```
"""Aim:WAP to perform following operation on queue (i) Add
(ii) Delete
(iii) Search
(iv) Exit """
# Branch:Comps
# Year:SE
# Sem:IV
# Subject:Python
# Name: Sidra Shaikh
# UIN:231P064
# Roll No:40

class Queue:
    def _init_(self):
        self.queue = []

    def add(self, element):
        self.queue.append(element)
        print(f"Element {element} added to the queue.")

    def delete(self):
        if len(self.queue) == 0:
            print("Queue is empty, cannot delete.")
            return None
        return self.queue.pop(0)

    def search(self, element):
        if element in self.queue:
            return self.queue.index(element)
        return -1

    def display(self):
        if len(self.queue) == 0:
            print("Queue is empty.")
        else:
            print(f"Current queue: {self.queue}")
def menu():
    q = Queue()
    while True:
        print("\nQUEUE OPERATIONS")
        print("1. Add Element")
        print("2. Delete Element")
        print("3. Search Element")
        print("4. Exit")
        try:
            choice = int(input("Enter your choice: "))
        except ValueError:
            print("Invalid input! Please enter a number between 1 and 4.")
            continue
        if choice == 1:
            element = input("Enter element to add: ")
            q.add(element)
        elif choice == 2:
            deleted_element = q.delete()
            if deleted_element is not None:
                print(f"Deleted element: {deleted_element}")
        elif choice == 3:
            element = input("Enter element to search: ")
            position = q.search(element)
            if position == -1:
                print(f"Element {element} not found in the queue.")
            else:
                print(f"Element {element} found at position {position} in the
queue.")
        elif choice == 4:
            print("Exiting program...")
            break
        else:
            print("Invalid choice! Please enter a valid option between 1 and 4.")
        q.display()

menu()


# A Python program to create a linked list and perform operations on the list.
# Branch: Comps
# Year: SE
# Sem: IV
# Subject: Python
# Name: Sidra Shaikh
# UIN: 231P064
# Roll No: 40

# Create an empty linked list
l1 = []

# Add some string type elements to l1
l1.append("Hindustan")
l1.append("Bharat")
l1.append("India")

# Display the list
print("Existing list:", l1)

# Display menu
choice = 0
while choice < 5:
    print("\nLinked List Operations")
    print("1. Add Element")
    print("2. Remove Element")
    print("3. Replace Element")
    print("4. Search Element")
    print("5. Exit")

    choice = int(input("Enter Your Choice: "))

    # Perform task depending on the user's choice
    if choice == 1:
        element = input("Enter Element: ")
        pos = int(input("Enter Position: "))
        l1.insert(pos, element)

    elif choice == 2:
        try:
            element = input("Enter Element: ")
            l1.remove(element)
        except ValueError:
            print("Element not found.")

    elif choice == 3:
        element = input("Enter New Element: ")
        pos = int(input("Enter Position: "))
        if 0 <= pos < len(l1):
            l1.pop(pos)
            l1.insert(pos, element)
        else:
            print("Invalid position.")

    elif choice == 4:
        element = input("Enter Element to Search: ")
        try:
            pos = l1.index(element)
            print("Element found at position:", pos)
        except ValueError:
            print("Element not found.")

    elif choice == 5:
        print("Exiting program...")
        break

    else:
        print("Invalid choice! Please enter a number between 1 and 5.")

    # Display the updated list
    print("List:", l1)


# Aim: Write a Python Program to Reverse a Stack using Recursion.
# Branch: Comps
# Year: SE
# Sem: IV
# Subject: Python
# Name: Sidra Shaikh
# UIN: 231P064
# Roll No: 40

def reverse_stack(stack):
    if stack:
        # Pop the top element
        element = stack.pop()
        # Recursively reverse the rest of the stack
        reverse_stack(stack)
        # Push the popped element to the bottom
        insert_at_bottom(stack, element)

def insert_at_bottom(stack, element):
    if not stack:
        stack.append(element)
    else:
        # Pop and recursively insert
        top = stack.pop()
        insert_at_bottom(stack, element)
        stack.append(top)

# Driver code
stack = list(map(int, input("Enter elements of the stack separated by space: ").split()))
print("Original Stack:", stack)
reverse_stack(stack)
print("Reversed Stack:", stack)
```

```
# Aim: Write a program to implement circular queue.
# Branch: Comps
# Year: SE
# Sem: IV
# Subject: Python
# Name: Sidra Shaikh
# UIN: 231P064
# Roll No: 40

class CircularQueue:
    def _init_(self, size):
        self.size = size
        self.queue = [None] * size
        self.front = self.rear = -1

    def enqueue(self, value):
        if (self.rear + 1) % self.size == self.front:
            print("Queue is full!")
        elif self.front == -1:
            self.front = self.rear = 0
            self.queue[self.rear] = value
        else:
            self.rear = (self.rear + 1) % self.size
            self.queue[self.rear] = value

    def dequeue(self):
        if self.front == -1:
            print("Queue is empty!")
        elif self.front == self.rear:
            self.front = self.rear = -1
        else:
            self.front = (self.front + 1) % self.size

    def display(self):
        if self.front == -1:
            print("Queue is empty!")
        else:
            i = self.front
            while i != self.rear:
                print(self.queue[i], end=" ")
                i = (i + 1) % self.size
            print(self.queue[self.rear])

# Main Program
size = int(input("Enter the size of the queue: "))
cq = CircularQueue(size)

while True:
    print("\n1. Enqueue")
    print("2. Dequeue")
    print("3. Display")
    print("4. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        value = int(input("Enter value to enqueue: "))
        cq.enqueue(value)
    elif choice == 2:
        cq.dequeue()
    elif choice == 3:
        cq.display()
    elif choice == 4:
        print("Exiting program...")
        break
    else:
        print("Invalid choice! Please try again.")


# Aim: limit the number of items printed in output of numpy array
# Branch: Comps
# Year: SE
# Sem: IV
# Subject: Python
# Name: Sidra Shaikh
# UIN: 231P064
# Roll No: 40
import numpy as np
data = np.array([1, 2, 3, 10, 20, 30])
clipped_data = data.clip(2, 10)
print(clipped_data)


# Aim:To get the common items between two python numpy arrays
# Branch: Comps
# Year: SE
# Sem: IV
# Subject: Python
# Name: Sidra Shaikh
# UIN: 231P064
# Roll No: 40
import numpy as np
ar1 = np.array([0, 1, 2, 3, 4])
ar2 = [1, 3, 4]
# Common values between two arrays
print(np.intersect1d(ar1, ar2))


"""Aim:Write a menu driven python program to perform basic mathematical operations
on
two polynomials or integers using numpy."""
# Branch: Comps
# Year: SE
# Sem: IV
# Subject: Python
# Name: Sidra Shaikh
# UIN: 231P064
# Roll No: 40
def add(A, B, m, n):
    size = max(m, n)
    sum_poly = [0 for _ in range(size)]

    for i in range(m):
        sum_poly[i] = A[i]

    for i in range(n):
        sum_poly[i] += B[i]

    return sum_poly

def printPoly(poly, n):
    result = []
    for i in range(n):
        if poly[i] != 0:
            if i == 0:
                result.append(f"{poly[i]}")
            else:
                result.append(f"{poly[i]}x^{i}")

    print(" + ".join(result))

if __name__ == '__main__':
    A = [5, 0, 10, 6]
    B = [1, 2, 4]

    m = len(A)
    n = len(B)

    print("First polynomial is:")
    printPoly(A, m)

    print("Second polynomial is:")
    printPoly(B, n)

    sum_poly = add(A, B, m, n)
    size = max(m, n)

    print("Sum of polynomials is:")
    printPoly(sum_poly, size)


# Aim: WAP to find area of circle, rectangle and triangle using objects and
classes.
# Branch: Comps
# Year: SE
# Sem: IV
# Subject: Python
# Name: Sidra Shaikh
# UIN: 231P064
# Roll No: 40
import math

class Shape:
    def area(self):
        pass  # This is a base class with no implementation
```

```
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius * self.radius

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

class Triangle(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height

    def area(self):
        return 0.5 * self.base * self.height

# Example usage
radius = float(input("Enter radius of the circle: "))
circle = Circle(radius)
print(f"Area of Circle: {circle.area():.2f}")

length = float(input("Enter length of the rectangle: "))
width = float(input("Enter width of the rectangle: "))
rectangle = Rectangle(length, width)
print(f"Area of Rectangle: {rectangle.area():.2f}")

base = float(input("Enter base of the triangle: "))
height = float(input("Enter height of the triangle: "))
triangle = Triangle(base, height)
print(f"Area of Triangle: {triangle.area():.2f}")
```