/*Implementing a real-time undo/redo system for a text editing application using a Stack data structure. The system should support the following operations:
• Make a Change: A new change to the document is made.
• Undo Action: Revert the most recent change and store it for potential redo.
• Redo Action: Reapply the most recently undone action.
• Display Document State: Show the current state of the document after undoing or redoing an action*/

```
class TextEditor:

    def __init__(self):

        self.document = []          # List to store each line of the document

        self.undo_stack = []        # Stack to store previous states for Undo

        self.redo_stack = []        # Stack to store undone states for Redo


    # Add a new line/text to the document

    def make_change(self, change):

        self.undo_stack.append(self.document.copy())  # Save current state for Undo

        self.document.append(change)              # Add new line/text

        self.redo_stack.clear()                   # Clear Redo stack after a new change

        print(f"\nChange Made: '{change}'")


    # Undo the last change

    def undo(self):

        if not self.undo_stack:

            print("\nNothing to Undo!")

            return

        self.redo_stack.append(self.document.copy())  # Save current state for Redo

        self.document = self.undo_stack.pop()       # Revert to the previous state

        print("\nUndo performed.")
```

```
    # Redo the last undone change
    def redo(self):
        if not self.redo_stack:
            print("\nNothing to Redo!")
            return
        self.undo_stack.append(self.document.copy())      # Save current state for Undo
        self.document = self.redo_stack.pop()             # Reapply the undone change
        print("\nRedo performed.")


# Display the current document
def display_state(self):
    print("\n--- Current Document ---")
    if not self.document:
        print("[Empty Document]")
    else:
        for i in range(len(self.document)):    # loop over index numbers
            print(f"{i+1}: {self.document[i]}")   # i+1 for line number, document[i] for text
    print("-------------------------")
```

```
# ---------- Menu Driven Code ----------
editor = TextEditor()
while True:
    # Display menu
    print("\n--- Menu ---")
    print("1. Add Line/Text")
    print("2. Undo")
    print("3. Redo")
    print("4. Show Document")
    print("5. Exit")
    choice = input("Enter your choice: ")
    if choice == '1':
        text = input("Enter line/text to add: ")
        editor.make_change(text)  # Add new line/text
    elif choice == '2':
        editor.undo()           # Undo last change
    elif choice == '3':
        editor.redo()           # Redo last undone change
    elif choice == '4':
        editor.display_state()    # Display current document
    elif choice == '5':
        print("Exiting...")
        break
    else:
        print("Invalid choice! Try again.")
```

Output:

--- Menu ---

1. Add Line/Text

2. Undo

3. Redo

4. Show Document

5. Exit

Enter your choice: 1

Enter line/text to add: Hello

Change Made: 'Hello'

Enter your choice: 1

Enter line/text to add: World

Change Made: 'World'

Enter your choice: 4

--- Current Document ---

1: Hello

2: World

Enter your choice: 2

Undo performed.

Enter your choice: 4

--- Current Document ---

1: Hello

Enter your choice: 3

Redo performed.

Enter your choice: 4

--- Current Document ---

1: Hello

2: World
