Dt: 20/08/25

Task 4: Use various data types, List, Tuples and Dictionary in python programming key terms covered:

Data types, List, Tuple, Set, Dict

4.1 List - Cafeteria Sales

In your college cafeteria, the sales (in units) of a new snack are recorded for 7 days, Monday to Sunday. Store these values in a list, then find the total and average sales, identify the best & worst sales during index.

Aim:

Record a cafeteria's snack sales for 7 days using a list; compute total and average sales, find the best / worst day, and count how many days crossed a target.

Algorithm:

1. Start
2. Create an empty list sales = []
3. for 7 days, append integer sales to the list using append()
4. compute total = Sum(sales) and avg = total/7.
5. Find max_val = max(sales), min_val = min(sales)
6. Find corresponding days with index() (add +1 to convert to day number)
7. Count days above a target using count() on a boolean re-map or with a loop.
8. Stop.

Sample Input / output :

=RESTART: C:/Users/Bhavani/AppData/Local/Programs/Python/Python313/list_sales.py

enter the seven days sales count 100
enter the seven days sales count 450
enter the seven days sales count 1250
enter the seven days sales count 589
enter the seven days sales count 98
enter the seven days sales count 348
enter the seven days sales count 900
enter the seven days sales count 239
Sales (Mon..Sun): [100, 450, 1250, 589, 98, 348, 900, 239]
Total: 3974
Average: 567.71
Best Day: 3 with 1250
Worst Day: 5 with 98

4.2 Tuple - Lab TimeTable

your department has a fixed daily lab schedule represented by a tuple of starting hours (24-hour format). Write a program to check if a given start time exists in the tuple, count how many times it appears using count(), find its first position using index(), and display morning & afternoon slots using slicing.

Aim:

To manage and query an immutable daily lab slot schedule using a tuple, demonstrating membership checks, count(), index(), and slicing.

Algorithm:

1. Start
2. Define slots as a fixed tuple of integers.
3. Read query hour.
4. Check existence with query in slots.
5. Use count(); if positive, use index() to find the first position.
6. Slice into morning and afternoon
7. Print results
8. Stop.

Python Program

```
# TUPLE Scenario
slots = (9, 11, 14, 16, 14) # immutable daily schedule
query = 14
exists = (query in slots)
freq = slots.count(query)  # tuple.count()
first_pos = slots.index(query) + 1 if exists else "N/A"
                # tuple.index()
morning = slots[:2]
afternoon = slots[2:]
print("All lab slots:", slots)
print(f"is {query}:00 present?", exists)
print(f"{query}:00 occurs", freq, "time (s)")
print("First occurence position (1-based):", first_pos)
print("Morning slots:", morning)
print("Afternoon slots:", afternoon)
```

Sample Input / output :

```
= RESTART: C:\Users\Bhavani\AppData\Local\Programs\
Python\Python313\4b.py
All lab slots: (9, 11, 14, 16, 14)
Is 14:00 present? True
14:00 occurs 2 times (s)
First occurrence position (1-based): 3
Morning slots: (9, 11)
Afternoon slots: (14, 16, 14)
```

4.3

Dictionary :- Bookstore Billing

A bookstore's price list is stored in a dictionary where keys are item names and values are prices. Update the price of an item using (), find the costliest item using max() on items(), remove an out-of-stock item using pop(), and display all keys, values and items.

Aim :

To manage a live price list and bill a customer using dictionary methods and views.

Algorithm :

1. Start
2. Create an empty dictionary prices.
3. Ask the user for the number of items in the price list (n1)
4. Repeat for each item:
5. Get the item name.
6. Get the item price.
7. Add the item and price to prices.
8. Ask the user for an item to update (or press Enter to skip)
9. If the item exists in prices, get the new price and update it.
10. Find the costliest item by checking each item's price
11. Ask the user for an item to remove (or press Enter to skip)
12. If given, remove that item from prices
13. Show all available items, their prices, the costliest item, and the removed item's price.
14. Stop.

Python Program

```
prices = {}
n1 = int(input("Enter number of items in price list:"))
for _ in range(n1):
    item = input("Enter item name:")
    price = float(input(f"Enter price of {item}:"))
    prices[item] = price

# Optional price revision
rev_item = input("Enter item to update price (or press Enter
                  to skip):")
if rev_item in prices:
    new_price = float(input(f"Enter new price for {rev_item}:
    prices.update({rev_item: new_price}) # dict.update()

# Find costliest item
costliest_item = None
max_price = 0
for item, price in prices.items():
    if price > max_price:
        max_price = price
        costliest_item = item
# Remove out-of-stock item
remove_item = input("Enter an item to remove from price list
                     (or press Enter to skip):")
removed_price = None
if removed_item:
    removed_price = prices.pop(remove_item, None) # dict.pop()
# Display results
print("\nAvailable Items:", list(prices.keys())) # dict.keys()
print("Prices:", list(prices.values()))          # dict.values()
If costliest_item:
print("Costliest item:", costliest_item, "at", max_price)
If remove_item:
Print(f"Removed '{remove_item}' price (if existed):", removed
```

Sample Input / output :

```
= RESTART : C:/Users/Bhavani/AppData/Local/Programs/
Python/Python313/4c.py
Enter number of items in price list : 3
Enter item name : box
Enter price of box : 15
Enter item name : pen
Enter price of pen : 10
Enter item name : pencil
Enter price of pencil : 5
Enter item to update price (or press Enter to skip): box
Enter new price for box : 20
Enter an item to remove from price list (or press Enter to
                                    skip) : pen
Available Items : ['box', 'pencil']
Prices : [20.0, 5.0]
costliest item : box at 20
Removed 'pen' price (if existed) : 10.0
```

4.4 Set - Tech Fest Participation

Two events, AI Hackthon and Robotics Challenge, have participants IDS stored in two sets. Add a late registration to AI Hackthon, remove a withdrawn participant from Robotics using discard(). then find participates in both events (intersection) only in one (difference()), the total unique participants (union)

Aim:- To implement the python program for the participants ID's storing

Algorithm:-

1. Start
2. get AI hackathon participants details
3. Done the set opelations
4. Print the Hackthon details of the participants.

Program:-

```
# Get AI Hackathon participants
ai-hackthon =set()
n1 = int (input ("Enter number of participants in AI Hackathon :"))
for_in range (n1):
    Pid = input ("Enter participant ID:")
    ai - hackathon. add (Pid)
# Get Robotics challenge Participants
 robotics -chatlenge = set()
 n2 = int (input ("Enter number of participants in Robotics
                   challenge :")
 for_in- range (n2):
    Pid = input ("Enter participant ID:")
    robotics - challenge. add (pid)
# Add a late registrant
late - id = input ("Enter number of participants in Robotics challenge")
late. id = input ("Enter late registration ID for AI Hackathon (or
                   Press Enter to skip ):")
if late_id :
    ai - hackthon. add (late_id) # set add ()
```

```
# Remove a withdraw participant
remove_id = input("Enter withdraw participant ID from Robotics
                   Challenge (or press Enter to skip): ")
if remove_id:
    robotics_challenge.discard(remove_id) # set discard()

# set Operations

both = ai_hackathon.intersection(robotics_challenge)
only_ai = ai_hackathon.difference(robotics_challenge)
only_robotics = robotics_challenge.difference(ai_hackathon)
unique_all = ai_hackathon.union(robotics_challenge)

# output
Print("\nAI Hackathon:", ai_hackathon)
Print("Robotics Challenge:", robotics_challenge)
 Print("Both events:", both)
 Print("only AI:", only_ai)
 Print("only Robotics:", only_robotics)
 Print("Total unique Participants:", len(unique_all))
```

VEL TECH - CSE	
EX NO	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	5
RECORD (4)	
TOTAL (15)	
SIGN WITH DATE	15

Result:

Thus the implementation of various tuples, list, data types and dictionary was successfully executed.