Task 4: Use various data type, list, Tuples, and Dictionary
25/8/25 in python programming key terms covered. Data types, List
set, Diet

4.1: Cafeteria sales.

In your college, cafeteria, the sales (in units) of a new snack are recorded for 7 days.

Aim: Record a cafeteria's snack sales for 7 days using, compute total and average sales, find 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 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. stop.

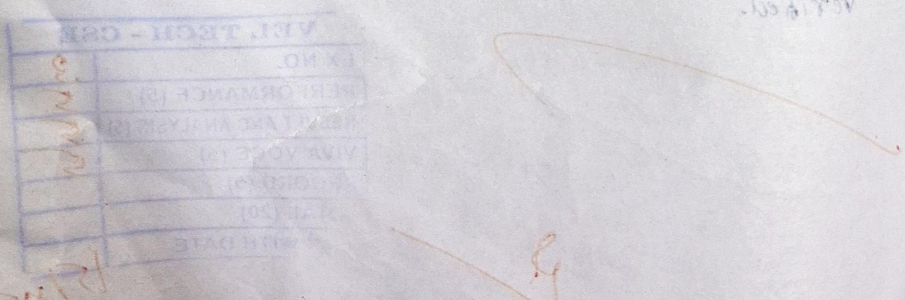
program (uses append(), index(), count():

```
# LIST scenario
days = 7
sales = []
target = 500

for s 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
```

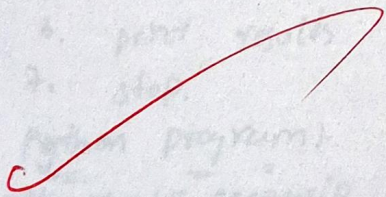
Sample Input/output:

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 900
enter the seven days sales count 348
enter the seven days sales count 400
enter the seven days sales count 239.

total : 3974
Average : 567.71
Best Day : 3 with 1250
worst day : 5 with 239.



```
print("sales (mon - sun):", sales)
print("total:", total)
print("Average:", round(avg, 2))
print("Best day:", best_day, "with", max_val)
print("worst day:", worst_day, "with", min_val)
```



Sample Input/output:

All lab slots : (9, 11, 14, 16, 14)
Is 14:00 present? True
14:00 occurs 2 time(s)
First occurence position (1-based): 3
Morning slots : (9, 11)
Afternoon slots : (14, 16, 14).

23/5/25

4.2 Tuple - Lab Timetable

Aim: To manage and query an immutable daily Lab sl
shedule using tuple. demonstrating membership checks, co
index(), and silicing.

Algorithm:

1. Start
2. Define slots as a fixed tuple of integers.
3. Read query hour.
4. check existence with query in slots.
5. slice into morining and a afternoon.
6. print resolts
7. stop.

Python program:

```
# TUPLE scenario
slots = (9, 11, 14, 16, 14).
query = 14,
exists = (query in slots).
first - pos = slots.index(query)+1 if exists else "N/A"
morning = slots[:2]
afternoon = slots[2:]
print("All lab slots,"slots)
print(f" is {query}:00 present?", exists)
print("First occurrence position (1-based):", first-pos)
print("Morning slots:", morning)
print("Afternoon slots:", afternoon)
```

Sample Input / output:

Enter number of items in price list: 3
Enter item name: box
Enter price of box: 15
Enter item name: pen
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 20.0
Removed 'pen' price (if existed): 10.0

28/8/25

4.3 - Dictionary - Bookstore Billing.

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

Algorithm:

1. Start
2. Create an empty dictionary price.
3. Ask the user for number of items in price list (n).
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 checking each item.
9. Ask the user for remove (or press enter to skip).
10. If given, remove that item from prices
11. show, all available items prices, the costliest item, and removed item's price.
12. Stop.

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}: "))
    pirice[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 to {rev_item}: ")
    print.update{rev_item : new_price})
# find costliest item.
costliest_item = None.
max_price = 0
for item, price in prices.items():
    if price > max_price:
        costliest_item = item
removed_item = input("Enter an item to remove price list (or press
                     Enter to skip): ")

removed_price = None.
```

```
print("In Available items:", list(prices.keys()))
print("prices:", list(prices.values()))
if costliest_item:
    print("costliest item:", costliest_item, "at", max_price)
if remove_item:
    print(f"Removed '{remove_item}' price (if existed):", removal
    - price).
```

Sample Input/output:

Enter number of participants in AI Hackthon: 4
Enter participant ID: A1
Enter participant ID: A2
Enter participant ID: A3
Enter participant ID: A4
Enter number of participants in Robotics challenge: 4
Enter participant ID: A1
Enter participant ID: A2
Enter participant ID: A7
Enter late registrant ID for AI Hackthon (or press Enter to skip): A8
Enter withdrawn participant ID from Robotics Challenge
(or press Enter to skip): A1

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

25
Set - TechFest participation

~~two events, AI~~

```
net AI Hackathon participants.
ai_hackathon = set()
n1 = int(input("Enter number of participants in AI Hackthon:"))
or _ in range(n1):
  pid = input("Enter participant ID:")
  ai_hackathon.add(pid)

net Robotics challenge participants.
obotics_challenge = 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 late registrant ID for AI Hackathon:")
if late_id:
  ai_hackathon.add(late_id)

remove_id = input("Enter withdrawn participant ID from Robotics
                   challenge (or press Enter to Skip):")
if remove_id:
  robotics_challenge.discard(remove_id)

# set operations
both = ai_hackton.intersection(robotics_challenge)
only_ai = ai_hackton.difference(robotics_challenge)
only_robotics = robtics_challenge.difference(ai_hackathon)
unique_all = ai_hackathon union(robotics_challenge)

# output
Print["\nAI Hackthon:", ai_hackathon)
Print("Robotics challenge:", robotics_challenge)
Print("Both events:", both)
print("only Robotics, only_robotics)
Print("Total unique participants:", len(unique_all)
```

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 the successfully to develop a program, using various data type, list, and dictionary in python programming key team executed.