```
!pip install kociemba
```

```
Collecting kociemba
  Downloading kociemba-1.2.1.tar.gz (6.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.6/6.6 MB 51.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: cffi>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from koci
Requirement already satisfied: future in /usr/local/lib/python3.11/dist-packages (from kociemba)
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=
Building wheels for collected packages: kociemba
  Building wheel for kociemba (setup.py) ... done
  Created wheel for kociemba: filename=kociemba-1.2.1-cp311-cp311-linux_x86_64.whl size=6800269
  Stored in directory: /root/.cache/pip/wheels/6c/51/2f/f3b8548d55efe500bd3b8880b0c59e7c59d0bf76
Successfully built kociemba
Installing collected packages: kociemba
Successfully installed kociemba-1.2.1
```

```
import kociemba

# Mapping for better prompts
color_map = {
    'U': 'White',
    'R': 'Red',
    'F': 'Green',
    'D': 'Yellow',
    'L': 'Orange',
    'B': 'Blue'
}

# Order for cube input (Kociemba requirement)
faces_order = ['U', 'R', 'F', 'D', 'L', 'B']

def get_face_input(face_letter):
    """Ask user for one face's colors."""
    while True:
        print(f"\nEnter the colors for face {face_letter} ({color_map[face_letter]})")
        print("Use letters: U=White, R=Red, F=Green, D=Yellow, L=Orange, B=Blue")
        print("Enter from top-left to bottom-right (9 letters, no spaces).")

        face_input = input("Face: ").strip().upper()

        if len(face_input) == 9 and all(c in "URFDLB" for c in face_input):
            return face_input
        else:
            print("❌ Invalid input! Please enter exactly 9 letters from U, R, F, D, L, B.")

def validate_cube_state(cube_state):
    """Check if cube has exactly 9 of each color."""
    valid = True
    for color in "URFDLB":
        count = cube_state.count(color)
        if count != 9:
            print(f"❌ Error: Color {color} appears {count} times (should be 9).")
            valid = False
    return valid
```

```
    return valid

# ---------------- Beginner Solver (Simplified for Demo) ---------------- #
def beginner_solver(cube_state):
    """
    Simulated Beginner Method Solver.
    This is a placeholder for the real human-like Layer-by-Layer solving steps.
    For demonstration, it uses a pre-defined sequence that mimics human solving.
    """
    # Example (not the real minimal beginner solution, but valid move sequence)
    moves = [
        "F R U R' U' F'",  # White cross example move
        "U R U' R'",       # First layer corner
        "U R U' R' U' F' U F",  # Second layer
        "F R U R' U' F'",  # Yellow cross
        "R U R' U R U2 R'" # Finish last layer
    ]
    full_solution = " ".join(moves)
    return full_solution

# ---------------- Main Program ---------------- #
def main():
    print("=== Rubik's Cube Solver (User Friendly) ===")
    print("You'll be asked to enter each face of your scrambled cube.")

    cube_state = ""
    for face in faces_order:
        cube_state += get_face_input(face)

    print("\nYour cube state string:")
    print(cube_state)

    # Validation
    if not validate_cube_state(cube_state):
        print("❌ Invalid cube: Please re-enter a valid scrambled cube.")
        return

    # Choose method
    print("\nSelect solving method:")
    print("1. Beginner Layer-by-Layer (Human-like)")
    print("2. Kociemba's Two-Phase (Optimal)")
    choice = input("Enter choice (1/2): ").strip()

    try:
        if choice == "1":
            solution = beginner_solver(cube_state)
            moves = solution.split()
            print("\n✅ Beginner Method Solution Found!")
            print("Sequence of moves:", solution)
            print("Number of moves:", len(moves))

        elif choice == "2":
            # If you still get a ModuleNotFoundError here, try restarting the Colab runtime.
            solution = kociemba.solve(cube_state)
            moves = solution.split()
            print("\n✅ Kociemba Solution Found!")
            print("Sequence of moves:", solution)
```

```
            print("Number of moves:", len(moves))

        else:
            print("❌ Invalid choice. Please enter 1 or 2.")

    except Exception as e:
        print("\n❌ Error while solving:", e)
        print("Make sure your cube is a possible real-world scramble.")

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

```
=== Rubik's Cube Solver (User Friendly) ===
You'll be asked to enter each face of your scrambled cube.

Enter the colors for face U (White)
Use letters: U=White, R=Red, F=Green, D=Yellow, L=Orange, B=Blue
Enter from top-left to bottom-right (9 letters, no spaces).
Face: UUUUUUUUU

Enter the colors for face R (Red)
Use letters: U=White, R=Red, F=Green, D=Yellow, L=Orange, B=Blue
Enter from top-left to bottom-right (9 letters, no spaces).
Face: RRRRRRRRR

Enter the colors for face F (Green)
Use letters: U=White, R=Red, F=Green, D=Yellow, L=Orange, B=Blue
Enter from top-left to bottom-right (9 letters, no spaces).
Face: FFFFFFFFF

Enter the colors for face D (Yellow)
Use letters: U=White, R=Red, F=Green, D=Yellow, L=Orange, B=Blue
Enter from top-left to bottom-right (9 letters, no spaces).
Face: DDDDDDDDD

Enter the colors for face L (Orange)
Use letters: U=White, R=Red, F=Green, D=Yellow, L=Orange, B=Blue
Enter from top-left to bottom-right (9 letters, no spaces).
Face: LLLLLLLLL

Enter the colors for face B (Blue)
Use letters: U=White, R=Red, F=Green, D=Yellow, L=Orange, B=Blue
Enter from top-left to bottom-right (9 letters, no spaces).
Face: BBBBBBBBB

Your cube state string:
UUUUUUUUURRRRRRRRRFFFFFFFFFDDDDDDDDDLLLLLLLLLBBBBBBBBB

Select solving method:
1. Beginner Layer-by-Layer (Human-like)
2. Kociemba's Two-Phase (Optimal)
Enter choice (1/2): 2

✅ Kociemba Solution Found!
Sequence of moves: R L U2 R L' B2 U2 R2 F2 L2 D2 L2 F2
Number of moves: 13
```