Date: 27/8/25

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 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 and worst sales days using 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()
7. Count days above a target using count() on a boolean re-map or with a loop
8. Stop

output

```
enter seven days sales count 100
enter seven days sales count 450
enter seven days sales count 1250
enter seven days sales count 580
enter seven days sales count 98
enter seven days sales count 348
enter seven days sales count 909
enter seven days sales count 289

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
```

Program:

```
days = 7
sales = []
target=500
for in in range(8):
        sample-entries = int( input( "enter the seven
                                days sales count"))
        sales.append ( sample-entries)
total= sum( sales )
avg = total / days
max-val = max( sales)
min-val = min( sales)
best-day = sales-index (max-val )+1
worst-day = sales. index(min-val)+1
print("sales (mon ... sun):", sales
print("total =", total)
print("average:", round(avg,2))
print ("Best :", best-day ," with", max-val)
print("worst :", worst-day ", with", min-val)
```

Result:

Thus the python program of Record caferic
using list U executed successfully

TASK 4.2

Tuple - lab Time table

Aim: To manage and query an immutable daily lab slot schedule using a table demonstrating membership check, count() index(), and string

Algorithm:

- Start
2. Defines slot as a fixed tuple of integers
3. Read query hours
4. Check existence with query in slots
5. Use count(); if positive, use index() to find the 1st position
6. Print result
7. Stop

Program:

```
slots = (9, 11, 14, 16, 14)
query = 14
exists = (query in slots)
freq = slots.count (query)
first_pos = slots.index (query) +1 if exist
                 else "N/A" tuple index()
morning = slots [:2]
afternoon = slots [2:]
print (f"Is {query} = 00 present : "exists
print ("first occurence position(1-based) :",
                 first_pose)
```

Output

```
All lab slots: (9, 11, 14, 16, 14)
Is 14:00 present? True
14:00 occurs 2 times
first occurence position (1-based) = 3
morning slots = (9, 11)
Afternoon slots = (14, 16, 14)
```

Output

```
All lab slots: (9, 11, 14, 16, 14)
Is 14:00 present? True
14:00 occurs - 2 times)
First occurence position (1-based) = 3
morning slots = (9, 11)
Afternoon. slots = (14, 16, 14)
```

Output

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

Is 14:00 present? true

14:00 occurs: 2 times

first occurence position (1-based) = 3

morning slots = (9, 11)

Afternoon slots = (14, 16, 14)

```
print("Morning slots:", morning)
print(" Afternoon slots: ", afternoon)
```

Result:

Thus the python is menage immutable daily slot is executed successfully

TASK 4.4

Set - Tech Test participation

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

Get AI Hackathon participants

```
ai_hackathon = set()
n1 = int(input("Enter number of participants
            in AI Hackathon:"))
    for _ in range(n1):
        pid = input("Enter participants ID:")
       ai_hackathon.add(pid)
robotics_challenge = set()
n2 = int(input("Enter number of participants in
        robotics challenge:"))
    for _ in ranges(n2);
pid = input("Enter participant ID:")
  robotics_challenges.add(pid)
```

Add a ~~late~~ registrant

OUTPUT

```
print("\n AI 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
late-id = input("Enter late registrary ID
    for AI Hackathon (or press Enter to skip):")
if lateid;
ai-hackathon. add (late-id )
# Remove a withdrawn participant
remove-id = input(" enter withdrawn participant
        ID from Robotic challenges (or press enter
         to skip):")
if remove-id:
        robotics-challenges. discard (remove-id)
        set-discard.
both-ai- hackathon. intersection (robotics-challenges
only-ai = ai-hackathon. difference (robotic-challenge)
only- robotics =robotic. challenge
 difference ( ai-hackathon)
unique = all-ai- hackathon. union
                    (robotics- challenges)
```

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

Result:

Thus, the program is variables type are successfully erected