QUESTION 1:- In python, print the minimum and maximum elements in an array. Display all the elements as well. Enter input from the keyboard

```
arr = []
n = int(input("Enter the number of elements in the array: "))
for i in range(n):
    arr.append(int(input(f"Enter element {i+1}: ")))

print("Array elements:", ' '.join(map(str, arr)))

min_element = arr[0]
max_element = arr[0]
for num in arr:
    if num < min_element:
        min_element = num
    if num > max_element:
        max_element = num

print(f"Minimum element: {min_element}")
print(f"Maximum element: {max_element}")
```

```
Enter the number of elements in the array: 5
Enter element 1: 32
Enter element 2: 54
Enter element 3: 43
Enter element 4: 69
Enter element 5: 99
Array elements: 32 54 43 69 99
Minimum element: 32
Maximum element: 99
```

QUESTION 2: WAP in python to implement array(list) & perform the following operations. Insert (using i/p), Display and sorting.

```
def array_operations():
    arr = []
    while True:
        print("Choose an operation:")
        print("1. Insert element")
        print("2. Display array")
        print("3. Sort array")
        print("4. Exit")

        choice = input("Enter your choice: ")

        if choice == '1':
            try:
                element = int(input("Enter the element to insert: "))
                arr.append(element)
                print("Element inserted successfully!")
            except ValueError:
                print("Invalid input. Please enter an integer.")
        elif choice == '2':
            if not arr:
                print("Array is empty.")
            else:
                print("Array elements:", arr)
        elif choice == '3':
            if not arr:
                print("Array is empty.")
            else:
                arr.sort()
                print("Array sorted successfully!")
        elif choice == '4':
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    array_operations()
```

```
Choose an operation:
1. Insert element
2. Display array
3. Sort array
4. Exit
Enter your choice: 1
Enter the element to insert: 56
Element inserted successfully!
Choose an operation:
1. Insert element
2. Display array
```

```
3. Sort array
4. Exit
Enter your choice: 1
Enter the element to insert: 88
Element inserted successfully!
Choose an operation:
1. Insert element
2. Display array
3. Sort array
4. Exit
Enter your choice: 1
Enter the element to insert: 32
Element inserted successfully!
Choose an operation:
1. Insert element
2. Display array
3. Sort array
4. Exit
Enter your choice: 1
Enter the element to insert: 69
Element inserted successfully!
Choose an operation:
1. Insert element
2. Display array
3. Sort array
4. Exit
Enter your choice: 1
Enter the element to insert: 99
Element inserted successfully!
Choose an operation:
1. Insert element
2. Display array
3. Sort array
4. Exit
Enter your choice: 2
Array elements: [56, 88, 32, 69, 99]
Choose an operation:
1. Insert element
2. Display array
3. Sort array
4. Exit
Enter your choice: 3
Array sorted successfully!
Choose an operation:
1. Insert element
2. Display array
3. Sort array
```

QUESTION 3: WAP to implement stack using list and perform PUSH, POP, TOS AND DISPLAY

```
stack = []

def push(element):
  stack.append(element)
  print(f"{element} pushed to the stack.")

def pop():
  if not stack:
    print("Stack underflow!")
    return None
  else:
    popped_element = stack.pop()
    print(f"{popped_element} popped from the stack.")
    return popped_element

def tos():
  if not stack:
    print("Stack is empty.")
    return None
  else:
    print(f"Top of the stack: {stack[-1]}")
    return stack[-1]

def display():
  if not stack:
    print("Stack is empty.")
  else:
    print("Stack elements:", stack)

while True:
  print("\nChoose an operation:")
  print("1. Push")
  print("2. Pop")
  print("3. Top of Stack (TOS)")
  print("4. Display")
  print("5. Exit")
```

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

if choice == '1':
  try:
      element = int(input("Enter the element to push: "))
      push(element)
  except ValueError:
      print("Invalid input. Please enter an integer.")
elif choice == '2':
  pop()
elif choice == '3':
  tos()
elif choice == '4':
  display()
elif choice == '5':
  break
else:
  print("Invalid choice. Please try again.")
```

```
Choose an operation:
1. Push
2. Pop
3. Top of Stack (TOS)
4. Display
5. Exit
Enter your choice: 1
Enter the element to push: 34
34 pushed to the stack.

Choose an operation:
1. Push
2. Pop
3. Top of Stack (TOS)
4. Display
5. Exit
Enter your choice: 1
Enter the element to push: 99
99 pushed to the stack.

Choose an operation:
1. Push
2. Pop
3. Top of Stack (TOS)
4. Display
5. Exit
Enter your choice: 1
Enter the element to push: 45
45 pushed to the stack.

Choose an operation:
1. Push
2. Pop
3. Top of Stack (TOS)
4. Display
5. Exit
Enter your choice: 1
Enter the element to push: 46
46 pushed to the stack.

Choose an operation:
1. Push
2. Pop
3. Top of Stack (TOS)
4. Display
5. Exit
Enter your choice: 1
Enter the element to push: 79
79 pushed to the stack.

Choose an operation:
1. Push
2. Pop
3. Top of Stack (TOS)
4. Display
5. Exit
Enter your choice: 1
```

QUESTION 4: WAP to implement Queue using list and perform the following operations - Enqueue, Dequeue,Display

```
queue = []

def enqueue(element):
  queue.append(element)
  print(f"{element} enqueued to the queue.")
```

```
def dequeue():
  if not queue:
    print("Queue underflow!")
    return None
  else:
    dequeued_element = queue.pop(0)
    print(f"{dequeued_element} dequeued from the queue.")
    return dequeued_element

def display():
  if not queue:
    print("Queue is empty.")
  else:
    print("Queue elements:", queue)

while True:
  print("\nChoose an operation:")
  print("1. Enqueue")
  print("2. Dequeue")
  print("3. Display")
  print("4. Exit")

  choice = input("Enter your choice: ")

  if choice == '1':
    try:
        element = int(input("Enter the element to enqueue: "))
        enqueue(element)
    except ValueError:
        print("Invalid input. Please enter an integer.")
  elif choice == '2':
    dequeue()
  elif choice == '3':
    display()
  elif choice == '4':
    break
  else:
    print("Invalid choice. Please try again.")
```

```
Choose an operation:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the element to enqueue: 99
99 enqueued to the queue.

Choose an operation:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the element to enqueue: 76
76 enqueued to the queue.

Choose an operation:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the element to enqueue: 35
35 enqueued to the queue.

Choose an operation:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the element to enqueue: 72
72 enqueued to the queue.

Choose an operation:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 42
Invalid choice. Please try again.

Choose an operation:
```

```
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the element to enqueue: 18
18 enqueued to the queue.

Choose an operation:
1. Enqueue
2. Dequeue
```

QUESTION 5: WAP to implement Graph using Adjacency list and Adjacency matrix & perform : Traverse the graph using BFS, Traverse the graph using DFS.

```
from collections import defaultdict

#The first argument provides the initial value for the default_factory attribute; it defaults to None. All remaining arguments are treat

#Therefore, if you just write defaultdict without passing any value to the constructor, the default value is set to None

class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = defaultdict(list)  # Adjacency List
        self.matrix = [[0] * vertices for _ in range(vertices)]  # Adjacency Matrix

    def add_edge(self, u, v):
        self.graph[u].append(v)
        self.graph[v].append(u)  # For undirected graph
        self.matrix[u][v] = 1
        self.matrix[v][u] = 1  # For undirected graph

    def BFS(self, s):
        visited = [False] * self.V
        queue = [s]
        visited[s] = True

        while queue:
            u = queue.pop(0)
            print(u, end=" ")

            for v in self.graph[u]:
                if not visited[v]:
                    visited[v] = True
                    queue.append(v)
        print()

    def DFS(self, s, visited):
        visited[s] = True
        print(s, end=" ")

        for v in self.graph[s]:
            if not visited[v]:
                self.DFS(v, visited)
# Example usage
g = Graph(4)
g.add_edge(0, 1)
g.add_edge(0, 2)
g.add_edge(1, 2)
g.add_edge(2, 0)
g.add_edge(2, 3)
g.add_edge(3, 3)

print("Breadth-First Traversal (starting from vertex 2):")
g.BFS(2)

print("Depth-First Traversal (starting from vertex 2):")
visited = [False] * 4
g.DFS(2, visited)
print()
```

```
Breadth-First Traversal (starting from vertex 2):
2 0 1 3
Depth-First Traversal (starting from vertex 2):
2 0 1 3
```