Name - Bhavya Mahawar
batch - 80
sapid-590029516

1. Scan n values in range 0-3 and print the number of times each value has occurred.

```
counts = {0: 0, 1: 0, 2: 0, 3: 0}

while True:
    try:
        n = int(input("Enter the number of values to scan (n): "))
        if n < 0:
            print("Please enter a non-negative number for n.")
        else:
            break
    except ValueError:
        print("Invalid input. Please enter an integer.")

print(f"\nPlease enter {n} values, each in the range 0-3.")
for i in range(n):
    while True:
        try:
            value = int(input(f"Enter value {i + 1} (0-3): "))
            if 0 <= value <= 3:
                counts[value] += 1
                break
            else:
                print("Value out of range. Please enter a value between 0 and 3.")
        except ValueError:
            print("Invalid input. Please enter an integer.")
print("\nOccurrence counts:")
for value, count in counts.items():
    print(f"Value {value}: {count} times")
```

2. Create a tuple to store n numeric values and find average of all values

```
while True:
    try:
        n = int(input("Enter the number of numeric values for the tuple (n): "))
        if n < 0:
            print("Please enter a non-negative number for n.")
        else:
            break
    except ValueError:
        print("Invalid input. Please enter an integer.")
values_list = []
print(f"\nPlease enter {n} numeric values.")
for i in range(n):
    while True:
        try:
            value = float(input(f"Enter value {i + 1}: "))
            values_list.append(value)
            break
        except ValueError:
            print("Invalid input. Please enter a numeric value.")

my_tuple = tuple(values_list)
if my_tuple:
    average = sum(my_tuple) / len(my_tuple)
    print(f"\nThe created tuple is: {my_tuple}")
    print(f"The average of the values in the tuple is: {average:.2f}")
else:
    print("No values were entered, so an average cannot be calculated.")
```

3. WAP to input a list of scores for N students in a list data type. Find the score of the runner-up and print the output. Sample Input N = 5 Scores= 2 3 6 6 5 Sample output 5 Note: Given list is [2, 3, 6, 6, 5]. The maximum score is 6, second maximum is 5. Hence, we print 5 as the runner-up score.

```
while True:
    try:
```

```
        N = int(input("Enter the number of students (N): "))
        if N <= 0:
            print("Please enter a positive integer for N.")
        else:
            break
    except ValueError:
        print("Invalid input. Please enter an integer.")

print(f"\nPlease enter {N} scores, separated by spaces:")
while True:
    try:
        scores_str = input()
        scores = list(map(int, scores_str.split()))
        if len(scores) != N:
            print(f"You entered {len(scores)} scores, but N is {N}. Please enter exactly {N} scores.")
        else:
            break
    except ValueError:
        print("Invalid input. Please enter numeric scores separated by spaces.")
unique_scores = list(set(scores))
unique_scores.sort(reverse=True)
if len(unique_scores) >= 2:
    runner_up_score = unique_scores[1]
    print(f"\nThe scores are: {scores}")
    print(f"The runner-up score is: {runner_up_score}")
else:
    print("\nCould not determine a runner-up score. Ensure there are at least two unique scores.")
```

4. Create a dictionary of n persons where key is name and value is city. a) Display all names b) Display all city names c) Display student name and city of all students. d) Count number of students in each city.

```
from collections import Counter
while True:
    try:
        n = int(input("Enter the number of persons (n): "))
        if n < 0:
            print("Please enter a non-negative number for n.")
        else:
            break
    except ValueError:
        print("Invalid input. Please enter an integer.")
persons_dict = {}
print(f"\nPlease enter the name and city for {n} persons.")
for i in range(n):
    name = input(f"Enter name for person {i + 1}: ")
    city = input(f"Enter city for {name}: ")
    persons_dict[name] = city

print("\n--- Dictionary Created ---")
print(persons_dict)
print("\na) All Names:")
for name in persons_dict.keys():
    print(name)
print("\nb) All City Names (Unique):")
# Using a set to get unique city names
unique_cities = set(persons_dict.values())
for city in unique_cities:
    print(city)
print("\nc) Student Name and City:")
for name, city in persons_dict.items():
    print(f"{name}: {city}")
print("\nd) Number of Students in Each City:")
city_counts = Counter(persons_dict.values())
for city, count in city_counts.items():
    print(f"{city}: {count} student(s)")
```

5. Store details of n movies in a dictionary by taking input from the user. Each movie must store details like name, year, director name, production cost, collection made (earning) & perform the following :- a) print all movie details

b) display name of movies released before 2015 c) print movies that made a profit. d) print movies directed by a particular director.

```
while True:
    try:
        n = int(input("Enter the number of movies (n): "))
        if n < 0:
            print("Please enter a non-negative number for n.")
        else:
            break
    except ValueError:
        print("Invalid input. Please enter an integer.")

movies = {}
print(f"\nPlease enter details for {n} movies.")
for i in range(n):
    print(f"\nEntering details for Movie {i + 1}:")
    movie_name = input("  Enter movie name: ")
    while True:
        try:
            year = int(input("  Enter release year: "))
            break
        except ValueError:
            print("Invalid input. Please enter an integer for the year.")
    director_name = input("  Enter director name: ")
    while True:
        try:
            production_cost = float(input("  Enter production cost: "))
            break
        except ValueError:
            print("Invalid input. Please enter a number for production cost.")
    while True:
        try:
            collection_made = float(input("  Enter collection made (earning): "))
            break
        except ValueError:
            print("Invalid input. Please enter a number for collection made.")

    movies[movie_name] = {
        'year': year,
        'director': director_name,
        'production_cost': production_cost,
        'collection_made': collection_made
    }

print("\n--- Movie Details Stored ---")
print("\na) All Movie Details:")
if not movies:
    print("  No movie details to display.")
else:
    for name, details in movies.items():
        print(f"  Movie: {name}")
        for key, value in details.items():
            print(f"    {key.replace('_', ' ').title()}: {value}")
print("\nb) Movies Released Before 2015:")
found_before_2015 = False
for name, details in movies.items():
    if details['year'] < 2015:
        print(f"  - {name} ({details['year']})")
        found_before_2015 = True
if not found_before_2015:
    print("  No movies found released before 2015.")
print("\nc) Movies That Made a Profit:")
found_profit_movies = False
for name, details in movies.items():
    profit = details['collection_made'] - details['production_cost']
    if profit > 0:
        print(f"  - {name} (Profit: {profit:.2f})")
        found_profit_movies = True
if not found_profit_movies:
    print("  No movies found that made a profit.")
print("\nd) Movies Directed by a Particular Director:")
if movies:
    search_director = input("  Enter director's name to search: ")
    found_director_movies = False
    for name, details in movies.items():
        if details['director'].lower() == search_director.lower():
            print(f"  - {name} (Year: {details['year']})")
            found_director_movies = True
```

```
    if not found_director_movies:
        print(f"  No movies found by director '{search_director}'.")
else:
    print("  No movie details are stored to search by director.")
```

6. Create a contact book where users can store, search, update, and delete contacts. Use dictionary for storing contacts.

```
contacts = {}

def add_contact():
    name = input("Enter contact name: ")
    if name in contacts:
        print(f"Contact '{name}' already exists. Use update option to modify it.")
    else:
        phone = input("Enter phone number: ")
        email = input("Enter email address: ")
        contacts[name] = {'phone': phone, 'email': email}
        print(f"Contact '{name}' added successfully.")

def view_contacts():
    if not contacts:
        print("Contact book is empty.")
    else:
        print("\n--- Your Contacts ---")
        for name, details in contacts.items():
            print(f"Name: {name}, Phone: {details['phone']}, Email: {details['email']}")
        print("-----------------------")

def search_contact():
    name = input("Enter contact name to search: ")
    if name in contacts:
        details = contacts[name]
        print(f"\n--- Contact Found ---")
        print(f"Name: {name}, Phone: {details['phone']}, Email: {details['email']}")
        print("---------------------")
    else:
        print(f"Contact '{name}' not found.")

def update_contact():
    name = input("Enter contact name to update: ")
    if name in contacts:
        print(f"Current details for '{name}': Phone: {contacts[name]['phone']}, Email: {contacts[name]['email']}")
        new_phone = input("Enter new phone number (leave blank to keep current): ")
        new_email = input("Enter new email address (leave blank to keep current): ")

        if new_phone:
            contacts[name]['phone'] = new_phone
        if new_email:
            contacts[name]['email'] = new_email
        print(f"Contact '{name}' updated successfully.")
    else:
        print(f"Contact '{name}' not found.")

def delete_contact():
    name = input("Enter contact name to delete: ")
    if name in contacts:
        del contacts[name]
        print(f"Contact '{name}' deleted successfully.")
    else:
        print(f"Contact '{name}' not found.")

def contact_book_menu():
    while True:
        print("\n--- Contact Book Menu ---")
        print("1. Add Contact")
        print("2. View All Contacts")
        print("3. Search Contact")
        print("4. Update Contact")
        print("5. Delete Contact")
        print("6. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            add_contact()
```

```
        elif choice == '2':
            view_contacts()
        elif choice == '3':
            search_contact()
        elif choice == '4':
            update_contact()
        elif choice == '5':
            delete_contact()
        elif choice == '6':
            print("Exiting Contact Book. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")
contact_book_menu()
```

7. Create a Todo list Manager where users can add, view, and remove tasks. Use List for storing tasks.

```
tasks = []

def add_task():
    task = input("Enter the task to add: ")
    tasks.append(task)
    print(f"Task '{task}' added.")

def view_tasks():
    if not tasks:
        print("Your To-Do list is empty!")
    else:
        print("\n--- Your To-Do List ---")
        for i, task in enumerate(tasks, 1):
            print(f"{i}. {task}")
        print("-----------------------")

def remove_task():
    if not tasks:
        print("Your To-Do list is empty. No tasks to remove.")
        return

    view_tasks()
    while True:
        try:
            task_number = int(input("Enter the number of the task to remove: "))
            if 1 <= task_number <= len(tasks):
                removed_task = tasks.pop(task_number - 1)
                print(f"Task '{removed_task}' removed.")
                break
            else:
                print("Invalid task number. Please enter a number from the list.")
        except ValueError:
            print("Invalid input. Please enter an integer.")

def todo_list_manager():
    while True:
        print("\n--- To-Do List Manager ---")
        print("1. Add Task")
        print("2. View Tasks")
        print("3. Remove Task")
        print("4. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            add_task()
        elif choice == '2':
            view_tasks()
        elif choice == '3':
            remove_task()
        elif choice == '4':
            print("Exiting To-Do List Manager. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")
todo_list_manager()
```

Github repo link-https://github.com/mahawarbhavya/pythonexperiments.git