

Task - 4.

use various data type list tuples and dictionary
in Python Programming key terms covered Data type
List tuples set 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 crossed 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 rem? or with a for loop
8. Stop.

Program: causes offend() index() count()

~~TEST~~ 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 : 384
enter the seven days	sales count : 298
enter the seven days	sales count : 348
enter the seven days	sales count : 900
enter the seven days	sales count : 239

sales (unsorted) : (100, 450, 1250, 589,

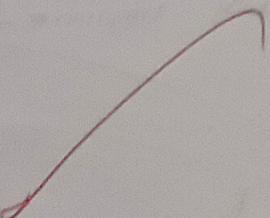
98, 348, 900, 239.

Total : 3974.

Average : 567.71

Best Day = 3 with 1250

worst Day = 5 with 98.



target = 5.

for s in storage(8):

sample_entries = [int(storage[i][0]) for i in range(7)]
days_sales_count = len(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 and () index() and slicing.

Algorithm:

1. Start
2. Define slots as adixed 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 result
8. Stop.

Python Program

```
structure s
# TUPLE scenario

slots = (9, 11, 14, 16, 18)

query = 14
exists = (query in slots)
for eg = slots.count(query)
morning = slot[:2].
```

sample output

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

is 14:00 Present Tive.

14:00 occurs ~~estimes~~ (5)

first occurrence Position (1-based) = 3

morning slots : (9, 11)

afternoon slots = (14, 16, 14).

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

```
Pointf("All lab slots:" slots)
```

```
Printf(" {q, very} : 00 occur ; -freq , "times")");
```

```
Printf(" First occurrence position (1-based);  
first - pos )
```

```
Printf(" morning slots " morning )
```

```
Printf("afternoon slots " afternoon )
```

Result: Thus the Python Program is strange

immutable daily that is executed
successfully.



Dictionary - Bookstore Billing

Aim: To merge a line price list and bill a customer using dictionary method and views

Algorithm:

1. Start
2. Create an empty dictionary Price
3. Ask the user for the number of items in the Price list (n_1)
4. Repeat 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
 11. If given remove that Item from Price
 13. Stop

Program

```
Prices = {}

n = int(input("Enter number of items in Price list:"))

for i in range(n):
    item = input("Enter item name:")
    price = float(input(f"Enter Price of {item}"))
    Prices[item] = price

# optional Price revision
new_item = input("Enter Item to update Price or Press enter to skip:")

if new_item in Prices:
    new_price = float(input("Enter new Price"))
    for item in Prices:
        if item == new_item:
            Prices[item] = new_price

# find costliest item
costliest_item = None
max_price = 0

for item in Prices.items():
    if item[1] > max_price:
        max_price = item[1]
        costliest_item = item[0]

# 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 for update Price for

Press Enter to skip) : box

Enter new Price for box = 20

Enter new ? an item to remove from Price

list for Press enter to skip) = 0 pen

Available Hence = [box ; 'Pencil']

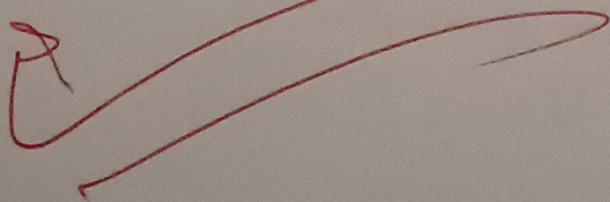
Price : (20, 0, 5, 0)

Cost last item box at 20.0

Removed Pen Price (if existence) : 10.0

```
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
    Print ("Removed " {remove - Item} Price
        (if enter removed , Price)).
```

Result: The the Python Program is manage
line Price bill customer is executed
successfully.



Task - 4.4

Tech Fest Participation

Two events AI Hackathon and Robotics challenge ge have participation ID's stored in two sets. Add a late registrant to AI Hackathon remove a withdraw participant from robotics using discard(). Then find participants in both events Intersection() only in one difference() the total unique participants (union())

Get AI Hackathon participation

ai = hackathon = set()

n1 = int(input("Enter number of participants in AI Hackathon:"))

for i in range(n1):

pid = input("Enter Participant ID: ")

ai.add(pid)

Get Robotics challenge participation

robotics_challenge = set()

n2 = int(input("Enter number of participants in robotic challenge:"))

for i in range(n2):

pid = input("Enter Participant ID: ")

robotics_challenge.add(pid)

Add a late registrant
late_id := input ("Enter late registrant ID for
" + hackathon);

if late_id:
 ai_hackathon.add(late_id)

Remove a withdrawn participant.

removed_id := input ("Enter withdrawn participant
ID from Robotics challenge:"),

Inset remove ID:

robotics_challenge.discard(removed_id)

set operations

both = ai_hackathon.intersection(robotics_

only_ai = ai_hackathon.difference(robotics_
challenge)

only_robotics = robotics_challenge.difference(ai_
hackathon);

unique_all = ai_hackathon.union(robotics_challenge)

output

Print ("Final Hackathon:" ai_hackathon)

Print ("Robotics challenge")

Print ("Both events " both)

Print ("only AI" "only AI")

Print ("only Robotics only")

Print ("Total unique participants" len(unique_

Results from the program variable type are successfully
executed

Robotics VELTECH challenge	
EX No.	
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	
MARKING DATE	13

outlined:

Print ("InA Hackathon : " ai - hackathon

Print (" Robotics challenge : " robotics
challenge).

Print ("Both events " . Both

Print ("only AI: " only - ai)

Print ("only Robotics : " , only - robotics

Print (" total unique Points : "
len (unique - all)).