

## 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.

The screenshot shows a code editor interface with the following details:

- File Explorer (EXPLORER):** Shows a folder named "AIASSCODE" containing files: 11LAB1.PY, code.py, electricity\_billing.py, lab1.py, lab2.py, multiples.py, n.py, sample.txt, student.py, and word\_frequency.py.
- Code Editor:** The main pane displays Python code for a "Smart Contact Manager".

```
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
23 # ----- Linked List Implementation -----
24 class Node:
25     def __init__(self, name, phone):
26         self.name = name
27         self.phone = phone
28         self.next = None
29
30
31 class LinkedListManager:
32     def __init__(self):
33         self.head = None
34
35     def add(self, name, phone):
36         new = Node(name, phone)
37         new.next = self.head
38         self.head = new
39
```
- Terminal:** Shows the command "Ln 11, Col 26" and other terminal-related information.
- Bottom Status Bar:** Shows "Spaces: 4", "UTF-8", "CRLF", "Python", "3.14.2", "Go Live", and "CODEGPT".

```
1 lab1.py A 2 student.py A 3 11LAB1.PY U 4 n.py U 5 lab2.py U 6 COMMIT_EDITMSG D 7 8 CHAT 9 SESSIONS 10 11LAB1.PY > ... 12 13 class LinkedListManager: 14 15     def search(self, name): 16         temp = self.head 17         while temp: 18             if temp.name == name: 19                 return temp.phone 20             temp = temp.next 21         return None 22 23     def delete(self, name): 24         temp = self.head 25         prev = None 26         while temp: 27             if temp.name == name: 28                 if prev: 29                     prev.next = temp.next 30                 else: 31                     self.head = temp.next 32             return True 33             prev, temp = temp, temp.next 34         return False 35 36 # ----- Menu ----- 37 choice = input("Choose 1.Array 2.LinkedList: ") 38 39 manager = ArrayManager() if choice == "1" else LinkedListManager() 40 41 while True: 42     print("\n1.Add 2.Search 3.Delete 4.Exit") 43     ch = input("Enter choice: ") 44 45     if ch == "1": 46         name = input("Name: ") 47         phone = input("Phone: ") 48         manager.add(name, phone) 49 50     elif ch == "2": 51         name = input("Search name: ") 52 53     elif ch == "3": 54         name = input("Delete name: ") 55 56     elif ch == "4": 57         break 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
```

The screenshot shows the PyCharm IDE interface. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and several icons. The left sidebar has sections for Explorer, AIASSCODE, and Chat/Sessions. The main area displays a Python script named `11LAB1.PY`. The code implements a linked list manager with functions for adding, searching, deleting, and exiting. It uses `ArrayManager` or `LinkedListManager` based on user input. The code editor has syntax highlighting and a vertical scrollbar. A status bar at the bottom shows file paths, spaces used, and other details.

```
File Edit Selection View Go Run Terminal Help < > aiaascode

EXPLORER
AIASSCODE
  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 > ...
31 class LinkedListManager:
32     def delete(self, name):
33         if self.head == None:
34             return False
35         else:
36             self.head = temp.next
37             return True
38         prev, temp = temp, temp.next
39         return False
40
41 # ----- Menu -----
42 choice = input("Choose 1.Array 2.LinkedList: ")
43
44 manager = ArrayManager() if choice == "1" else LinkedListManager()
45
46 while True:
47     print("\n1.Add 2.Search 3.Delete 4.Exit")
48     ch = input("Enter choice: ")
49
50     if ch == "1":
51         name = input("Name: ")
52         phone = input("Phone: ")
53         manager.add(name, phone)
54
55     elif ch == "2":
56         name = input("Search name: ")
57         result = manager.search(name)
58         print("Found:", result if result else "Not found")
59
60     elif ch == "3":
61         name = input("Delete name: ")
62         print("Deleted" if manager.delete(name) else "Not found")
63
64     elif ch == "4":
65         break

Ln 86, Col 14 Spaces: 4 UTF-8 CRLF () Python 3.14.2 ⌂ Go Live CODEGPT
```

## OUTPUT:

```
-3.14-64/python.exe c:/Users/Administrator/Desktop/aiasscode/11LAB1.PY
Choose 1.Array 2.LinkedList: 1
1.Add 2.Search 3.Delete 4.Exit
Enter choice: 1
Name: hitesh
Phone: 234
A
1.Add 2.Search 3.Delete 4.Exit
Enter choice: 2
Search name: karthik
Name: hitesh
Phone: 234
A
1.Add 2.Search 3.Delete 4.Exit
Enter choice: 2
Search name: karthik
Found: 846
A
1.Add 2.Search 3.Delete 4.Exit
Enter choice: 2
Search name: karthik
Name: hitesh
Phone: 234
A
1.Add 2.Search 3.Delete 4.Exit
Enter choice: 2
Search name: karthik
Found: 846
A
1.Add 2.Search 3.Delete 4.Exit
Enter choice: 2
Search name: karthik
Name: hitesh
Phone: 234
A
1.Add 2.Search 3.Delete 4.Exit
Enter choice: 2
Search name: karthik
Found: 846
A
```

## 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.

```
File Edit Selection View Go Run Terminal Help <- > Q aiascode
EXPLORER ... lab1.py A student.py A 11LAB1.PY U 11LAB2.PY U n.py U lab2.py U D ...
AIASSCODE
11LAB1.PY
code.py
electricity_billing.py
lab1.py
lab2.py
LLAB2.PY
multiples.py
n.py
sample.txt
student.py
word_frequency.py

l11AB2.PY > -
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
16 # ----- Priority Queue -----
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
29 # ----- Menu -----
30 choice = input("Choose 1.Queue 2.PriorityQueue: ")
31
32 manager = BookQueue() if choice == "1" else PriorityBookQueue()
33
34 while True:
35     print("\n1.Enqueue 2.Dequeue 3.Exit")
36     ch = input("Enter choice: ")
37
38     if ch == "1":
39         name = input("Enter name: ")
40         manager.enqueue(name)
41
42     elif ch == "2":
43         print("Served:", manager.dequeue())
44
45     elif ch == "3":
46         break
47
48
49
50
```

In 50, Col 14 Spaces: 4 UTF-8 CRLF () Python 3.14.2 Go Live CODEGPT

```
File Edit Selection View Go Run Terminal Help <- > Q aiascode
EXPLORER ... lab1.py A student.py A 11LAB1.PY U 11LAB2.PY U n.py U lab2.py U D ...
AIASSCODE
11LAB1.PY
code.py
electricity_billing.py
lab1.py
lab2.py
LLAB2.PY
multiples.py
n.py
sample.txt
student.py
word_frequency.py

l11AB2.PY > -
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
29 # ----- Menu -----
30 choice = input("Choose 1.Queue 2.PriorityQueue: ")
31
32 manager = BookQueue() if choice == "1" else PriorityBookQueue()
33
34 while True:
35     print("\n1.Enqueue 2.Dequeue 3.Exit")
36     ch = input("Enter choice: ")
37
38     if ch == "1":
39         name = input("Enter name: ")
40         if choice == "2":
41             role = input("Role (faculty/student): ")
42             manager.enqueue(name, role)
43         else:
44             manager.enqueue(name)
45
46     elif ch == "2":
47         print("Served:", manager.dequeue())
48
49     elif ch == "3":
50         break
51
52
53
54
```

In 50, Col 14 Spaces: 4 UTF-8 CRLF () Python 3.14.2 Go Live CODEGPT

## OUTPUT:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left sidebar has a tree view labeled 'EXPLORER' with a node 'AIASSCODE' expanded, showing files like '11LAB1.PY', 'code.py', 'electricity\_billing.py', 'lab1.py', 'lab2.py', 'LLLAb2.PY', 'multiples.py', 'n.py', 'sample.txt', 'student.py', and 'word\_frequency.py'. The main area is a terminal window titled 'aiasscode' with the following content:

```
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: HITESH
Role (faculty/student): 1291
1.Enqueue 2.Dequeue 3.Exit
Enter choice: 2
Served: HITESH
1.Enqueue 2.Dequeue 3.Exit
Enter choice: 1
Enter name: HITESH
Role (faculty/student): 1291
1.Enqueue 2.Dequeue 3.Exit
Enter choice: 2
Served: HITESH
1.Enqueue 2.Dequeue 3.Exit
Enter choice: 3
PS C:\Users\Administrator\Desktop\aiasscode>
```

The bottom status bar indicates: In 50, Col 14, Spaces: 4, UTF-8, Python, 3.14.2, Go Live, CODEGPT.

## 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.
```

```
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.peek()
37 desk.pop()
38 desk.pop()
39 desk.pop()
```

## OUTPUT:

```
PS C:\Users\Administrator\Desktop\aiasscode> & C:/Users/Administrator/AppData/Local/Python/pythoncore-3.14-64/python.e
xe c:/Users/Administrator/Desktop/aiasscode/11LAB3.PY
xe 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
Top Ticket: System Crash

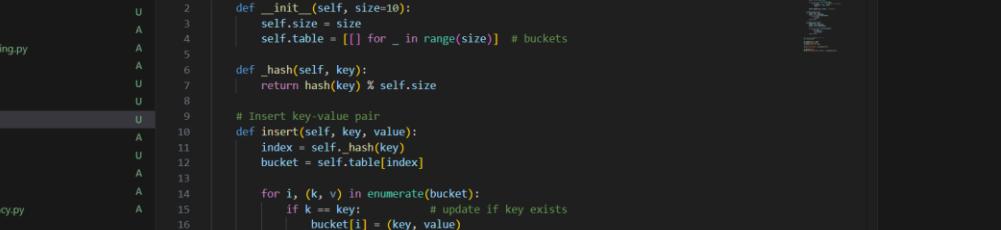
Resolving Tickets:
Peek:
Top Ticket: System Crash

Resolving Tickets:
Resolved: Login Issue
Resolved: WiFi Not Working
Resolved: Software Install
Resolved: Printer Error
Resolved: System Crash
Resolved: Printer Error
Resolved: System Crash
Resolved: System Crash
```

## 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.
```



The screenshot shows a code editor interface with multiple tabs open. The left sidebar lists files under 'EXPLORER' and 'SESSIONS'. The main area displays the content of 'LLL4.PY'.

```
class HashTable:
    def __init__(self, size=10):
        self.size = size
        self.table = [[] for _ in range(size)] # buckets

    def _hash(self, key):
        return hash(key) % self.size

    # Insert key-value pair
    def insert(self, key, value):
        index = self._hash(key)
        bucket = self.table[index]

        for i, (k, v) in enumerate(bucket):
            if k == key: # update if key exists
                bucket[i] = (key, value)
                return

        bucket.append((key, value)) # chaining

    # Search value by key
    def search(self, key):
        index = self._hash(key)
        for k, v in self.table[index]:
            if k == key:
                return v
        return None

    # Delete key-value pair
    def delete(self, key):
        index = self._hash(key)
        bucket = self.table[index]
```

## OUTPUT:

The screenshot shows a dark-themed interface of the Visual Studio Code (VS Code) code editor. The left sidebar displays a file tree under the 'EXPLORER' tab, showing files like 'lab1.py', 'student.py', '11LAB1.PY', '11LAB2.PY', '11LAB3.PY', and 'LLLAB4.PY'. The 'LLLAB4.PY' file is currently selected and open in the main editor area. The code implements a HashTable class with methods for insertion, search, and deletion. It includes a testing section with sample data ('Alice', 'Bob', 'Charlie') and their corresponding values (101, 102, 103). The bottom status bar shows the command line output:

```
PS C:\Users\Administrator\Desktop\aiasscode & c:/Users/Administrator/appData/Local/Python/pythoncore-3.14-64/python.exe
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.

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files in the 'AIASSCODE' folder, including 11LAB1.PY, 11LAB3.PY, 11LAB5.PY, 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.
- Code Editor:** The main window displays Python code for an 'AttendanceSystem' class. The code includes methods for marking attendance, searching attendance, and displaying all records. It uses a menu-driven approach with choices 1 through 4.
- Status Bar:** Shows 'Ln 43, Col 5' and other standard status bar information.
- Bottom Right:** A small floating window titled '11LAB5.PY' with the text 'Explore and understand your code'.

```
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     elif ch == "4":
42         break
43
```

The screenshot shows a second instance of a code editor with the same Python code for the 'AttendanceSystem' class. The code is identical to the one in the first screenshot, including the menu-driven logic and file structure.

```
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     elif ch == "4":
42         break
43
```

## OUTPUT:

```
11LAB5.PY > ...
3 CLASS AttendanceSystem:
11LAB5.PY U
11LAB1.PY U
11LAB3.PY U
11LAB5.PY U
code.py A
electricity_billng.py A
lab1.py A
lab2.py U
LLLAb2.PY U
LLLAb4.PY U
multiples.py A
n.py A
sample.txt A
student.py A
word_frequency.py A

xe c:/Users/Administrator/Desktop/aiasscode/11LAB5.PY

1.Mark 2.Search 3.Display 4.Exit
Enter choice: 1
Student ID: HITHESH
Present/Absent: 23
Attendance marked.

1.Mark 2.Search 3.Display 4.Exit
Enter choice: 1
Student ID: KARTHIK
Present/Absent: 20
Attendance marked.

1.Mark 2.Search 3.Display 4.Exit
Enter choice: 3
HITHESH : 23
KARTHIK : 20

1.Mark 2.Search 3.Display 4.Exit
Enter choice: 3
HITHESH : 23
KARTHIK : 20

1.Mark 2.Search 3.Display 4.Exit
Enter choice: 3
HITHESH : 23
KARTHIK : 20

1.Mark 2.Search 3.Display 4.Exit
Enter choice: 4
PS C:/Users/Administrator/Desktop/aiasscode>
```