27-8-25

Task 4 use various data types, List, Tuples and Dictionary in Python programming

Aim : Record a cafeteria's snack sales for 7 Days using a list, compute total and average sales, find the best day.

Algorithm

1. Start
2. create an empty list sales = []
3. For 7 Days, append integer sales append()
4. compute total = sum(sales), min_val = total/7
5. Find max_val = max(sales), min_val = min(sales)
6. Find corresponding days with index().
7. 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 count"))
sales.append(sample_entries) # list.append()
    total = sum(sales)
    avg = total / days

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

Sample Input

= Restart: C:/Users/Bhavani/AppData/Local/Programs/Python/Python313/listsales.py

enter the seven days count 100

450

1250

589

198

348

900

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

Sample input/output:

=Restart: C:\Users\Bhavani\AppData\Local\Programs\Python\Python313\lab.py)

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

Is 14:00 present? True

14:00 occurs 2 time(s)

First occurance position (1-based): 3

morning slots: (9, 11)

after noon slots: (14, 16, 14)

4.2

your Department has a fixed Daily lab schedule represented by a tuple of starting hours

Aim : To manage and query an immutable daily lab slot schedule tuple, demonstrating, membership, checks, count(), index(), and slicing

Algorithm

1. Start
2. Define slots as a fixed tuple
3. Read query hour
4. check existence query in slots
5. use count();
6. Print result.
7. Stop

Program

```
# Tuple Scenario
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:]
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("after noon slots:", afternoon)
```

4.3

A Book store's price is stored in a Dictionary where keys are item names and values are prices

Aim: To manage a live price list and bill a customer using Dictionary methods and views

Algorithm

Start

2. create an empty Dictionary prices
3. Ask the user for the no of item in price
4. Repeat for each item
5. Get the item price
6. Ask the user for an item to update
7. If the item exists in prices, get new price
8. If- given, remove that item from prices
9. stop

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): ")
```

Sample input

```
= Restart: C:\Users\Bhavani\AppData\Local\Programs
Python\Python313\uc.py
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
available Items: ['box', 'Pencil']
Prices: [20.0, 5.0]
costliest item: box at 20.0
'Pen' Price (If existed): 100
```

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

print("prices:", list(prices.values()))   # dict.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 challang have participants' ID's stored in two sets

```
# Get AI Hackathon participants
ai_hackathon = set()
n1 = int(input("Enter no of participants in AI Hackathon: "))
for _ in range(n1):
pid = input("Enter participant ID: ")
    ai_hackathon.add(pid)
# Get Robotics challenge participants
robotics_challenge = set()
n2 = int(input("Enter no 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) # set.add()
# Remove a withdrawn participant
remove_id = input("Enter withdrawn participant ID from Robotics challenge (or press. enter to skip): ").
```

Sample input/output:

no of participants in AI Hackathon: 4

participant ID : A1

participant ID : A2

participant ID : A4

" ID = A5

Robotics challenge: 4

" ID : A1

" " = A2

" " = A3

" " = A6

AI Hackathon : {'A1', 'A4', 'A5', 'A8', 'A2'}

Robotics challenge : {'A2', 'A7', 'A3'}

Both events : {'A2'}

only AI : {'A4', 'A8', 'A5', 'A1'}

only Robotics : {'A7', 'A3'}

unique participants : 7

```
If remove_id:
robotics_challenge.discard(remove_id) # set.
                                      discard()

# 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)

# output
print("\n AI 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 participants:", len(unique_all))
```

VEL TECH	
EX NO.	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	

Result: Thus the Recorda cafeteria's use of various Data types, List, Tuples Dictionary in Python is done successfully.