Q-1

import random

odd\_integers = [random.choice(range(1, 100, 2)) for \_ in range(5)]

print("List of 5 odd integers:", odd\_integers)

even\_integers = [random.choice(range(2, 100, 2)) for \_ in range(4)]

print("List of 4 even integers:", even\_integers)

odd\_integers[2] = even\_integers

print("Updated list of odd integers with 3rd element replaced:", odd\_integers)

flattened\_list = []

for item in odd\_integers:

    if isinstance(item, list):

        flattened\_list.extend(item)

    else:

        flattened\_list.append(item)

print("Flattened list:", flattened\_list)

flattened\_list.sort()

print("Sorted flattened list:", flattened\_list)

OUTPUT:

List of 5 odd integers: [53, 37, 89, 67, 89]

List of 4 even integers: [78, 14, 74, 70]

Updated list of odd integers with 3rd element replaced: [53, 37, [78, 14, 74, 70], 67, 89]

Flattened list: [53, 37, 78, 14, 74, 70, 67, 89]

Sorted flattened list: [14, 37, 53, 67, 70, 74, 78, 89]

Q-2

import random

random\_integers = [random.randint(1, 100) for \_ in range(20)]

print("Generated list of 20 random integers:", random\_integers)

user\_number = int(input("Enter a number to find its positions in the list: "))

positions = [index for index, value in enumerate(random\_integers) if value == user\_number]

if positions:

    print(f"The number {user\_number} is found at the following position(s): {positions}")

else:

    print(f"The number {user\_number} is not found in the list.")

OUTPUT:

Generated list of 20 random integers: [10, 33, 73, 6, 93, 100, 64, 73, 41, 26, 36, 78, 10, 82, 50, 38, 55, 23, 36, 41]

Enter a number to find its positions in the list: 100

The number 100 is found at the following position(s): [5]

Q-3

import random

random\_numbers = [random.randint(1, 30) for \_ in range(50)]

print("Generated list of 50 random numbers:", random\_numbers)

unique\_numbers = list(set(random\_numbers))

print("List after removing duplicates:", unique\_numbers)

OUTPUT:

Generated list of 50 random numbers: [8, 5, 7, 25, 6, 16, 27, 28, 3, 26, 4, 16, 13, 23, 8, 30, 23, 20, 1, 19, 9, 20, 6, 15, 16, 3, 8, 11, 3, 15, 2, 11, 17, 14, 4, 9, 28, 17, 17, 26, 30, 7, 11, 12, 27, 3, 26, 18, 29, 2]

List after removing duplicates: [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 23, 25, 26, 27, 28, 29, 30]

Q-4

import random

random\_numbers = [random.randint(-100, 100) for \_ in range(30)]

print("Generated list of 30 random numbers:", random\_numbers)

positive\_numbers = [num for num in random\_numbers if num > 0]

negative\_numbers = [num for num in random\_numbers if num < 0]

print("List of positive numbers:", positive\_numbers)

print("List of negative numbers:", negative\_numbers)

OUTPUT:

Generated list of 30 random numbers: [-92, 41, -29, -25, 95, 36, 78, -89, -27, 47, 14, -24, -73, 10, 87, -19, 26, 1, 97, -47, 80, -88, 14, -46, 27, 48, 11, 82, 97, 98]

List of positive numbers: [41, 95, 36, 78, 47, 14, 10, 87, 26, 1, 97, 80, 14, 27, 48, 11, 82, 97, 98]

List of negative numbers: [-92, -29, -25, -89, -27, -24, -73, -19, -47, -88, -46]

Q-5

strings = ["hello", "world", "tushar", "kathiriya", "patel"]

print("Original list of strings:", strings)

uppercase\_strings = [string.upper() for string in strings]

print("List of strings in uppercase:", uppercase\_strings)

OUTPUT:

Original list of strings: ['hello', 'world', 'tushar', 'kathiriya', 'patel']

List of strings in uppercase: ['HELLO', 'WORLD', 'TUSHAR', 'KATHIRIYA', 'PATEL']

Q-6

fahrenheit\_temps = [90,87,76,25,35]

print("Temperatures in Fahrenheit:", fahrenheit\_temps)

celsius\_temps = [(temp - 32) \* 5/9 for temp in fahrenheit\_temps]

print("Temperatures in Celsius:", celsius\_temps)

OUTPUT:

Temperatures in Fahrenheit: [90, 87, 76, 25, 35]

Temperatures in Celsius: [32.22222222222222, 30.555555555555557, 24.444444444444443, -3.888888888888889, 1.6666666666666667]

Q-7

class Stack:

    def \_\_init\_\_(self):

        self.stack = []

    def push(self, item):

        self.stack.append(item)

        print(f"{item} has been pushed onto the stack.")

    def pop(self):

        if self.is\_empty():

            print("Stack is empty, cannot pop.")

        else:

            popped\_item = self.stack.pop()

            print(f"{popped\_item} has been popped from the stack.")

    def peek(self):

        if self.is\_empty():

            print("Stack is empty.")

        else:

            print(f"Top element is: {self.stack[-1]}")

    def is\_empty(self):

        return len(self.stack) == 0

    def display(self):

        if self.is\_empty():

            print("Stack is empty.")

        else:

            print("Stack elements:", self.stack)

def menu():

    stack = Stack()

    while True:

        print("\nMenu:")

        print("1. Push element to stack")

        print("2. Pop element from stack")

        print("3. Peek top element")

        print("4. Check if stack is empty")

        print("5. Display stack")

        print("6. Exit")

        choice = input("Enter your choice: ")

if choice == "1":

            item = int(input("Enter the element to push onto the stack: "))

            stack.push(item)

        elif choice == "2":

            stack.pop()

        elif choice == "3":

            stack.peek()

        elif choice == "4":

            if stack.is\_empty():

                print("Stack is empty.")

            else:

                print("Stack is not empty.")

        elif choice == "5":

            stack.display()

        elif choice == "6":

            print("Exiting program.")

            break

        else:

            print("Invalid choice, please try again.")

menu()

OUTPUT:

Menu:

1. Push element to stack

2. Pop element from stack

3. Peek top element

4. Check if stack is empty

5. Display stack

6. Exit

Q-8

class Queue:

    def \_\_init\_\_(self):

        self.queue = []

    def enqueue(self, item):

        self.queue.append(item)

        print(f"{item} has been added to the queue.")

    def dequeue(self):

        if self.is\_empty():

            print("Queue is empty, cannot dequeue.")

        else:

            dequeued\_item = self.queue.pop(0)

            print(f"{dequeued\_item} has been removed from the queue.")

    def peek(self):

        if self.is\_empty():

            print("Queue is empty.")

        else:

            print(f"Front element is: {self.queue[0]}")

    def is\_empty(self):

        return len(self.queue) == 0

    def display(self):

        if self.is\_empty():

            print("Queue is empty.")

        else:

            print("Queue elements:", self.queue)

def menu():

    queue = Queue()

    while True:

        print("\nMenu:")

        print("1. Enqueue element to queue")

        print("2. Dequeue element from queue")

        print("3. Peek front element")

        print("4. Check if queue is empty")

        print("5. Display queue")

        print("6. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":

            item = int(input("Enter the element to enqueue to the queue: "))

            queue.enqueue(item)

        elif choice == "2":

            queue.dequeue()

        elif choice == "3":

            queue.peek()

        elif choice == "4":

            if queue.is\_empty():

                print("Queue is empty.")

            else:

                print("Queue is not empty.")

        elif choice == "5":

            queue.display()

        elif choice == "6":

            print("Exiting program.")

            break

        else:

            print("Invalid choice, please try again.")

menu()

OUTPUT:

Menu:

1. Enqueue element to queue

2. Dequeue element from queue

3. Peek front element

4. Check if queue is empty

5. Display queue

6. Exit

Q-9

list1 = [1, 2, 3, 4, 5, 6, 7]

list2 = [4, 5, 6, 8, 9]

list3 = [num for num in list1 if num not in list2]

print("Numbers from list1 that are not in list2:", list3)

OUTPUT:

Numbers from list1 that are not in list2: [1, 2, 3, 7]