

ASSIGNMENT 11.3

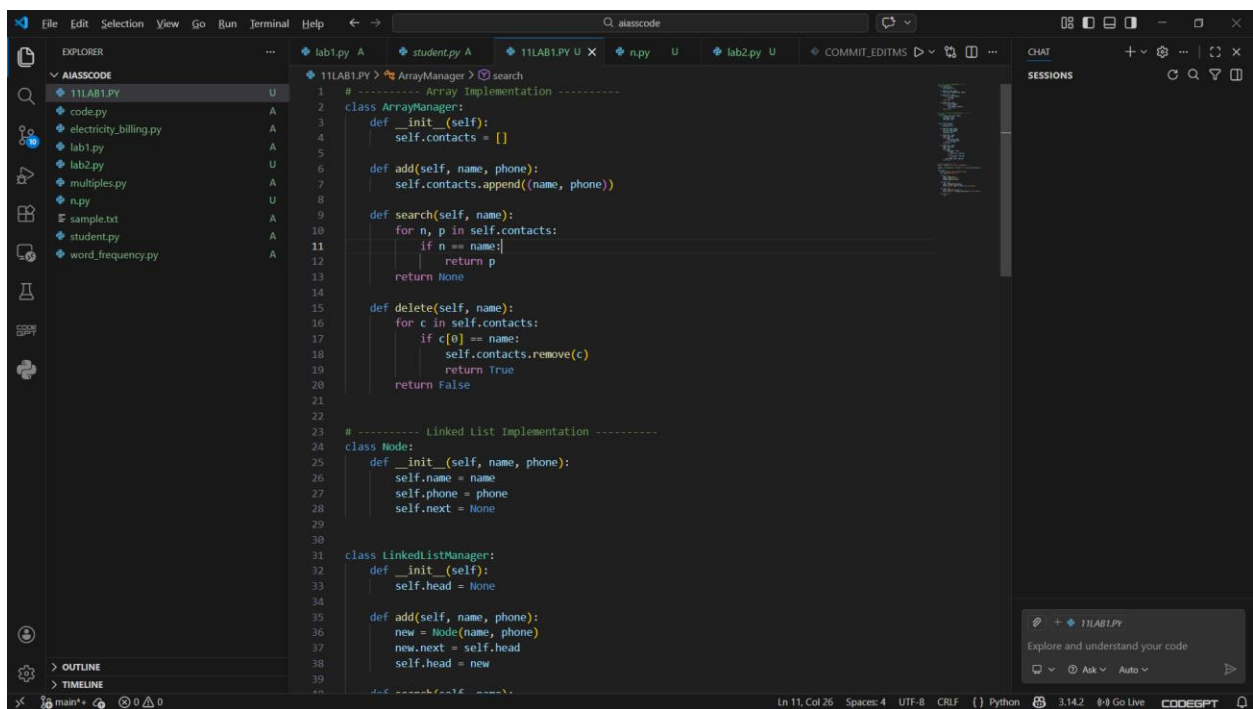
NAME:

HALLTICKET:

TASK1:

PROMPT

Create a Smart Contact Manager in Python using both Array (list) and Linked List. Implement operations to add, search, and delete contacts. Provide a simple menu to choose implementation and perform operations.



```
1 # ----- Array Implementation -----
2 class ArrayManager:
3     def __init__(self):
4         self.contacts = []
5
6     def add(self, name, phone):
7         self.contacts.append((name, phone))
8
9     def search(self, name):
10        for n, p in self.contacts:
11            if n == name:
12                return p
13        return None
14
15    def delete(self, name):
16        for c in self.contacts:
17            if c[0] == name:
18                self.contacts.remove(c)
19                return True
20        return False
21
22 # ----- Linked List Implementation -----
23 class Node:
24     def __init__(self, name, phone):
25         self.name = name
26         self.phone = phone
27         self.next = None
28
29 class LinkedListManager:
30     def __init__(self):
31         self.head = None
32
33     def add(self, name, phone):
34         new = Node(name, phone)
35         new.next = self.head
36         self.head = new
37
38     def search(self, name):
39         current = self.head
40         while current:
41             if current.name == name:
42                 return current.phone
43             current = current.next
44         return None
45
46     def delete(self, name):
47         current = self.head
48         while current:
49             if current.name == name:
50                 if current == self.head:
51                     self.head = current.next
52                 else:
53                     previous.next = current.next
54                 return True
55             previous = current
56             current = current.next
57         return False
```

```
lab1.py A student.py A 11LAB1.PY U x n.py U lab2.py U COMMIT_EDITMS CHAT SESSIONS
class LinkedListManager:
    def search(self, name):
        temp = self.head
        while temp:
            if temp.name == name:
                return temp.phone
            temp = temp.next
        return None

    def delete(self, name):
        temp = self.head
        prev = None
        while temp:
            if temp.name == name:
                if prev:
                    prev.next = temp.next
                else:
                    self.head = temp.next
                return True
            prev, temp = temp, temp.next
        return False

# ----- Menu -----
choice = input("Choose 1.Array 2.LinkedList: ")
manager = ArrayManager() if choice == "1" else LinkedListManager()

while True:
    print("\n1.Add 2.Search 3.Delete 4.Exit")
    ch = input("Enter choice: ")

    if ch == "1":
        name = input("Name: ")
        phone = input("Phone: ")
        manager.add(name, phone)

    elif ch == "2":
        name = input("Search name: ")
```

```
File Edit Selection View Go Run Terminal Help 11LAB1.PY U x n.py U lab2.py U COMMIT_EDITMS CHAT SESSIONS
EXPLORER
11LAB1.PY U
code.py A
electricity_billing.py A
lab1.py A
lab2.py U
multiples.py A
n.py U
sample.txt A
student.py A
word_frequency.py A
11LAB1.PY U x
class LinkedListManager:
    def delete(self, name):
        temp = self.head
        while temp:
            if temp.name == name:
                if prev:
                    prev.next = temp.next
                else:
                    self.head = temp.next
                return True
            prev, temp = temp, temp.next
        return False

# ----- Menu -----
choice = input("Choose 1.Array 2.LinkedList: ")
manager = ArrayManager() if choice == "1" else LinkedListManager()

while True:
    print("\n1.Add 2.Search 3.Delete 4.Exit")
    ch = input("Enter choice: ")

    if ch == "1":
        name = input("Name: ")
        phone = input("Phone: ")
        manager.add(name, phone)

    elif ch == "2":
        name = input("Search name: ")
        result = manager.search(name)
        print("Found:", result if result else "Not found")

    elif ch == "3":
        name = input("Delete name: ")
        print("Deleted" if manager.delete(name) else "Not found")

    elif ch == "4":
        break
```

OUTPUT:

The screenshot shows a Visual Studio Code editor window with the following components:

- EXPLORER:** A file tree on the left showing a project named 'AIASSCODE'. It contains several Python files: 'T1LAB1.PY' (selected), 'code.py', 'electricity_billing.py', 'lab1.py', 'lab2.py', 'multiples.py', 'n.py', 'sample.txt', 'student.py', and 'word_frequency.py'.
- TERMINAL:** The central pane displays the output of running 'T1LAB1.PY'. The output shows a menu with options 1 (Add), 2 (Search), 3 (Delete), and 4 (Exit). It demonstrates adding a student named 'hithesh' with phone number '234' and searching for a student named 'karthik' with phone number '846'.
- CHAT:** A panel on the right with a 'Python' icon and a 'SESSIONS' section.
- STATUS BAR:** The bottom of the window shows 'Ln 86, Col 14', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', '3.14.2', and 'Go Live'.

TASK2:

PROMPT:

Create a Library Book Request System in Python.

Implement a normal Queue (FIFO) and a Priority Queue where faculty requests have higher priority than student requests. Include enqueue and dequeue methods and a simple menu for testing.

```
1 from collections import deque
2 import heapq
3
4 # ----- Normal Queue -----
5 class BookQueue:
6     def __init__(self):
7         self.q = deque()
8
9     def enqueue(self, name):
10         self.q.append(name)
11
12     def dequeue(self):
13         return self.q.popleft() if self.q else "Empty"
14
15 # ----- Priority Queue -----
16 class PriorityBookQueue:
17     def __init__(self):
18         self.pq = []
19
20     def enqueue(self, name, role): # role = faculty/student
21         priority = 0 if role == "faculty" else 1
22         heapq.heappush(self.pq, (priority, name))
23
24     def dequeue(self):
25         return heapq.heappop(self.pq)[1] if self.pq else "Empty"
26
27 # ----- Menu -----
28 choice = input("Choose 1.Queue 2.PriorityQueue: ")
29
30 manager = BookQueue() if choice == "1" else PriorityBookQueue()
31
32 while True:
33     print("\n1.Enqueue 2.Dequeue 3.Exit")
34     ch = input("Enter choice: ")
35
36     if ch == "1":
37         name = input("Enter name: ")
38         # ...
```

```
17 class PriorityBookQueue:
18     def __init__(self):
19         self.pq = []
20
21     def enqueue(self, name, role): # role = faculty/student
22         priority = 0 if role == "faculty" else 1
23         heapq.heappush(self.pq, (priority, name))
24
25     def dequeue(self):
26         return heapq.heappop(self.pq)[1] if self.pq else "Empty"
27
28 # ----- Menu -----
29 choice = input("Choose 1.Queue 2.PriorityQueue: ")
30
31 manager = BookQueue() if choice == "1" else PriorityBookQueue()
32
33 while True:
34     print("\n1.Enqueue 2.Dequeue 3.Exit")
35     ch = input("Enter choice: ")
36
37     if ch == "1":
38         name = input("Enter name: ")
39         if choice == "2":
40             role = input("Role (faculty/student): ")
41             manager.enqueue(name, role)
42         else:
43             manager.enqueue(name)
44
45     elif ch == "2":
46         print("Served:", manager.dequeue())
47
48     elif ch == "3":
49         break
50
```

OUTPUT:

```
class PrioritybQueue:
    def __init__(self):
        self.queue = []

    def enqueue(self, name, role): # role = faculty/student
        # ...
        self.queue.append((name, role))

    def dequeue(self):
        # ...
        return self.queue.pop()
```

```
PS C:\Users\Administrator\Desktop\aiasscode> & C:/Users/Administrator/AppData/Local/Python/pythoncore
-3.14-64/python.exe c:/Users/Administrator/Desktop/aiasscode/LLLAB2.PY
Choose 1.Queue 2.PriorityQueue: 2

1.Enqueue 2.Dequeue 3.Exit
Enter choice: 1
Enter name: HITHESH
Role (faculty/student): 1291

1.Enqueue 2.Dequeue 3.Exit
Enter choice: 2
Served: HITHESH

1.Enqueue 2.Dequeue 3.Exit

1.Enqueue 2.Dequeue 3.Exit
Enter choice: 1
Enter name: HITHESH
Role (faculty/student): 1291

1.Enqueue 2.Dequeue 3.Exit
Enter choice: 2
Served: HITHESH

1.Enqueue 2.Dequeue 3.Exit
Enter choice: 2
Served: HITHESH

1.Enqueue 2.Dequeue 3.Exit
Enter choice: 3
PS C:\Users\Administrator\Desktop\aiasscode>
```

TASK3:

PROMPT:

Create an Emergency Help Desk Ticket System using Stack in Python.Implement push, pop, peek operations and methods to check if stack is empty or full.Simulate at least five tickets being raised and resolved to demonstrate LIFO behavior.

```
11LAB3.PY > HelpDeskStack > pop
1 # ----- Stack Implementation -----
2 class HelpDeskStack:
3     def __init__(self, size=10):
4         self.stack = []
5         self.size = size
6
7     def push(self, ticket):
8         if len(self.stack) == self.size:
9             print("Stack Full")
10        else:
11            self.stack.append(ticket)
12            print("Ticket added:", ticket)
13
14    def pop(self):
15        if self.is_empty():
16            print("No tickets")
17        else:
18            print("Resolved:", self.stack.pop())
19
20    def peek(self):
21        if self.is_empty():
22            print("No tickets")
23        else:
24            print("Top Ticket:", self.stack[-1])
25
26    def is_empty(self):
27        return len(self.stack) == 0
28
29 desk = HelpDeskStack()
30 desk.push("Login Issue")
31 desk.push("WiFi Not Working")
32 desk.push("Software Install")
33 desk.push("Printer Error")
34 desk.push("System Crash")
35 print("\npeek:")
36 desk.peak()
37 print("\nresolving Tickets:")
38 desk.pop()
39 desk.pop()
40 desk.pop()
41 desk.pop()
42 desk.pop()
```

OUTPUT:

```
PS C:\Users\Administrator\Desktop\aiasscode> & c:/Users/Administrator/AppData/Local/Python/pythoncore-3.14-64/python.exe
c:/Users/Administrator/Desktop/aiasscode/11LAB3.PY
Ticket added: Login Issue
Ticket added: Login Issue
Ticket added: WiFi Not Working
Ticket added: Software Install
Ticket added: Printer Error
Ticket added: System Crash
Ticket added: Printer Error
Ticket added: System Crash
Ticket added: System Crash

Peek:
Top Ticket: System Crash

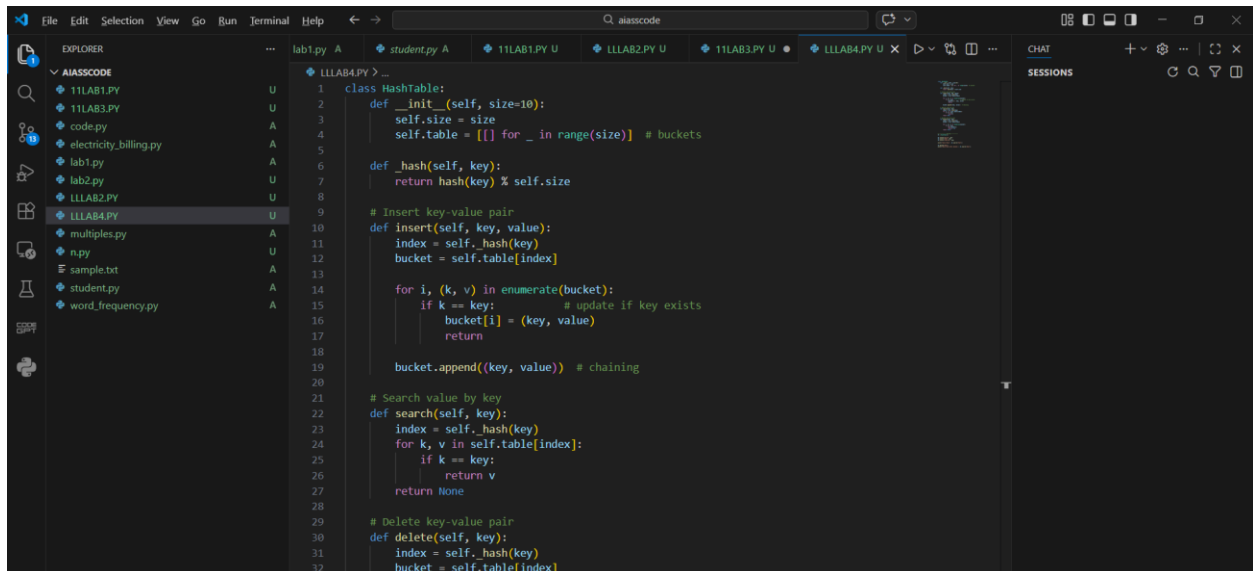
Resolving Tickets:
Peek:
Top Ticket: System Crash

Resolving Tickets:
Resolving Tickets:
Resolved: System Crash
Resolved: System Crash
Resolved: Printer Error
Resolved: Software Install
Resolved: WiFi Not Working
Resolved: Login Issue
PS C:\Users\Administrator\Desktop\aiasscode>
```

TASK4:

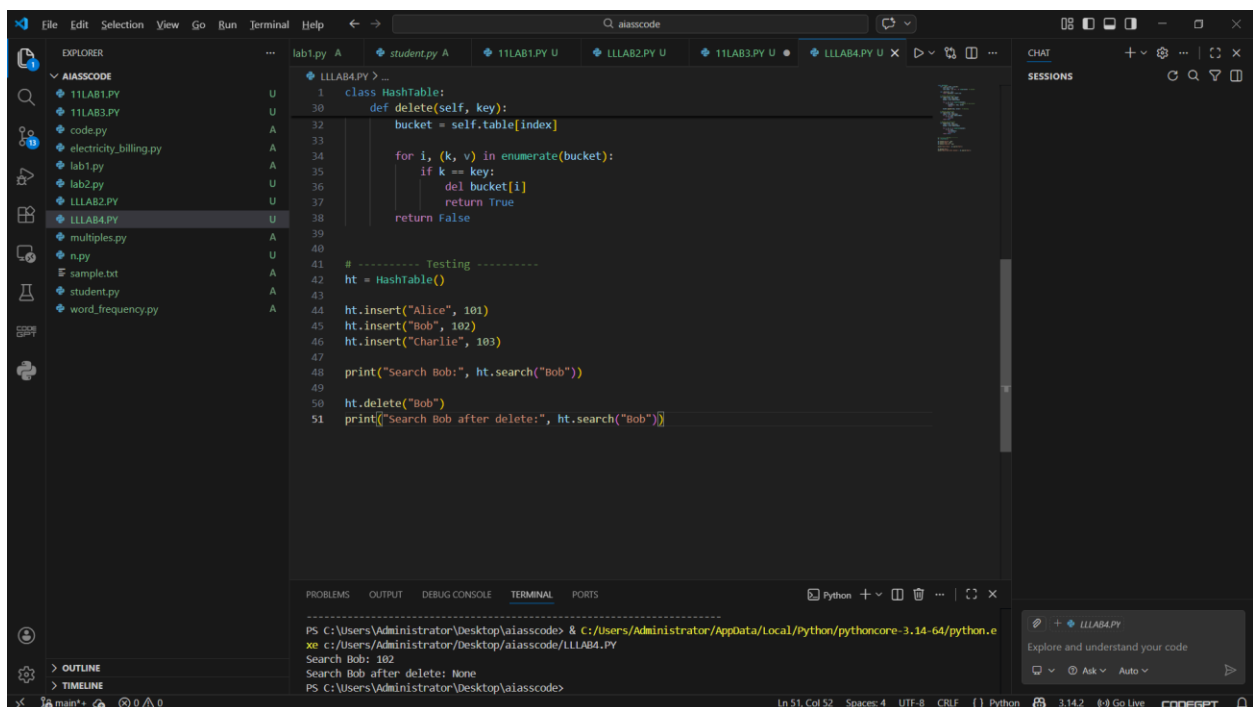
PROMPT:

Create a Hash Table in Python using chaining for collision handling. Implement insert, search, and delete methods with proper comments. Test the hash table with some sample data.



```
1 class HashTable:
2     def __init__(self, size=10):
3         self.size = size
4         self.table = [[] for _ in range(size)] # buckets
5
6     def _hash(self, key):
7         return hash(key) % self.size
8
9     # Insert key-value pair
10    def insert(self, key, value):
11        index = self._hash(key)
12        bucket = self.table[index]
13
14        for i, (k, v) in enumerate(bucket):
15            if k == key: # update if key exists
16                bucket[i] = (key, value)
17                return
18
19        bucket.append((key, value)) # chaining
20
21    # Search value by key
22    def search(self, key):
23        index = self._hash(key)
24        for k, v in self.table[index]:
25            if k == key:
26                return v
27        return None
28
29    # Delete key-value pair
30    def delete(self, key):
31        index = self._hash(key)
32        bucket = self.table[index]
```

OUTPUT:



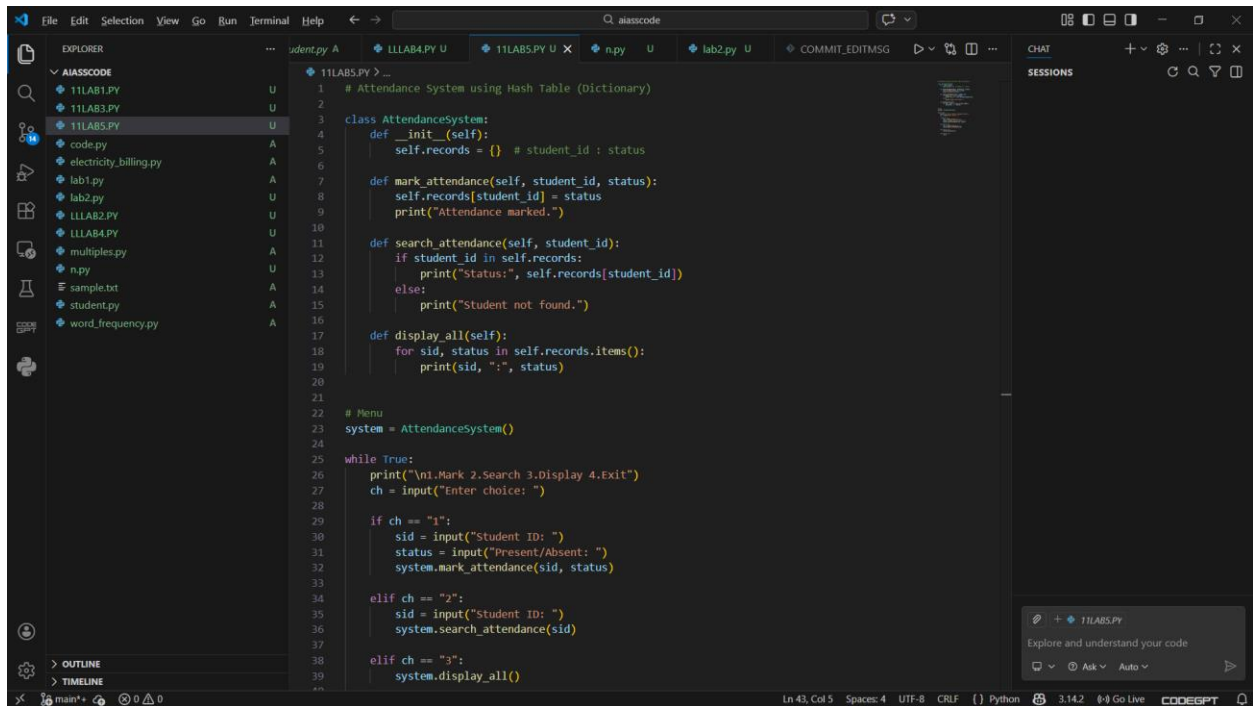
```
1 class HashTable:
2     def __init__(self, size=10):
3         self.size = size
4         self.table = [[] for _ in range(size)] # buckets
5
6     def _hash(self, key):
7         return hash(key) % self.size
8
9     # Insert key-value pair
10    def insert(self, key, value):
11        index = self._hash(key)
12        bucket = self.table[index]
13
14        for i, (k, v) in enumerate(bucket):
15            if k == key: # update if key exists
16                bucket[i] = (key, value)
17                return
18
19        bucket.append((key, value)) # chaining
20
21    # Search value by key
22    def search(self, key):
23        index = self._hash(key)
24        for k, v in self.table[index]:
25            if k == key:
26                return v
27        return None
28
29    # Delete key-value pair
30    def delete(self, key):
31        index = self._hash(key)
32        bucket = self.table[index]
33
34        for i, (k, v) in enumerate(bucket):
35            if k == key:
36                del bucket[i]
37                return True
38        return False
39
40    # ----- Testing -----
41    ht = HashTable()
42
43    ht.insert("Alice", 101)
44    ht.insert("Bob", 102)
45    ht.insert("Charlie", 103)
46
47    print("Search Bob:", ht.search("Bob"))
48
49    ht.delete("Bob")
50    print("Search Bob after delete:", ht.search("Bob"))
```

PS C:\Users\Administrator\Desktop\aiasscode> C:\Users\Administrator\AppData\Local\python\pythoncore-3.14-64\python.exe
c:\Users\Administrator\Desktop\aiasscode\LLLAB4.PY
Search Bob: 102
Search Bob after delete: None
PS C:\Users\Administrator\Desktop\aiasscode>

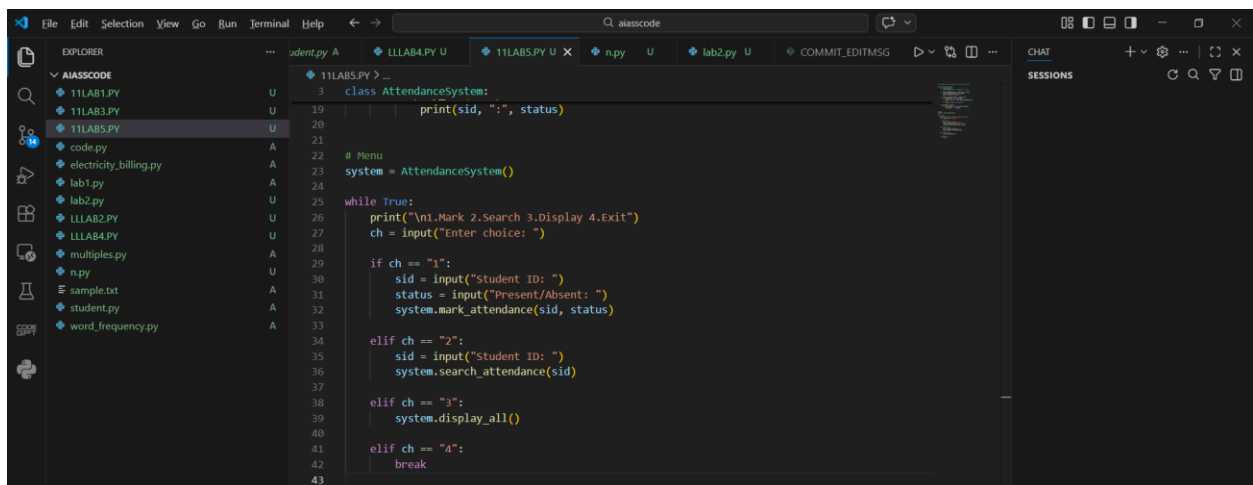
TASK5:

PROMPT:

Create a Student Attendance Tracking System using a dictionary (hash table). Implement functions to mark attendance, search attendance, and display all records. Use a simple menu-driven program.



```
1 # Attendance System using Hash Table (Dictionary)
2
3 class AttendanceSystem:
4     def __init__(self):
5         self.records = {} # student_id : status
6
7     def mark_attendance(self, student_id, status):
8         self.records[student_id] = status
9         print("Attendance marked.")
10
11     def search_attendance(self, student_id):
12         if student_id in self.records:
13             print("Status:", self.records[student_id])
14         else:
15             print("Student not found.")
16
17     def display_all(self):
18         for sid, status in self.records.items():
19             print(sid, ":", status)
20
21
22 # Menu
23 system = AttendanceSystem()
24
25 while True:
26     print("\n1.Mark 2.Search 3.Display 4.Exit")
27     ch = input("Enter choice: ")
28
29     if ch == "1":
30         sid = input("Student ID: ")
31         status = input("Present/Absent: ")
32         system.mark_attendance(sid, status)
33
34     elif ch == "2":
35         sid = input("Student ID: ")
36         system.search_attendance(sid)
37
38     elif ch == "3":
39         system.display_all()
40
41
42
43
```



```
3 class AttendanceSystem:
4     def __init__(self):
5         self.records = {} # student_id : status
6
7     def mark_attendance(self, student_id, status):
8         self.records[student_id] = status
9         print("Attendance marked.")
10
11     def search_attendance(self, student_id):
12         if student_id in self.records:
13             print("Status:", self.records[student_id])
14         else:
15             print("Student not found.")
16
17     def display_all(self):
18         for sid, status in self.records.items():
19             print(sid, ":", status)
20
21
22 # Menu
23 system = AttendanceSystem()
24
25 while True:
26     print("\n1.Mark 2.Search 3.Display 4.Exit")
27     ch = input("Enter choice: ")
28
29     if ch == "1":
30         sid = input("Student ID: ")
31         status = input("Present/Absent: ")
32         system.mark_attendance(sid, status)
33
34     elif ch == "2":
35         sid = input("Student ID: ")
36         system.search_attendance(sid)
37
38     elif ch == "3":
39         system.display_all()
40
41     elif ch == "4":
42         break
43
```


OUTPUT:

The screenshot shows a VS Code editor with the following components:

- Explorer:** A list of files in the project, including 11LAB1.PY, 11LAB3.PY, 11LAB5.PY (selected), code.py, electricity_billing.py, lab1.py, lab2.py, LLLAB2.PY, LLLAB4.PY, multiples.py, n.py, sample.txt, student.py, and word_frequency.py.
- Editor:** The code for 11LAB5.PY is displayed. It defines a class `AttendanceSystem` with methods `mark_attendance` and `display_student_info`. The `mark_attendance` method uses a dictionary to track attendance, and `display_student_info` prints the status of a student based on their ID.
- Terminal:** The output of the program is shown. It displays the menu options (1. Mark, 2. Search, 3. Display, 4. Exit) and the results of user input. For example, entering choice 1 and student ID 'HITHESH' results in 'Attendance marked.'.
- Chat/Output Panel:** On the right, there is a 'CHAT' section and a 'SESSIONS' list. Below these, there is a search bar and a list of sessions, including '11LAB5.PY'.