27-8-25

4. Use Various data types, List Tuples and Dictionary in Python Programming Key terms covered: Data types, List, Tuple, set, Dict

Aim: Record a Cafeteria's snack sales for 7 days using a list; compute total and average sales, find the best 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 () (add + 1 to convert to day numbers)

Program:

```
#List Scenario
days=7
sales=[]
target=500 #target sales for the day
for s in range(8):
    sample_entries=int(input("enter the seven days sales count"))
    sales.append(sample_entries) # list.append()
total = sum(sales)
avg = total/days
```

Sample input

enter the seven days count 100

450

1250

589

198

348

900

239

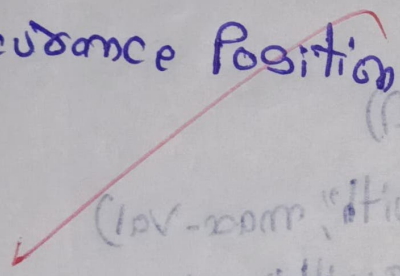
```
max_val = max(sales)
min_val = min(sales)

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("Worst Day:", worst_day, "with", min_val)
```

all lab slots (9, 11, 14, 16, 14)

Is 14:00 Present ? True

14:00 occurs.times(3)

First occurance Position (1 - based) : 3



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 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. Print result
7. stop

Python Program

```
# TUPLE Scenario
slots = (9, 11, 14, 16, 14)  #imm
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:]
Print ("All lab slots:", slots)
Print(f"is {query} ;oo Present ?", exists)
Print(f"{query} : oo occurs", freq, "time(s)")
Print ("First occurrence Position (1-based):", first_pos)
Print ("morning slots:", morning)
Print ("Afternoon slots:", afternoon)
```

Task 4.3

Aim: To manage a live Price list and bill a Customer using dictionary method and views.

Algorithm:

1. Start
2. Create an empty dictionary Prices
3. Get the item name
4. Get the item Price
5. Add the item and Price to Prices
6. Ask the user for an item to update
7. Ask the user for an item to remove
8. if given remove that item from Prices
9. Stop

Python 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}:"))
    Prices[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}) # dict.update()
# Find costliest item
costliest_item = None
    max_Price = 0
for item, Price in Prices.items():
```

- Enter no of items in Price list: 3
  item name: box
  Price of box: 15
  item name: Pen
  Price of Pen: 10
  item name: Pencil
  Price of Pencil: 5
  Update Price: box
  New Price for box: 20
  Remove from Price list: Pen

```
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):")
removed_Price = None
if remove_item:
    removed_Price = Prices.pop(remove_item, None)
# Display results
Print("\n Available items:", list(Prices.key())
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):", removed_Price)
```

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 participate from Robotics using discard(), then find participants 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 Participant ID: ")
    ai_hackathon.add(pid)
# Get Robotics challange Participants
robotics_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 (Or Press
    Enter to skip): ")
if late_id:
    ai_hackathon.add(late_id)
# Remove a withdrawn Participant
remove_id = input("Enter withdrawn Participant ID from Robotics
    challenge(or Press Enter to skip): ")
if remove_id:
    robotics_challenge.discard(remove_id)
```

output:

Participant ID: A_1

Participant ID: A_2

ID: A_4

ID: A_5

Robotics challenge: y

ID = A_1

" = A_2

" A_3

" A_4

AI Hackathon = {'A_1', 'A_4', 'A_5', 'A_8', 'A_2'}

Robotics challenge = {'A_2', 'A_7', 'A_3'}

Both events: {'A_2'}

only AI: {'A_4', 'A_8', 'A_5', 'A_1'}

set operations

both = ai_hackathon.intersection(robotics_challenge)
only_ai = ai_hackathon.difference(robotics_challenge)
only_robotics = robotics_challenge.difference(ai_hackathon)
unique_all = ai_hackathon.union(robotics_challenge)

Result: Thus that use of various data type list, tuple, dictionary in Python done succesfully.

VEL TECH	
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 use of various