Task 4:-

use various data type, lists, tuples

4.1:- List - cafeteria sales

In your college cafeteria the sales of a new snack for 7days, monday to sunday. Store values in a list then find the average sales identify the best and worst sales day using index 4.

Algorithm:-

1. Start
2. create an empty list sales = []
3. for 7days, append integers sales
4. Compute total = sum
5. min_val = min
6. Best_day
7. Print ("sales (max.sun):", sales).
8. Print ("total:", total)
9. Print with a loop
10. Stop.

Program:

list scenario

days = 7

sales = [ ]

target = 500 # target sales for the day

for s in range (8):

avg = total

sample - entires = int

total = sum

max-val = max

min-val = min (sales)

best-day = sales

Print ("Total:," total)

Print ("Best Day:, ", best day," with, max-val)

Print ("worst day: ", worst-day;" with min-val).

Sample input / output :-

Enter the seven day sales count 100

enter the seven day sales count 450

enter the seven day sales count 1250

enter the seven days sales count 589

enter the seven days sales count 98

enter the seven days sales count 345

enter the seven days sales count 900

enter the seven days sales count 239

sales (mon-sun) ; [100, 450, 1250, 589, 98, 345, 900, 239]

Total : 3971

Avg = 567.71

best days : 3 with 1750

lowist day :- 5 with 98

Python Program:-

```
# Triple Scenario
slots = (9,11,14,16,14)
query = 14
exists = (query in slots)
eq_pos = slots[2]
after_noon = slots[2:]
Print (f" is {query} :00 present? " exists).
Print (f"{query} :00 occurs", xx time (s)")
Print (" first occurrence position (1-based) :" first_pos)
Print ("morning slots :", morning)
Print ("Afternoon slots :", afternoon)
```

TASK 4.2 Lab time table

AIM:- To manage department has a fixed daily lab schedule represented by a tuple of starting slicing.

Algorithm:-

1. Start
2. Define slots as a fixed tuple of integerals
3. Read query hours.
4. Check existences with query in slots.
5. use count(); if positive, use index
6. Slice into morning and afternoon
7. Print results
8. Stop.

Sample input / output:-

All lab slots :- (9, 11, 14, 16, 14)

Is 14:00 Present? True

14:00 occurs

morning slots : (9, 11)

afternoon slots :- (14, 16, 14)

Python Program:-

```
Prices = {}
n = int(input("enter number of items in Price list:"))
for _ in range(n):
    item = input("Enter item name:")
    Price = float(input(f"enter price of {item}:"))
    Price[item] Price
# Optional Price revision
rev-item = input("enter internal item y))
Prices update ({ rev-item : new-Price}yy)
# dict.update
max-Price=0
for item, Price 0;
if Price > max-Price
costliest-item = item
# Remove out-of-struck item Price list
removed-item
removed-Price = Prices.pop(remove-item> none) #
d.ot.pop()
```

Task 4:3

Dictionary - Booking Billing:

AIM:-

To manage a live Price list and billing a customers using and views.

Algorithm:-

1. Start
2. Create an empty dictionary Prices
3. Ask the user for the number of items in the Price
4. Reach for each items.
5. Get the items name
6. Get the Price.
7. Add the items to Prices
8. Ask the Users for an items to update
9. If the item exists in Prices, get the new Price or update it.
10. find the cortliest item by checking each item Price.
11. Ask the user for an item
12. If given, remove that item from Prices.
13. Stop.

```
# Display results
print("In available items:", list(Price.keys())) # dict keys
print("Price :", list(Price.values())) # dict.values()

if costliest_item:
    print("costliest item : at" max_Price)

if remove_item:
    print(f'removed {remove_item} Price (if existed):"
                                    removed_Price).
```

sample input/output :-

Enter number of items in price list :- 3

Enter item name :- box

Enter Price name : 15

Enter item of Pen : 10

Enter Price of Pencil :- 5

Enter item of update Price : box

Enter new price for box :- 20

Enter an item of remove from price list :- Pen

→ variable items : ['box', 'Pencil']

Prices :- [20, 5]

costliest item :- box at 20.0

Removed 'Pen' . Price :- 10.0

```
# Get AI hackathon Patrons

ai_hackathon = set()
n = int(input("enter number of participants in AI hackathon"))
for _ in range(n):
    pid = input
    ai_hackathon.add(p_id)

# Get Robotics challenges Participants
robotics_challenge = set()

# Add a late registrat
late_id = input("enter late registrant ID for all A
    if late_id:
    ai_hackathon.add(late_id) # set.add()

# Remove a with draw participant
remove_id = input
    robotics challenge
    # set operations
    both = ai_hackathon.intersection(robotic_challenge)
    only_robotic ai = ai_hackathon.different(robotics_challe
    only_robotics = robotics_challenge.difference
    unique_all = ai_hackathon.union(robotics_challenge
    # output
Print("\nA Hackthon:", 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))
```

4.4 Set - Tech Fest Participation

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

SAMPLE INPUT / OUTPUT:-

```
>>>>
= RESTART: C:/Users/Bhavani/AppData/Local/Programs/Python/Python3...
Enter number of participants in AI Hackathon: 4
Enter participant ID: A1
Enter participant ID: A2
Enter participant ID: A4
Enter participant ID: A5
Enter number of participants in Robotics challenge: 4
Enter participant ID: A1
Enter participant ID: A2
Enter participant ID: A3
Enter participant ID: A7
Enter late registrant ID for AI Hackathon (or press enter to skip):
Enter withdrawn participant ID from Robotics Challenge (or press
to skip): A1

AI Hackathon: {'A1', 'A4', 'A5', 'A8', 'A2'}
Robotics challenge: {'A2', 'A7', 'A3'}
Both events: {'A2'}
only AI: {'A4', 'A8', 'A5', 'A1'}
only Robotics: {'A7', 'A3'}
>>> total unique participants: 7
```

VEL TECH - CSE	
EX NO.	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	15

Result: Thus, various data types, List, Tuples and Dictionary in Python Programming was used and verified successfully