Assignment:

Simple agent where considering 7x7 dimensional area

Course Name: Artificial Intelligence

Submitted by:

Sojun Chandra Shil
Student ID: 210215
Computer Science & Engineering Discipline,
Khulna University

Submitted to:

Dr. Amit Kumar Mondal
Associate Professor
Computer Science & Engineering Discipline,
Khulna University,
Khulna.

Theoretical Analysis

The goal of the code is to simulate a random process where two values (`a` and `b`) are chosen randomly, and the process continues until one of the values is equal to 1 or 7. The number of iterations is tracked and reported as a percentage of "performance."

> **Time Complexity**: The algorithm's time complexity is indeterminate because it depends on random numbers. In the best case, it stops after one iteration (if `a` or `b` is initialized to 1 or 7). In the worst case, it could theoretically run indefinitely if the random number generator never produces 1 or 7.
> **Space Complexity**: The space complexity is constant, O(1), because the only variables stored are `a`, `b`, and `count`.

Data Structure

> **Integers (`a`, `b`, `count`)**:
> `a` and `b`: Hold randomly generated values.
> `count`: Tracks the number of iterations.

My code doesn't use any complex data structures. It uses simple integers, making the memory requirement minimal.

Algorithm to Function/Method Representation

The code can be encapsulated into a function for modularity and reusability. Here's how the function might look:

```
import random

def performance_simulation():
    count = 0
    a = random.randint(2, 7)
    b = random.randint(2, 7)
    while True:
        if a == 1 or a == 7 or b == 1 or b == 7:
            return count * 100
        count += 1
        a = random.randint(1, 8)
        b = random.randint(1, 8)
```

This function can be called to execute the random process and return the final performance value.

Implementation

The implementation of the algorithm is simple and relies on the `random.randint()` function to generate random integers. Here's a more detailed implementation:

1. Initialization:

Initialize `count` to 0, which will keep track of how many times the loop runs.
Generate initial random values for `a` and `b` within the range of 2 to 7 (inclusive).

2. Looping Condition:

The loop runs indefinitely using `while True`.

Inside the loop, the values of `a` and `b` are checked.

If `a == 1`, `a == 7`, `b == 1`, or `b == 7`, the loop stops and prints the result.

If neither condition is met, the `count` is incremented, and new random values for `a` and `b` are generated.

3. Termination:

The loop terminates when either `a` or `b` is 1 or 7, and the performance percentage is printed or returned.

Input Test Cases Format

Since the code uses randomly generated values, there is no direct input from the user. However, we can assume different random number ranges to simulate test cases. In the function form, the inputs are internally generated and require no explicit format.

Output Format

The output format is a simple percentage that represents the performance based on how many iterations were performed before the loop terminated.

Example Output:

Case 1: Random values generate `a=7` or `b=1` immediately

performance: 0 %

Case 2: The loop runs for 5 iterations before `a=1` is generated

performance: 500 %