3. Challenge: Push ["X", "Y", "Z"], pop twice, push "W"

Algorithmic Sequence with Code Explanation:

1. Initialize empty stack

challenge_stack = []

Explanation: Stack container created.

2. Push "X"

challenge_stack.append("X")

Explanation: X is at the bottom.

3. Push "Y"

challenge_stack.append("Y")

Explanation: Y is on top of X.

4. Push "Z"

challenge_stack.append("Z")

Explanation: Z is now top.

5. Pop twice

challenge_stack.pop() # removes Z

challenge_stack.pop() # removes Y

Explanation: Top elements removed, leaving ["X"].

6. Push "W"

challenge_stack.append("W")

Explanation: W is placed on top → final stack ["X", "W"].

7. Check top element

print(challenge_stack[-1])

Top element now: "W"

4. Reflection: Why Stack is not good for serving people in order

A stack uses LIFO (Last In, First Out).

This means the last person to arrive is served first.

In real-world queues (banks, MoMo agents, clinics), this is unfair, because earlier arrivals wait longer while late arrivals cut ahead.

Queues (FIFO) are better for fairness in such situations. Practical (Rwanda) 1: MoMo Stack

Code + Output (screenshot-style):

```
# MoMo stack

momo_stack = []

momo_stack.append("Enter Code")

momo_stack.append("Enter PIN")

momo_stack.append("Confirm")

print("MoMo Stack (before pop):", momo_stack)

# Pop last step

momo_stack.pop()

print("MoMo Stack (after pop):", momo_stack)
```

Output:

MoMo Stack (before pop): ['Enter Code', 'Enter PIN', 'Confirm']

MoMo Stack (after pop): ['Enter Code', 'Enter PIN']

Answer: Left in the stack → ["Enter Code", "Enter PIN"]

2. Practical (Rwanda) 2: UR Stack

Code + Output (screenshot-style)

```
# UR stack

ur_stack = []

ur_stack.append("Attend Class")

ur_stack.append("Write Assignment")

ur_stack.append("Sit Exam")

print("UR Stack (before undo):", ur_stack)

# Undo two

ur_stack.pop()

ur_stack.pop()

print("UR Stack (after undo):", ur_stack)
```

Output:

UR Stack (before undo): ['Attend Class', 'Write Assignment', 'Sit Exam']

UR Stack (after undo): ['Attend Class']

Answer: Left in the stack → ["Attend Class"]

3. Challenge: Push ["X", "Y", "Z"], pop twice, push "W"

Algorithmic Sequence with Code Explanation:

1. Initialize empty stack

challenge_stack = []

Explanation: Stack container created.

2. Push "X"challenge_stack.append("X")

Explanation: X is at the bottom.

3. Push "Y"

challenge_stack.append("Y")

Explanation: Y is on top of X.

4. Push "Z"

challenge_stack.append("Z")

Explanation: Z is now top.

5. Pop twicechallenge_stack.pop() # removes Z

challenge_stack.pop() # removes Y

Explanation: Top elements removed, leaving ["X"].

6. Push "W"challenge_stack.append("W")

Explanation: W is placed on top → final stack ["X", "W"].

7. Check top element

print(challenge_stack[-1])

Top element now: "W"

4. Reflection: Why Stack is not good for serving people in order

A stack uses LIFO (Last In, First Out).

This means the last person to arrive is served first.

In real-world queues (banks, MoMo agents, clinics), this is unfair, because earlier arrivals wait longer while late arrivals cut ahead

Queues (FIFO) are better for fairness in such situations.