# Assignment 1.1: The Gambler's Ruin

*A Study in Vectorization and Variance*

---

## 1. Objective

Before we simulate complex games like Blackjack, we must master the art of **Vectorized Simulation**. A "Random Walk" is the mathematical foundation of financial modeling, physics simulations, and risk analysis.

In this assignment, you will simulate **10,000 gamblers** playing a simple coin-flip game simultaneously. You will analyze their wealth over time and visualize the brutal reality of variance.

**THE IRON RULE: NO FOR LOOPS**

To pass this assignment, you are **strictly forbidden** from using Python `for` or `while` loops for the simulation logic.
You must use NumPy arrays and broadcasting. Loops are **only** allowed for:

- Plotting (e.g., iterating through a list of axes).
- Printing summary statistics at the very end.

If you iterate through 10,000 gamblers one by one, your code will be rejected.

## Part A: The Matrix (Data Generation)

We will simulate a simplified betting game.

- **The Setup:** Create a simulation with $N = 10,000$ gamblers.
- **The Stakes:** Each gambler starts with a bankroll of **$100**.
- **The Game:** They play $T = 1,000$ rounds of a fair coin flip.
  - **Heads (+1):** Win $1.
  - **Tails (-1):** Lose $1.

**Task:** Generate a NumPy array of outcomes (wins/losses) with shape `(10000, 1000)`.
*Hint: Look into `np.random.choice` or `np.random.randint`.*

## Part B: The Path (Cumulative Sum)

You have the individual wins/losses. Now, calculate the actual bankroll over time.

1. Use a cumulative sum function to transform the wins/losses into a trajectory.
2. Ensure every gambler starts at $100. (The first column of your trajectory matrix should reflect the starting wealth).

## Part C: The Ruin (The Challenge)

In the real world, casinos do not let you play with debt. If a gambler hits **$0**, they are bankrupt ("Ruined"). They stop playing immediately.

**Task:** Implement logic to handle bankruptcy.

- Identify which gamblers hit $\leq 0$ at any point in the simulation.
- For those gamblers, their bankroll must remain at 0 for all subsequent steps (flatten the curve).

**Constraint:** You must achieve this using Boolean Masking or NumPy functions. Do not iterate through the rows to check for zeros.

## Part D: Visualization

Use `matplotlib` to create a professional dashboard with **two subplots**.

### Plot 1: The Spaghetti Plot

Visualize the trajectories of the **first 100 gamblers**.

- Plot their bankroll vs. time (0 to 1000).
- Use a low alpha value (transparency) so we can see the density of the paths.
- Plot the **Mean Path** (average of all 10,000 gamblers) as a thick **Red Dashed Line**.
- Plot the **Max Winner** and **Max Loser** in distinct colors.

### Plot 2: The Final Distribution

Create a histogram of the **Final Wealth** (at step 1000) of all 10,000 gamblers.

- Use at least 50 bins.
- Add vertical lines indicating the **Mean** and **Median** final wealth.

## Part E: Analysis

In a Markdown cell at the end of your Notebook, answer the following:

> *"The average expected return of a fair coin flip is 0. However, looking at your histogram, you will likely see a Bell Curve (Normal Distribution). Why does the wealth spread out so much if the game is fair? Relate this to the **Central Limit Theorem** and **Variance**."*

## Deliverables

1. A Jupyter Notebook or Python file containing the code.
2. The code must run from top to bottom without errors.
3. The visualizations must be clearly labeled (Title, X-axis, Y-axis, Legend).