Task - 4.

Use various data type List Tuples and Dictionary in Python Programming key terms covered Data type List Tuples sed Dict

Aim:

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

Algorithm:

1. start
2. create an empty list sales = []
3. for 7 days append integer sales to the list
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

Program: causes append ( ) index ( ) count ( )

```
# List scenario
days = 7
sales = []
```

sample input/output

enter the seven days sales count 100

enter the seven days sales count 250

enter the seven days sales count 584

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 (union ... sun): (100, 450, 1250, 589.

98, 348, 900, 239.

Total : 3974.

Average : 567.71

Best Day : 3 with 1250

work Day : 5 with 98.

```
target = 5.
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)
best_day = sales. index (max_val)
worst_day = sales.index (min_val) +1
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)
```

Result: Thus to use various data types list Tuples and Dictionary in Python Programming.

Task-4.2

Tuple - lab Timetable

Aim: To manage and query an immutable daily lab slots 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

```
#include s
# TUPLE scenario
slots = (9, 11, 14, 16, 14)
query = 14
exists = (query in slots)
freq = slots count (query)
morning = slot [:2].
```

Sample output

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

is 14:00 present true.

14:00 occurs 3 times (s)

first occurence position (1-based) : 3

morning slots : (9, 11)

Afternoon slots : (14, 16, 14).

```
afternoon = slots[2:]
Print("All lab slots:" slots)
Print-f(" {query} : 00 occur ; freq, "times)");
Print-f (" First occurence Position (1-based);
        first - Pos)
Print-f("morning slots" morning)
Print-f("afternoon slots", afternoon)
```

Result: Thus the Python Program is strange immutable daily that is executed successfully.

Dictionary - Bookstore Billing

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

Algorithm:

1. Start
2. Create an empty dictionary Price
3. As the user for the number of Items in the Price list (n_1)
4. Read for each item;
5. Get the Item Price;
6. Get the Item name
7. Add the Item and Price to Prices
8. Ask the user for an Item to update
9. If the Item exists in Price get the new Price and update it
10. find the costliest Item by Exchange each Item's Price
12. If given remove that Item from Price
13. Stop

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}"))
    Price[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})
#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");
```

Sample output input

Enter number of items in Price list :3

Enter item name : box

Enter Price of box : 15

Enter item of name : Pen

enter Price of Pen : 10

Enter item name : Pencil

Enter Price of Pencil : 5

Enter item ~~of~~ of date Price for

Press Enter to skip): box

Enter new Price for box = 20

Enter new ? or item to remove from Price

list for Press enter to skip) : Pen

Available Hence = [box ; ' Pencil "]

Price : [20, 0.5.0]

Cost last item boxcat 20.0

Removed Pen Price (if existence] : 100.

```
removed_Price = None
if removed_Item
    removed_Price = Prices.pop(remove_Item name)
# Display results
Print("In Available item:", list(Prices keys()))
Print("Prices:", list(Prices values()).
if costliest_Item:
    Print("costliest Item:" costliest_Item "at:
          max_Price)
If remove_Item
    Printf("Removed "{remove_Item} Price
    (If enter removed_Price).
```

Result: Thus the Python Program is manage live Price bill customer is executed successfully

TASK - 4.4

Tech fest Participation

Two events AI Hackathon and Robotics challenge have participation ID's stored in two sets Add a late registrate to AI Hackation remove a with draw participant from robotics using discard() then find participants in both events Intersection() only in one difference() the total unique participate (union())

```
# Get AI Hackathon Participation
ai = hackathon = set()
n1 = Int (Input (Enter number of participants
                 In AI Hacktaton);
for - In range (n1);
    Pid = Input ("Enter Participand ID:)
    oifhackaton add (Pid)
# Get Robotics challange Participants
robotics_challange = std()
n1: int (input ("Enter number of participants
                 in robotic challange:")
dor - in range (n):
    Pid = input ("Enter Participant ID:")
    robotics_challange add (p.d):
```

```
# Add a lab1 registrate
late_id = input ("Enter late registrate ID for
    "Hackathon");
if late_id:
    ai_hackathon.add (late_id)
# Remove a with draw participat.
Remove_id = Input ("Enter with drawn participate
        in from Robotics analogy:");
# insere remove id:
    robotics_challenge discard (late_id)
# set operations
both = ai_hardchon interseaction (robotics_
                                challenge)
only_ai = ai hackanon difference robotics.
only robotics = robotics_challenge difference (ai_
                                hac lutco);
unique all = ai_hackathen union (robotics_challenge
#output
Print ("In A1 Hackathon:" ai_hackathon)
Print ("Robotics challenge):"
Print ("Both events" both)
Print ("only AI" only ai)
Print (only Robotic only
Print ("Total unique participants" lencunique
Result: Thus the program variable type ou success fully
executed
```

Robotics challenge	VELTECH
EX No.	7
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	3
RECORD (5)	5
TOTAL (25)	
SIGNATURE WITH DATE	13

output:

```
Print ("\nA Hackthon :", ai - hack.arthon
Print (" Robotic challenge :" robotics :
                challenge).

Print ("Both events ", Both
Print ("only AI :" only-ai)
Print ("only Robotic :", only-robotic
Print (" total unique Participants :",
                len (unique_all)).
```