Building a Blackjack Game in Python

Chun-Yen Lin

Hsin-Yu Liao

Master Science of Business Analytics, University of the Pacific

MSBA 232 Programming for Data Science

Dr. Xun Xu

Building a Blackjack Game in Python

Abstract

This project presents the development of a simplified Blackjack game using Python and the Tkinter graphical user interface library. The goal was to apply core programming concepts learned throughout the semester, including conditional statements, loops, functions, and basic GUI design, within an interactive and user-friendly application. The game replicates essential Blackjack mechanics such as card dealing, score calculation, and result determination for win, loss, Blackjack, and tie scenarios. A fixed deck was implemented using lists, with appropriate value handling for Aces and face cards. Visual feedback and gameplay logic were tested through multiple rounds, with results confirming the program's accuracy and responsiveness. While primarily educational in purpose, the project demonstrates how even basic programming skills can be extended into practical applications with potential relevance for user training or entertainment-based business tools.

Keywords: Python, Blackjack, Loop structures, if-else, Tkinter

1. Introduction / Background

Blackjack is one of the most popular table games in casinos around the world. It features simple rules and a fast-paced rhythm, making it highly favored by players. It is also one of the few casino games in which players can influence the outcome through strategy. As such, Blackjack has become a popular entry point for those interested in probability, decision-making, and card game tactics.

According to statistical studies, Blackjack odds refer to the probability of winning, losing, or drawing based on the player's hand and the dealer's upcard. On average, the player's chance of winning is about 42%, while the dealer wins approximately 49% of the time (Blackjack odds –

know the odds of winning, 2024). Despite the house edge, players can still improve their chances through practice and sound strategy.

However, for many beginners, Blackjack is deceptively simple. Beneath the surface lies a game filled with nuance and decision-making complexity. Many new players find themselves confused or frustrated at the table, often losing quickly due to a lack of understanding. This learning curve is one of the main challenges this project aims to address.

2. Project Objective

The objective of this project is to develop a Blackjack practice game that allows beginners to learn and play in a risk-free environment. Through an interactive program, users can repeatedly practice essential gameplay actions such as "Hit" and "Stand," and observe how the dealer responds according to preset rules. This helps players build familiarity and confidence with Blackjack mechanics.

From a programming perspective, this project serves as a comprehensive exercise in applying the core skills taught throughout the semester. It demonstrates the integration of key concepts such as function creation, conditional logic, loops, object-oriented programming, and GUI design using Tkinter. The goal is not only to create an engaging and educational tool, but also to solidify foundational Python programming skills through a hands-on, applied project.

3. Data / Problem Analytics

3.1 Data

In this project, the dataset is manually created to simulate a standard deck of 52 playing cards used in Blackjack. The card list is generated using the following values: ["A", 2, 3, 4, 5, 6, 7, 8, 9, 10, "J", "Q", "K"] * 4, representing all cards from Ace through King, with four copies of each to reflect the four suits.

Each card is associated with a numerical value based on Blackjack rules: Aces can count as either 11 or 1, depending on the hand's total score, while face cards (J, Q, K) are all valued at 10. These rules are implemented programmatically in a function that calculates the score of a hand. This structured data allows the game logic to simulate realistic Blackjack behavior and outcomes.

3.2 Methods

This project applied a variety of fundamental Python programming techniques and logic structures to implement the core mechanics and interactive functions of a Blackjack game. The program is structured using function-based modularity, where key components such as card dealing, score calculation, and Ace value adjustment are each defined in separate functions (def) to enhance clarity and maintainability.

Conditional statements (if-elif-else) are used to handle various game outcomes, such as detecting when a player exceeds 21 points (busts), determining win or loss conditions, and handling ties (push). A while loop is used to automate the dealer's actions, allowing the dealer to continue drawing cards until the hand value reaches at least 17, following standard Blackjack rules. The same loop structure is also used to dynamically adjust the value of Aces from 11 to 1 when necessary to prevent busting.

The program stores the player's and dealer's cards in Python lists, using list comprehensions and string operations to format the visual display of each hand and its total score. The overall program adopts an object-oriented design structure, with all core functions encapsulated within the BlackjackGame class and the interface initialized via the __init__ method.

For the user interface, the Tkinter module is used to build a basic graphical interface (GUI). The design includes labels for displaying hands and results, buttons for controlling game flow

(such as Hit, Stand, Play Again, and Quit), and frames for layout organization. This allows users to interact with the game in a visually intuitive way.

The project demonstrates a practical application of core Python concepts learned throughout the semester, including variable usage, conditional logic, loops, and function definitions…etc. These elements are combined to form a functioning Blackjack game that emphasizes both interactivity and logical flow.

3.3 Results of Problem Analytics

To evaluate the program's behavior and ensure that the implemented logic aligns with Blackjack rules, several screenshots were taken to illustrate the full range of user interaction and game outcomes.

Figure 1

Game start screen with inactive buttons

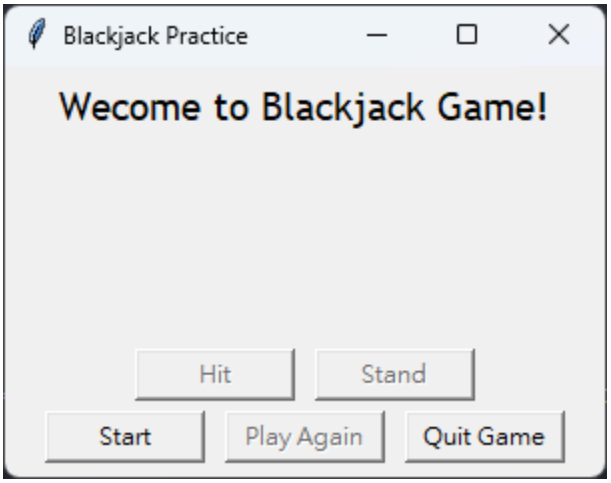
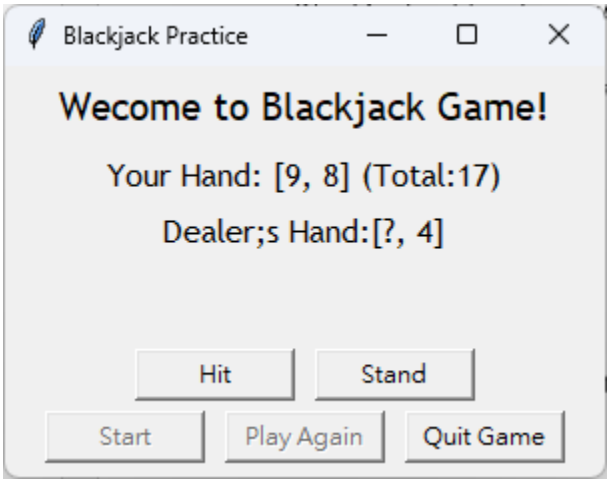


Figure 2

Initial hand after game starts



The game launches with a clean and minimalistic interface, as shown in Figure 1. At this stage, only the "Start" button is enabled, guiding the user to begin a new game session. This initial screen serves as the user's first point of contact and sets the tone for a smooth gameplay experience.

Once the user clicks "Start," the game begins by dealing two cards to both the player and the dealer. As seen in Figure 2, the player is immediately shown their total score, while the dealer's

hand remains partially hidden, consistent with conventional casino Blackjack procedures. This setup gives the player a clear view of their position before making strategic decisions such as choosing to "Hit" or "Stand."

Figure 3

Blackjack result

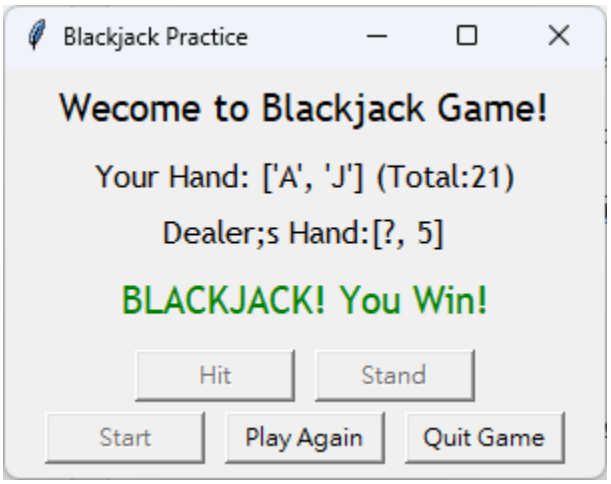
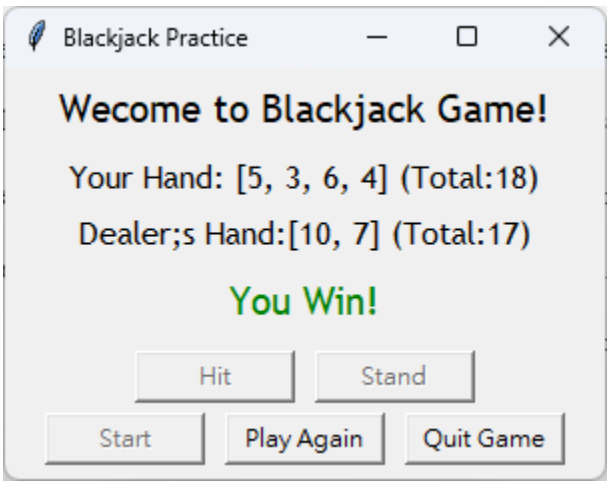


Figure 4

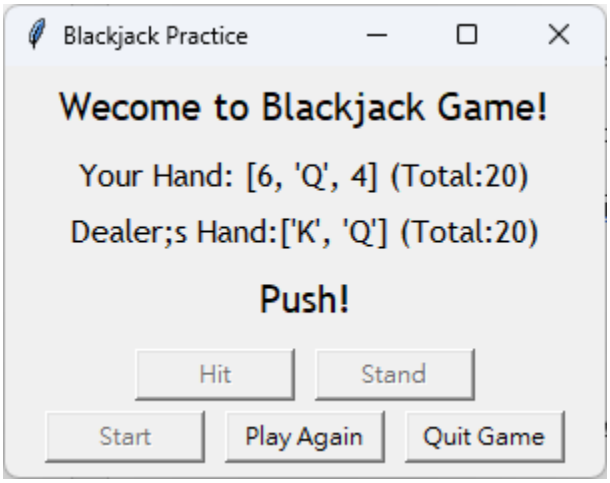
Player wins by total points



The program handles all possible outcomes correctly, each demonstrated through practical gameplay examples. In Figure 3, the player draws an Ace and a face card, forming a perfect Blackjack with a total of 21. The interface responds accordingly by displaying a green-highlighted success message: “BLACKJACK! You Win!”. In Figure 4, the player achieves a winning hand by drawing multiple low-value cards that total 18, which successfully beat the dealer’s hand of 17. This demonstrates that the game logic correctly handles cases where the player wins without reaching 21. The result is displayed clearly with a green "You Win!" message, confirming the game’s ability to interpret and compare hand totals accurately.

Figure 5

Tie (Push) Outcome



Meanwhile, Figure 5 captures a tie scenario in which both the player and the dealer have identical hand values. The game identifies the outcome as a "Push," indicating no winner. This reflects standard Blackjack rules and shows that the program is capable of recognizing and handling neutral results appropriately.

Figure 6

Dealer wins by higher total

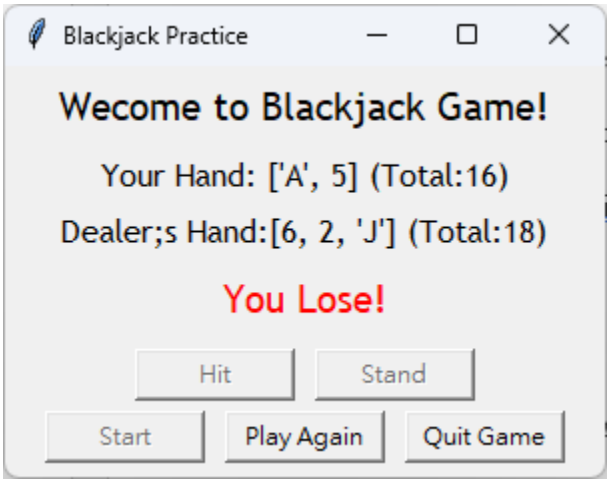
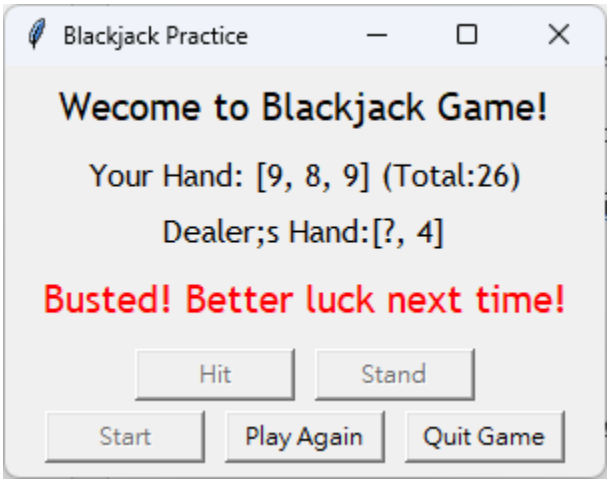


Figure 7

Player busts and loses



Additionally, Figure 6 displays a loss scenario where the dealer ends with a higher total than the player. The message "You Lose!" is shown in red, signaling the unfavorable result. Lastly,

Figure 7 illustrates a bust situation in which the player's total exceeds 21. The game logic correctly identifies the bust and ends the round accordingly.

4. Implications and Conclusions

The development of this simplified Blackjack game demonstrates the practical application of foundational Python programming concepts in creating an interactive and rule-based decision-making system. From a technical standpoint, the project effectively integrates data structures, logic flow, modular functions, and GUI elements to simulate a real-world game environment. The successful implementation of Blackjack mechanics highlights the potential for programming tools to be leveraged beyond academic exercises, especially in areas such as simulation training, user education, or entertainment software.

From a managerial perspective, basic programming knowledge can be transformed into functional tools that enhance user engagement and experiential learning. For instance, similar logic and interface structures could be applied to business simulations or customer onboarding tools in various industries, including finance, hospitality, or retail. Furthermore, the project highlights how even fundamental programming skills can contribute meaningfully to the development of functional tools that enhance user learning and process visualization. In business contexts, such tools could be adapted for purposes such as employee training simulations or interactive decision-making modules, supporting organizations in delivering more effective and engaging experiences.

5. Idea Sharing

Throughout the development of this project, one of the most insightful aspects was the implementation of the graphical user interface using Tkinter. While Tkinter is a relatively simple and lightweight library, it provided a valuable opportunity to explore how graphical components such as buttons, labels, and frames can be integrated with back-end logic to create an interactive

user experience. The process revealed that even basic GUI design choices, such as the timing and method of updating visual elements, can significantly influence usability and player engagement. In addition, working with Tkinter highlighted the importance of aligning program logic with user-facing responses. Achieving consistency between game events and their visual representation required careful coordination between event-handling functions and game rule enforcement. This experience emphasized that effective software development must consider both logical correctness and clarity in user presentation. Ultimately, this project expanded my understanding of Python's versatility, demonstrating its ability to support not only algorithmic problem-solving but also the creation of intuitive and engaging user interfaces.

6. References

Blackjack odds – know the odds of winning. WinStar Blog. (2024, November 7).

https://www.winstar.com/blog/blackjack-odds/