Securein Assessment

PART -B

CODE

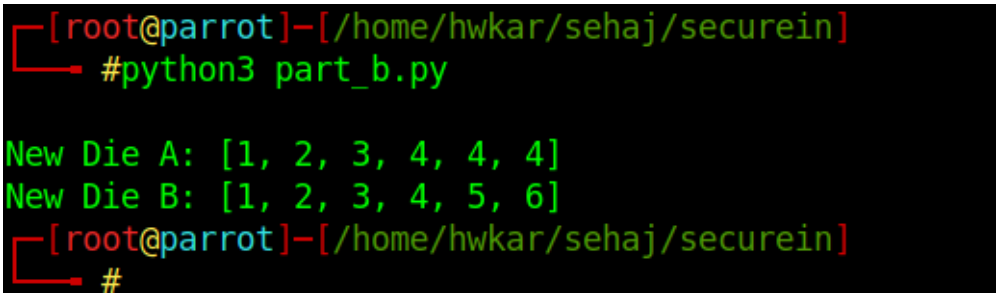
```
def undoom_dice(die_a, die_b):
    scaling_factor = sum(die_a) / sum(die_b)  # Calculate scaling factor
    new_die_a = [min(4, spots) for spots in die_a]  # Limit Die A spots to 4
    new_die_b = [min(6, round(spots * scaling_factor)) for spots in die_b]  # Scale and limit
Die B spots
    return new_die_a, new_die_b

# Define initial dice
die_a = [1, 2, 3, 4, 5, 6]
die_b = die_a

# Undoom the dice
new_die_a, new_die_b = undoom_dice(die_a, die_b)

# Print the results
print("\nNew Die A:", new_die_a)
print("New Die B:", new_die_b)
```

OUTPUT



Explantion

1. Defining a Function:

- `undoom_dice(die_a, die_b)`: This function takes two dice (represented as lists of numbers) and modifies them to "undoom" them.

2. Balancing the Dice:

- `scaling_factor = sum(die_a) / sum(die_b)`: Calculates a ratio to balance the overall values of the dice.
- `new_die_a = [min(4, spots) for spots in die_a]`: Limits each side of Die A to a maximum of 4.
- `new_die_b = [min(6, round(spots * scaling_factor)) for spots in die_b]`: Scales Die B's sides based on the scaling factor and limits them to a maximum of 6.

3. Returning the Modified Dice:

- `return new_die_a, new_die_b`: Sends back the transformed versions of Die A and Die B.

4. Example Usage:

- `die_a = [1, 2, 3, 4, 5, 6]`: Creates a typical 6-sided die.
- `die_b = die_a`: Copies Die A to create a second die.
- `new_die_a, new_die_b = undoom_dice(die_a, die_b)`: Calls the function to tweak the dice.
- `print("\nNew Die A:", new_die_a)`: Displays the modified Die A.
- `print("New Die B:", new_die_b)`: Displays the modified Die B.