2/11/25

Task4: Use various data types, List, Tuples and Dictionary in python programming key terms covered: Datatypes List, tuple, set, Dict

4.1 List - Cafeteria Sales

In your college cafeteria, the sales (in units) of a new snack are recorded for 7 days, Monday to sunday. Store these values in a list, then find the total and average sales, identify the best and worst sales days using index()

Aim: Record a cafeteria's snack sales 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 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).
7) Count days above a target using count() on a boolean re-map (or) with a loop.
8) Stop.

Program (uses append(), index(), count()):

```
# 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()
```

output:

[10,20,30]

[10,30]

[30]

[5,8,9,15,30,89]

The minimum is : 5

The maximum is : 89

The Sum is : 156

The average : 26.0

```
total = sum(sales)
avg = total / days
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)
```

4.2 Tuple_Lab timetable

Your department has a fixed daily lab schedule represented by a tuple of starting hours (24-hour format). write a program to check if a given start time exists in the tuple, count how many times it appears using count(), find its first position using index(), and display morning and afternoon slots using slicing

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 slot 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

Program

```
# Tuple Scenario
slot = (9, 11, 14, 16, 14) # immutable daily schedule
query = 14
exists = (query in slots)
freq = slots.count(query) # tuple.count()
first_pos = slot.index(query + 1 if exist else "N/A"
                                # tuple.index()
```

Output:

10 hello 3.14 world

10

hello

3.14

world

('hello', 3.14)

['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 occurrence position(1-based)!!", first_pos)
print("Morning slots:", morning)
print("Afternoon slots:", afternoon.
```

4.3 Dictionary – Bookstore Billing

A book store's price list is stored in a dictionary where keys are items names and values are prices. Update the price of an item using update(), find the costliest item using max() on items(), remove an out of stock item using pop and display all keys, values, and items.

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) Ask the user for the number of items in the price list
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 (or press Enter to skip)
9) If the costliest item by checking each item's price and update it.
10) Find the costliest item by checking each item's price
11) Ask the user for an item to remove (or press Enter to skip)
12) If given, remove that item from prices.
13) Show all available item their prices, the costliest item, and the removed item's price
14) Stop.

Python program:

```
prices = {}
n1 = int(input("Enter no 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 verision
rev_item = input("Enter item to update price list:"))
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():
    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): ")
remove_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.keys()
print("Prices", list(prices.values()))  # dict.values()
if costlist_item:
    print("costlist item", costlist_item, "at", max_price)
if remove_item:
    print(f"removed '{remove_item}' price{'' if existed else ':'}
                removed_price
```

Result: Thus the implement the Dictionary_BookStore Billing.

Output:

{'name': 'Alice', 'age': 30 'city': 'New york'}

Alice

30

{'name': 'James', 'age': 30, 'city': Newyork'}

{name': 'James', 'age': 30}

KEY: name

KEY: age

dict_items([('name', 'James'), ('age', 30)])

4.4 set -Tech Fest participation

Two events, AI Hackathon and Robotics Challenge. Have to Participants IDs stored in two sets. Add a late registrant to AI Hackathon, remove a withdrawn Participant 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 Challenge 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 participant
          ID from Robotics Challenge (or press Enter
          to skip):") AI Hackathon

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

Output:

Print("In 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))

```
if late_id:
    ai_hackathon.add(late_id) # set add()

# Remove a withdraw participant
remove_id = input("Enter with drawn participant ID
                  from Robotics chalenge (or press
                  Enter to skip):")

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_chalenge.differenc(ai_hackathon)
unique_all = ai_hackathon.union(robotics_chalenge)
# Output
print("In AI Hackathon:", ai_hackathon)
Print("Robotics chalenge:", robotics_challenge)
print("Both events:", both)
print("only AI:", only_ai)
Print("only Robotis:", only_robotig)
Print("Total unique participants:", len(unique_all))
```

Result:: Thus implement the set tech fest participation

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