Lab 2 - Red Dragon Inn D&D

Introduction

This app helps players calculate probabilities and expected values related to dice rolls in two scenarios:

Scenario A: Calculate the expected number of attempts to roll a single dice until a specified value or higher is obtained.

Scenario B: Determine the probability of rolling multiple dice and having at least one dice show a value greater than or equal to a desired value.

Using the code

For Scenario A: Expected number of roll attempts:

- Set the value of ‘n’ to represent the number of sides on your dice.
- Set the value of ‘x’ to represent the minimum face value you're interested in.
- Call the function with the parameters, setting the scenario to 'A'.

Example for scenario A:

To calculate the expected number of attempts to get a 5 or higher on a standard 6 sided dice:

```
# Input desired data
n = 6
x = 5

# Calculate the expected number of rolls to achieve a score of 'n' or higher
expectedRolls = computeDiceStatistics(None, n, x, 'A')
print(f"Expected number of rolls to get a score of {x} or higher on a {n}-sided die: {expectedRolls:.2f}")
```

```
Expected number of rolls to get a score of 5 or higher on a 6-sided die: 3.00
```

For Scenario B: Probability with multiple dice:

- Set the value of ‘m’ to represent the number of dice you are rolling.
- Set the value of ‘n’ to represent the number of sides on your dice.
- Set the value of ‘x’ to represent the minimum face value you're interested in.
- Call the function with the parameters, setting scenario to 'B'.

Example for scenario B:

To calculate the probability of getting a 5 or higher when rolling two 6 sided dice:

```
#....for rolling the die 4 times
m = 4
probabilitySuccess = computeDiceStatistics(m, n, x, 'B')
print(f"Probability of getting a score of {x} or higher when r
```

```
Probability of getting a score of 5 or higher when rolling the 6-sided die 2 times: 0.5556 or 55.56%
```

The results

For Scenario A

The result gives you the average or expected number of attempts you need to achieve the desired result. A result of 3 means you'd need 3 attempts.

For Scenario B

The result provides the probability of achieving the desired result in the given number of dice rolls. A result of 0.50 or 50% means there's a half chance of getting the desired value or higher in the rolls.

Walkthrough

A player needs to roll a 20 sided dice and hopes to get a score of 17 or higher. Therefore:

A: On average, how many attempts are expected to score 17 or higher?
B: If they roll the dice a maximum of 4 times, what's the probability they'll get a score of 17 or higher at least once?

Step 1 - Set Up the Parameters

n: The number of sides on the dice.
x: The minimum score the player wants to achieve.

For situation B

m: The number of attempts or dice rolls.

Step 2 - Calculate the Expected Number of Rolls

Utilize the function ‘computeDiceStatistics’ to calculate how many rolls, on average, a player can expect to make before getting a score of 17 or higher on the dice.

Explanation

The app calculates the average number of attempts to achieve the desired result. This will help the player decide whether they should expend resources for rerolls.

```
# Input desired data
n = 20
x = 17

# Calculate the expected number of rolls to achieve a sc
expectedRolls = computeDiceStatistics(None, n, x, 'A')
print(f"Expected number of rolls to get a score of {x} o
```

```
Expected number of rolls to get a score of 17 or higher on a 20-sided die: 5.00
```

Step 3 - Calculate the Probability of Success in 4 Rolls

If the player opts for a maximum of 4 rolls, we want to know the probability they have to achieve their desired score **at least once** in those rolls.

Explanation

The code calculates the likelihood of getting a score of 17 or higher in any of the 4 rolls. **A higher probability suggests a good chance** of achieving the desired result.

```
#....for rolling the die 4 times
m = 4
probabilitySuccess = computeDiceStatistics(m, n, x, 'B')
print(f"Probability of getting a score of {x} or higher when
```

```
Probability of getting a score of 17 or higher when rolling the 20-sided die 4 times: 0.5904 or 59.04%
```

Code:

```
# Global inputs:
# m: The number of dice rolls or the number of dice being rolled (relevant only in Scenario
B).
# n: The number of sides on each dice.
# x: The target value. A (calculates the expected number of rolls to achieve a score of 'x' or
higher). B (computes the probability of achieving a value of 'x' or higher when rolling 'm'
dice ).

# Dice Statistics:
# P: The probability of rolling a value 'x' or higher on a single roll of an 'n' sided dice.
# PNot: The probability P' of NOT rolling a value 'x' or higher.
# PNotHigher: The probability of NOT rolling 'x' or higher on any of the 'm' dice.

def computeDiceStatistics(m, n, x, scenario):

    # Input validation
    if scenario not in ['A', 'B']:
        raise ValueError("Scenario should be either 'A' or 'B'")

    #Scenario A Calculating the expected number of roll attempts with a n-sided dice until a
face showing a value of x or higher is observed.
    if scenario == 'A':
        P = (n - x + 1) / n
        return 1 / P

    #Scenario B Determining the probability of rolling m dice, each with n sides, and having
at least one dice show a value greater than or equal to x.
    else:
        PNot = (x - 1) / n
        PNotHigher = PNot**m
        return 1 - PNotHigher

# Input desired data
n = 20
x = 17

# Calculate the expected number of rolls to achieve a score of 'n' or higher
expectedRolls = computeDiceStatistics(None, n, x, 'A')
print(f"Expected number of rolls to get a score of {x} or higher on a {n}-sided dice:
{expectedRolls:.2f}")

#....for rolling the dice 4 times
m = 4
probabilitySuccess = computeDiceStatistics(m, n, x, 'B')
print(f"Probability of getting a score of {x} or higher when rolling the {n}-sided dice {m}
times: {probabilitySuccess:.4f} or {probabilitySuccess*100:.2f}%")
```