TOBK 4: , scales of retail stack recorded for 7 days

27/8/25

Aim:- Record a cafteria's snack sales for 7 days using a list; 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 the list using append()
4) Compute total = sum(sales) and avg = total/7
5) Find corresponding days with index()
6) Count days above a target using count() on a boolean re-map
7) Stop

Program

```
#List scenario
days = 7
Sales = []
target = 500
```

Sample Input:-

=Restare:C:/Users/Bharani/AppData/Local/Programs/Python/Python313/list sales.py

enter the seven days count 100
450
1250
589
198
348
198
348
900
239

Output:-

Sales (Mon...Sun): [100, 450, 1250, 589, 48, 348, 900, 239]

Total: 3974

average: 567.71

Best Day: 3 with 1250

Worst Day: 5 with 98

```
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

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 Day :", best_day, "with", max_val)
print ("Worst Day :", worst day, "with", min_val)
```

Result :- Thus, the program has been verified and executed successfully

4.2

Aim:- To manage and query an immutable daily lab slot schedule using a tuple, demonstrating membership 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

```
#Tuple scenario
slots = (9,11,14,16,14)
query = 14

exist = (query in slots)
Freq = slots.count(query)
First_Pos = slots.index(query) + 1 if exists
            else "N/A" # tuple.index()
```

Output :-

(10, 'hello', 3.14, 'world')

10

hello

3.14

(10, 'hello', 3.14)

```
morning = slots[:2]
afternoon = slots[2:]

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

Result:- Thus the program has been verified and executed successfully

4.3

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

Algorithm:-

1) Start
2) Create an empty dictionary prices.
3) Ask the user for the number of items in the 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 update
9) If the item exits in prices, get the new price, and update it
10) Find the costliest item by checking each item's price
11) ask if the user for an item to remove
12) Show all available items, their prices, the costliest item, and the removed item's price
13) Stop

sample input

enter no of items in price list : 3

item name : box

item 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

available items : ['box', 'pencil']

prices : [20.0, 5.0]

costliest item : box at 20.0

'pen' price (if existed) : 100

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 in prices:
    new_price = float (input (f"Enter new price
                                for {rev_item}:"))
    prices.update ({rev_item: new_price})
#Display results

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}
```

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:- To manage a live price list and bill a customer using dictionary methods are executed and verified successfully.