

Task: 4.1 27/8/25

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

Aim: To use various data types, List, Tuples and Dictionary in python programming.

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 $\text{total} = \text{sum}(\text{Sales})$ and $\text{avg} = \text{total} / 7$

5. Find $\text{max-val} = \text{max}(\text{Sales})$, $\text{min-val} = \text{min}(\text{Sales})$

6. Find corresponding days with Index (adding to cover day to number).

7. Count days above target using $\text{count}()$.

8. Stop.

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"))

Sales.append(sample-entries) # list.append()

total = sum(Sales)

avg = total / days

max-val = max(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 58

enter the seven days sales count 98

enter the seven days sales count 384

enter the seven days sales count 900

enter the seven days sales count 239

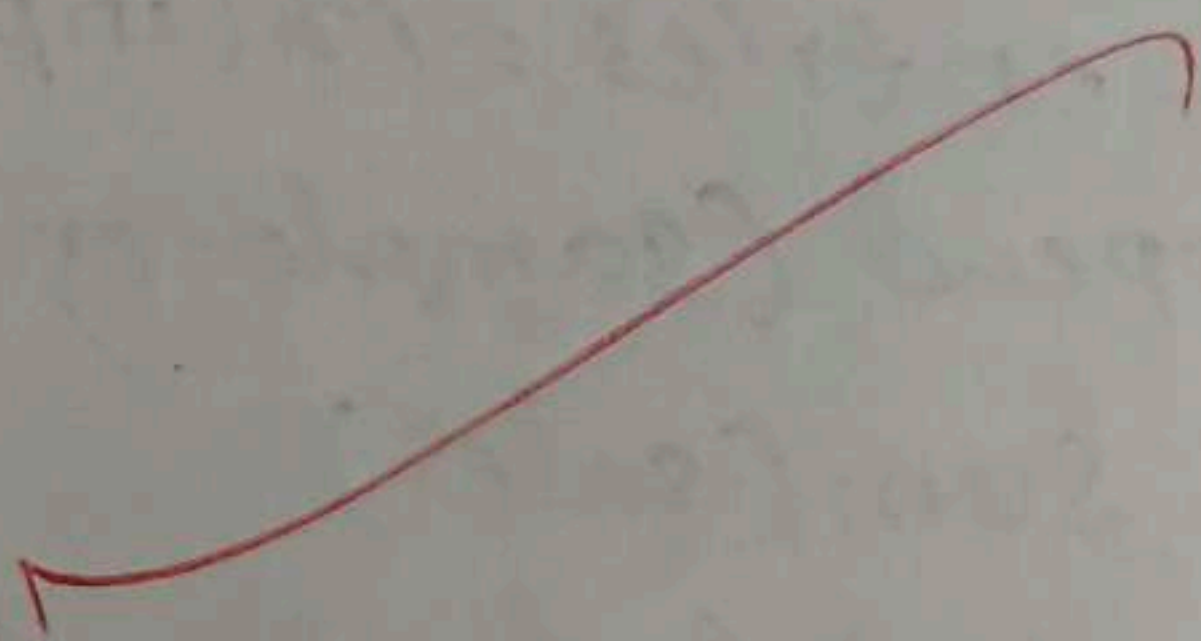
Sales (mon - sun): {100, 450, 1250, 58, 98, 384, 900, 239}

total: 3974

Average: 567.71

Best day: 3 with 1250

worst day: 5 with 98.



max-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(Avg²),

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.

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 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.

program:

```
# TUPLE Scenario
```

```
slots = (9, 11, 14, 16, 18) # immutable day schedule.
```

```
Query = 14
```

```
exists = (Query in slots)
```

```
freq = slots.count(Query) # tuple.count()
```

```
first_pos = slots.index(Query) + 1 if exists else None #  
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 time(s)

first occurrence position (1-based) = 3

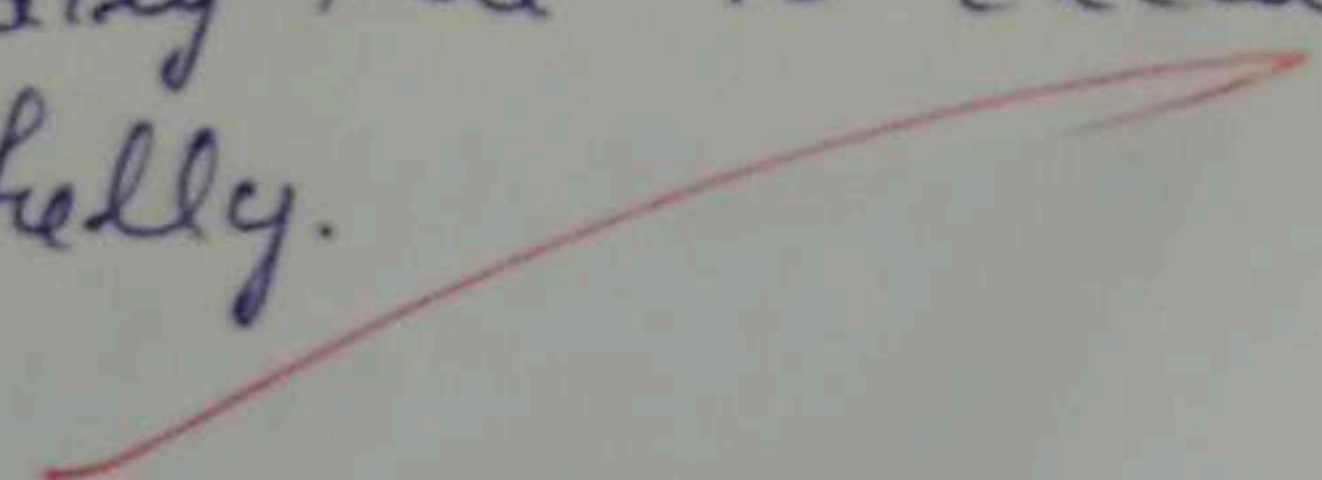
morning slots = (9, 11)

After noon slots = (14, 16, 14)


```
print("All the lab slots:", slots)
print(1. "Is {query}:00 present?" exists)
print(1. " {query}:00 occurs {freq}, "times(s)" )
print("First occurrence position (1-based: ", first
      pos)
print("morning slots: ", morning)
print("Afternoon slots: ", afternoon)
```



Result: Thus, the python program's manage
immutable daily that is executed
successfully.



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. 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 an item to update
8. If the item exists in 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 = {}
```

```
n1 = int(input("Enter number of items in price list"))
```

```
for _ in range(n1):
```

```
    item = input("Enter item name:")
```



```
price = float(input("Enter price of {item}:"))
```

```
prices[item] = price
```

```
# optional price revision
```

```
rev-item = input("Enter item to update price (as press  
Enter to skip):")
```

```
if rev-item in prices:
```

```
new-price = float(input("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():
```

```
    if price > max-price:
```

```
        max-price = price
```

```
        costliest-item = item.
```

```
# Remove out-of-stock item
```

```
remove-item = input("Enter an item to remove from p  
list")
```

```
removed-price = None
```

```
if remove-item in prices:
```

```
    removed-price = prices.pop(remove-item, None) #
```

```
dict.pop()
```

```
# Display results
```

```
print("\n Available items: ", list(prices.keys())) # dict
```

```
print("price: ", list(prices.values())) # dict.values
```

```
if costliest-item:
```


Sample output/Input:

Enter number of times in price list = 3

Enter item name: box

Enter price of box: 15

Enter item of i name = pen

Enter price of pen = 10

Enter item name = pencil

Enter price of pencil = 5

Enter item update, price for

Press Enter to Skip) = box

Enter new price for box = 20

Enter an item for remove from price

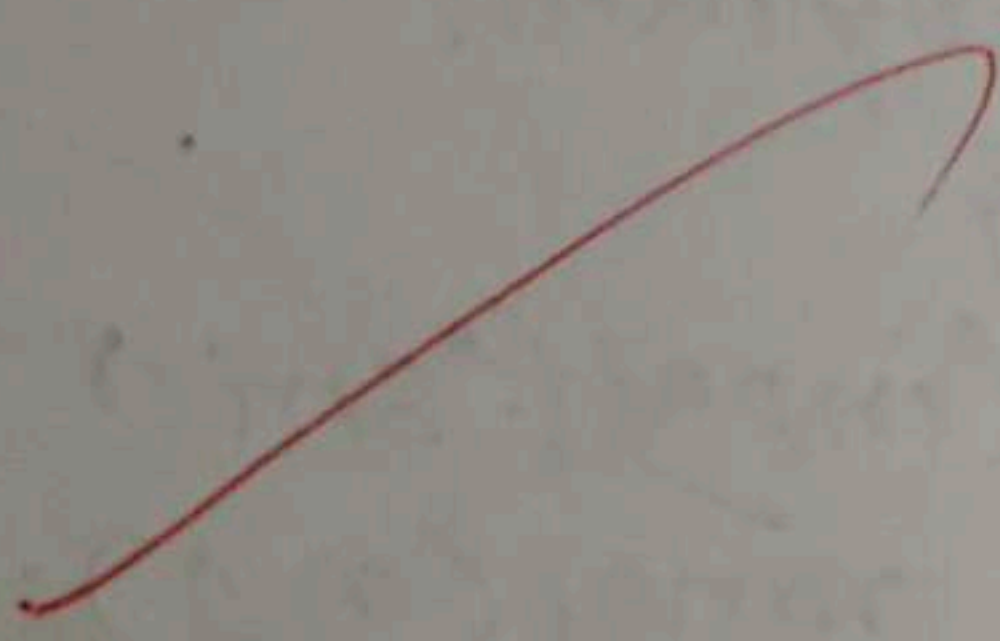
list (or press Enter of Skip) = Open

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

prices: {20, 0, 5, 0}

Costliest item = box at 20.0

Removed 'pen' price (if existed): 10.0



print f"Removed remove-item & "Price if existed"
removed-price)

VELTECH	
EX No.	6
PERFORMANCE	5
RESULT AND ANALYSIS	5
VIVA VOCE (3)	5
RECORD (4)	
TOTAL (15)	15
SIGN WITH DATE	

Result: Thus the python program is
manage like price bill customer is
~~executed~~ successfully.