – Keep track of available books and their due**

**dates.**

**4. Bus Scheduling System – Maintain bus routes and stop connections.**

**5. Cafeteria Order Queue – Serve students in the order they arrive.**

**Student Task:**

**• For each feature, select the most appropriate data structure from the list**

**below:**

**o Stack**

**o Queue**

**o Priority Queue**

**o Linked List**

**o Binary Search Tree (BST)**

**o Graph**

**o Hash Table**

**o Deque**

**• Justify your choice in 2–3 sentences per feature.**

**• Implement one selected feature as a working Python program with AI-**

**assisted code generation.**

**A computer screen shot of a computer program

AI-generated content may be incorrect.**

**Expected Output:**

**• A table mapping feature → chosen data structure → justification.**

**• A functional Python program implementing the chosen feature with**

**comments and docstrings.**

**Deliverables (For All Tasks)**

**1. AI-generated prompts for code and test case generation.**

**2. At least 3 assert test cases for each task.**

**3. AI-generated initial code and execution screenshots.**

**4. Analysis of whether code passes all tests.**

**5. Improved final version with inline comments and explanation.**

**6. Compiled report (Word/PDF) with prompts, test cases, asserti**ons**, code, and output.**

**A screenshot of a computer program

AI-generated content may be incorrect.**