**USE CASE IMPLEMENTATION**

**AIM**: To create an efficient and flexible digital tool that generates true random dice rolls for gaming, probability experiments, and educational purposes, replicating and expanding upon the functionality of physical dice in a virtual environment.​ This function returns a list of integers representing the outcome of rolling num\_dice dice, each with sides sides. Let me know if you want to extend this for batch rolls, weighted probabilities, or graphical output.

**Algorithm:**

1. Start the application and prompt the user for the number of dice, sides per die, and trials.​
2. Validate user inputs to ensure they are positive integers.​
3. Use NumPy to generate random integers for each dice roll over the specified number of trials.​
4. Store all roll results in an array for further analysis.​
5. Display individual roll results for a selected trial.​
6. Optionally, represent results using ASCII art per die roll.​
7. Analyze roll data to calculate frequency and distributions.​
8. Visualize distributions using a histogram for user interpretation.

**PROGRAM:**

import numpy as np

import matplotlib.pyplot as plt

def roll\_dice(num\_dice=2, sides=6):

return np.random.randint(1, sides + 1, size=num\_dice)

def batch\_rolls(num\_dice, sides, num\_trials):

return np.random.randint(1, sides + 1, size=(num\_trials, num\_dice))

def print\_dice\_ascii(rolls):

DICE\_ART = {

1: ("┌─────────┐","│ │","│ ● │","│ │","└─────────┘"),

2: ("┌─────────┐","│ ● │","│ │","│ ● │","└─────────┘"),

3: ("┌─────────┐","│ ● │","│ ● │","│ ● │","└─────────┘"),

4: ("┌─────────┐","│ ● ● │","│ │","│ ● ● │","└─────────┘"),

5: ("┌─────────┐","│ ● ● │","│ ● │","│ ● ● │","└─────────┘"),

6: ("┌─────────┐","│ ● ● │","│ ● ● │","│ ● ● │","└─────────┘"),

}

lines = [''] \* 5

for roll in rolls:

art = DICE\_ART.get(roll, DICE\_ART[1])

for i, line in enumerate(art):

lines[i] += line + ' '

print('\n'.join(lines))

def show\_histogram(all\_rolls, sides):

plt.hist(all\_rolls.flatten(), bins=range(1, sides + 2), align='left', rwidth=0.8)

plt.xlabel('Result')

plt.ylabel('Frequency')

plt.title('Histogram of Dice Rolls')

plt.xticks(range(1, sides + 1))

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

sides = int(input("Sides per die: "))

num\_dice = int(input("Number of dice: "))

num\_trials = int(input("Number of trials: "))

all\_rolls = batch\_rolls(num\_dice, sides, num\_trials)

print(f"\nSample rolls (first trial):")

print\_dice\_ascii(all\_rolls[0])

show\_histogram(all\_rolls, sides)

**Sample Output:**

* User enters:  
  Number of dice = 3  
  Sides per die = 6  
  Number of trials = 5​
* Rolls generated (first trial):  
  Dice results:​
* ASCII Art display:

text

┌─────────┐ ┌─────────┐ ┌─────────┐

│ ● ● │ │ ● │ │ ● ● │

│ │ │ │ │ ● ● │

│ ● ● │ │ ● │ │ ● ● │

└─────────┘ └─────────┘ └─────────┘

* Histogram of all rolls:
  + Shows frequency of each outcome (e.g., how many 1s, 2s, …, 6s rolled across all trials).​
* Frequency table:  
  | Result | Frequency |  
  |--------|-----------|  
  | 1 | 2 |  
  | 2 | 3 |  
  | 3 | 0 |  
  | 4 | 1 |  
  | 5 | 2 |  
  | 6 | 7 |  
  ​

**OUTPUT:**

A white background with blue dots and numbers

AI-generated content may be incorrect.

A graph of a graph

AI-generated content may be incorrect.

**RESULT**: This function returns a list of integers representing the outcome of rolling num\_dice dice, each with sides sides. Let me know if you want to extend this for batch rolls, weighted probabilities, or graphical output.