# Assignment 4.1: The Gambler's Strategy

*Monte Carlo Prediction, Control, and The Infinite Deck*

**Objective**

In Week 1, you built the casino. In Week 3, you derived the math. Now, you will build the player.
The goal of this assignment is to implement **Monte Carlo Control** to solve the game of Blackjack. Your agent will start with zero knowledge, play thousands of hands, and learn the optimal strategy purely from experience.

## Part I: The Setup (Environment)

## Option A: Your Custom Code (Recommended)

If you successfully completed Week 1 and have a working `Blackjack` class, use it!
**Requirement:** You must wrap your game in a class with a `step()` function that returns:

- `state`: A representation of the current cards.
- `reward`: $+1$ (Win), $-1$ (Loss), or $0$ (Draw).
- `done`: Boolean (True if round is over).

## Option B: OpenAI Gymnasium (Fallback)

If your Week 1 code is buggy, you may use the standard library. Please go through the documentation to get started.

**Python Implementation**

```
import gymnasium as gym
env = gym.make('Blackjack-v1', sab=True)
state, info = env.reset()

# Note: State is a tuple: (PlayerSum, DealerCard, UsableAce)
# e.g., (16, 10, False)
```

---

## Part II: The Algorithms (Implementation)

## Task 1: Monte Carlo Prediction

**The Policy:** "Stick if sum is 20 or 21. Hit otherwise."

### Your Task:

1. Run 10,000 episodes using this policy.
2. Keep track of the Returns ($G$) obtained from every state visited.
3. Implement **First-Visit Monte Carlo** to estimate $V(s)$.
4. **Deliverable:** Print the calculated value of having a "Player Sum of 21" vs "Player Sum of 5".

## Task 2: Monte Carlo Control

Now, we want to find the **Optimal Policy** $\pi_*$. We must learn **Action-Values** $Q(s, a)$.

### Your Task:

1. Initialize $Q(s, a)$ arbitrarily (e.g., all zeros).
2. Implement an $\epsilon$-greedy policy based on the current $Q$ values.
3. **The Loop:**
   - Generate an episode using the current $\epsilon$-greedy policy.
   - Calculate returns $G_t$ for each step.
   - Update $Q(s, a)$ using the incremental mean formula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[G_t - Q(s, a)]$$

4. Run for 500,000 episodes.

## Task 3: Visualization

1. **The Learning Curve:** Plot the "Rolling Average Reward".
2. **The Strategy Card:** Generate a heatmap showing the optimal action for every combination of (Player Sum, Dealer Card).

---

# Part III: The Reality Check

## Question A: The Infinite Deck Assumption

In real casinos, Blackjack is played with a "Shoe" of 6-8 decks without reshuffling. This allows for **Card Counting**.

### The Question:

1. **Non-Stationarity:** Explain why the environment is no longer Markovian if we define the state only as $(PlayerSum, DealerCard)$.
2. **State Space Design:** How would you modify the State definition $S$ to enable an RL agent to learn Card Counting?
3. **The Trade-off:** What happens to the convergence speed if you expand the state space?

## Question B: First-Visit vs. Every-Visit

**Scenario:** $A \to B \to A \to Terminate$.
Rewards: $(A \to B : +1)$, $(B \to A : -1)$, $(A \to Term : +10)$.

Calculate the return $G$ for state $A$ under:

1. First-Visit Monte Carlo.
2. Every-Visit Monte Carlo.