Task 4-1
27/8/25

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

Aim:- To use various data types, List, Tuples and Dictionary in Pythonprograming.

Algorithm:-

1. Start
2. For adding elements to a list first create a list with name "list" and assign the values within
3. For 7 days, append integer sales to list using
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 conver day to number).
7. Count days above a target using count().
8. Stop.

Program:-

```
# LIST scenario
    days=7
    Sales=[]
  target=500 # target sales for the day

for s in the range (8):
    Sample_entries = int (input("enter the seven days values"))
    Sales.append (sample_entries) # list.append()
  total = sum(sales)
    avg = total days
  max_val = max (sales)
  max_val  = min (sales)
```

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 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

```
best_day = sales.index (max_val)+1 # list.index()
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 ("\nWorst Day:", worst_day, "with ", min_val)
```

Result: Thus to use various data types, lists, Tuple and Dictionary in Pythonprogramming is verified successfully.

Task 4.2

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 fixed tuple of integers.
3. Read query hour.
4. Check existance 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.

Program:

```
## TUPLE Exercise
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:]
```

Sample Output:-

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

is 14:00 present: True

14:00 values 2 times)

First occurences position (1-based)=3

morning slots =(9,11)

After noon slots = (14,16,14)

```
print ("All the lab slots:", slots)
print (f"Is {query}:00 present?" exists)
print (f"{query}:00 ocurrs", freq, "times(s)")
print ("First occurrence position (1-based):", first_pos)
print ("Morning slots:", morning)
print ("Afternoon slots:", afternoon)
```

Result: Thus, the python programs manage immutable daily that is executed successfully.

Task 4.3

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

1. Start
2. Create an empty dictionary prices.
3. Repeat for each item.
4. Get the item name.
5. Get the item price.
6. Add the item price to prices.
7. Ask the user for a item to update
8. If the item exists prices, get the new price and update it.
9. Ask the user for an item by checking each item's price.
10. Ask the user for an item to remove.
11. If given, remove that item from prices.
12. Show all available items, their prices, the costliest item and removed price
13. 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}:"))
    prices[item] = price
```

Sample output / Input:-

Enter number of times in price list = 3

Enter item name : box

Enter price of box : 15

Enter item of 1 name = pen

Enter price of pen = 10

Enter item name = pencil

Enter price of pencil = 5

Enter item update, price/or

press Enter to skip) = box

Enter new price for box = 20

Enter an item for remove from price list (or press Enter to skip) = open

Available items = ['box', 'pencil']

Prices = [20.0, 5.0]

Costliest item = box at 20.0

Removed 'pen' price (if existed) : 10.0

```
# Optinal 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_prices})# dict.update()
# Find costliest item
 Costliest_item = None
 max_price = 0
for item, price in prices.items():
     if price > max_price:
          max_price = price
          Costliet_item = item
```

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

```
# Remove out-of_stock item
   remove_item = input Enter an item to remove from pric lst
   removed_price = None
   if items_remove:
      removed_price = prices.pop (remove_item, None)#
   dict.pop()
# Display results
   print ("In Available items"; lst prices.keys()) # dict.keys()
   print("Pricy:" lst prices.vals()   # dict.values
   if costlist_item:
   Print f" Removed  remove_item} 'price if exited):"
    removed_price)
```

Result:- Thus the Python programming is manage like price bill customer is executed successfully.