Final Projects for the OOP course
Due date: May 31st 2024

Introduction

The two options for the final project are linked to the card game known as War or Battle ("Bataille" in French). Before diving into the projects description, let us clarify the rules of the game.

Rules of War

1. The game is played with a deck of 32 cards (from 7s to Aces in each suit).
2. The objective is to win all the cards.
3. Each player draws the top card of their deck; the player with the higher card wins the round and takes both cards, placing them at the bottom of their deck.
4. In the event of a tie (war), each player draws the top card of their deck and places it face down and then draws the top card of their deck and places it face up. The player with the higher face-up card wins all the cards on the table. The above is repeated as long as there is a tie.
5. If a player runs out of cards during a war, that player immediately loses the game.
6. If both players run out of cards simultaneously during a war, the game is declared a draw.
7. Cards are shuffled before being placed back under the player's deck.
8. To avoid infinite scenarios, if no player has won after 5000 rounds, the game is declared a draw.

Project Option 1: Flask Web Application

Objective

Develop a web application using Flask that simulates different scenarios of the game War and calculates probabilities of winning based on initial conditions (e.g., holding more than 3 Aces, less than 3 Kings and no Jack).

Requirements

- Implement the game logic in a Python module, using classes and methods to represent game elements like the cards, the deck, the players and the game.
- Create a Flask application where users can input specific starting conditions and receive the probability of winning.

Project Option 2: Script with File Input

Objective

Create a `battle.py` script file that simulates the game and provides statistical insights based on user queries (e.g., probability to win if initially holding more than 3 Aces, less than 3 Kings and no Jack).

Requirements

- Implement the game logic in a Python module, using classes and methods to represent game elements like the cards, the deck, the players and the game.
- The script must accept initial scenarios through a file input (with your own encoding logic), e.g. `python battle.py -play scenario.txt`
- Implement parsing logic to interpret the user question and simulate the game based on the conditions specified.
- Provide a help option (`python battle.py -help`) that explains the file format and available commands.

Submission Guidelines

Submit your project by email by the deadline:

- Source code with appropriate comments.
- A report documenting your design decisions and a brief user guide.
- Results of sample simulations demonstrating the functionality of your tool.