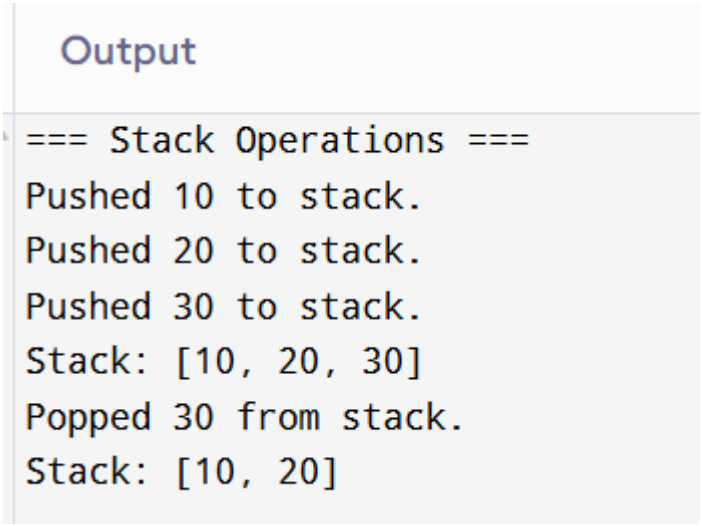
ASSIGNMENT

DAY 7 PRACTICE PROGRAMS

1.Write a program to implement the stack & queue data structure using list

```
class Stack:
    def __init__(self):
        self.stack = []

    def push(self, item):
        self.stack.append(item)
        print(f"Pushed {item} to stack.")

    def pop(self):
        if not self.is_empty():
            item = self.stack.pop()
            print(f"Popped {item} from stack.")
            return item
        else:
            print("Stack is empty.")
            return None

    def peek(self):
        if not self.is_empty():
            return self.stack[-1]
        return None

    def is_empty(self):
        return len(self.stack) == 0

    def display(self):
```

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


# Example usage of Stack
print("=== Stack Operations ===")
s = Stack()
s.push(10)
s.push(20)
s.push(30)
s.display()
s.pop()
s.display()
```



2.Write a program that prints all consonants in a string using list comprehension

```
# Input string
text = input("Enter a string: ")


# List comprehension to get all consonants
consonants = [char for char in text if char.lower() in 'bcdfghjklmnpqrstvwxyz' and char.isalpha()]


# Output
print("Consonants in the string:", consonants)
```

```
Output
Enter a string: kavya
Consonants in the string: ['k', 'v', 'y']
```

3.Write a program that creates a list of numbers from 1-50 that are either divisible by 3 or divisible by 6.

```
# Create list using list comprehension
numbers = [num for num in range(1, 51) if num % 3 == 0 or num % 6 == 0]

# Output the list
print("Numbers divisible by 3 or 6 from 1 to 50:")
print(numbers)
```

```
Output
Numbers divisible by 3 or 6 from 1 to 50:
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48]
```

4.Write a Python program to remove the intersection of a 2nd set from the 1st set.

```
# Define the two sets
set1 = {1, 2, 3, 4, 5, 6}
set2 = {4, 5, 6, 7, 8, 9}

# Remove intersection elements of set2 from set1
set1 = set1 - (set1 & set2)  # or use set1.difference_update(set2)

# Output the result
print("Set after removing intersection with second set:", set1)
```

```
Output
Set after removing intersection with second set: {1, 2, 3}
```

5.Write a Python program to remove an item from a set if it is present in the set

```
# Define a set
my_set = {10, 20, 30, 40, 50}

# Item to remove
item_to_remove = 30

# Remove the item if present
my_set.discard(item_to_remove)  # discard() does nothing if the item is not in the set

# Output the updated set
print("Updated set:", my_set)
```

```
Output
Updated set: {50, 20, 40, 10}
```

6.Write a Python program to create a symmetric difference.

```
# Define two sets
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

# Find symmetric difference
sym_diff = set1.symmetric_difference(set2)

# Output the result
```

```
print("Symmetric difference:", sym_diff)
```

```
Output

Symmetric difference: {1, 2, 3, 6, 7, 8}
```

7.Write a Python program to get the 4th element and 4th element from last of a tuple.

```
# Define a tuple
my_tuple = (10, 20, 30, 40, 50, 60, 70, 80, 90)

# Get the 4th element (index 3)
fourth_element = my_tuple[3]

# Get the 4th element from the last (index -4)
fourth_from_last = my_tuple[-4]

# Output the result
print("4th element:", fourth_element)
print("4th element from last:", fourth_from_last)
```

```
Output

4th element: 40
4th element from last: 60
```

8.Write a Python program to find the repeated items of a tuple.

```
# Define a tuple
my_tuple = (10, 20, 30, 10, 40, 50, 20, 60, 10)

# Create a dictionary to count occurrences of each element
count_dict = {}
```

```
# Count the occurrences of each element in the tuple
for item in my_tuple:
    count_dict[item] = count_dict.get(item, 0) + 1

# Find the repeated items (occurrence > 1)
repeated_items = [item for item, count in count_dict.items() if count > 1]

# Output the repeated items
print("Repeated items in the tuple:", repeated_items)
```

Output

```
Repeated items in the tuple: [10, 20]
```

9.Write a Python program to check whether an element exists within a tuple

```
# Define a tuple
my_tuple = (10, 20, 30, 40, 50)

# Element to check
element = 30

# Check if the element exists in the tuple
if element in my_tuple:
    print(f"Element {element} exists in the tuple.")
else:
    print(f"Element {element} does not exist in the tuple.")
```

Output

```
Element 30 exists in the tuple.
```