Tecnológico de Monterrey

Report:
Blackjack game coded in Python

Ndrine Avdiu - A01762207

15/10/2023

Professor in charge

Silvana De Gyvés Avila

Computational Thinking For Engineering

I. Detailed Description of the Problem

The software implementation of the popular card game Blackjack is called the Python Blackjack project. People all over the world enjoy playing the casino game blackjack, usually referred to as 21. The goal of the game is to beat the dealer by getting as near to 21 without going over it as you can with your hand value. A player loses the hand and the game if their hand value is higher than 21.

The desire to create this project was sparked by a love of card games, especially Blackjack, but it was also an act of retaliation for a previous failure to complete the project's coding. The goal of this project is to develop a fun and engaging way for people to play and understand many blackjack variations.

II. Description of the Solution

The project provides three distinct versions of the Blackjack game:

1. Classic Blackjack: This version follows the traditional rules of Blackjack, making it ideal for those who are familiar with the standard game. It features deck creation, player interaction, hand value calculations, and an automated dealer.

2. Blackjack with a Twist: In this variation, the value of the Ace is changed from 11 to 1. This change introduces a unique challenge and strategy for players.

3. Spanish 21: Spanish 21 utilizes a different deck, removing all 10s. The game includes a specialized deck creation, player choices, and hand value calculations following Spanish 21's specific scoring rules.

III. Key Features

The project includes the following features:

- Three different versions of Blackjack.
- User-friendly command-line interface.
- Interactive gameplay with choices for players.
- Automatic dealer functionality.
- Hand value calculation and checking for winning conditions.

IV. How to Use It

To use the project, follow these steps:

1. Launch the Python program.
2. Choose one of the three Blackjack versions or choose to see the rules.
3. Interact with the game by selecting 'hit' or 'stand' during your turn depending on the hand you got.
4. Develop a strategy in order to win the game and not be busted.

The project handles deck creation, card distribution, and winner determination.

V. How Does the User Execute It?

The user can execute the project by running the Python script using their preferred Python interpreter. Once executed, the user will have to choose between 4 options, what he wants to do, either he plays one of the three versions, or he decides to read the rules or just quit the code.

If he chose to play one of the different versions of Blackjack, he will be given a hand of card and his total score will appear, then he decides either he wants to add to the score and hit once again or just stand at the score he has and wait for le dealer’s hand. Once the dealer’s hand is given, we can tell who is the winner, and the game is over.

VI. Technical Dependencies

The Python programming language is used in this project. It uses the random library to generate unique hands for players each time, and it also uses the rules.txt file to enable the user to view the rules as necessary. We use for and while loops, if conditions, files, lists, and I even created a dictionary to make my deck of cards easier to manipulate. All project requirements are met. To operate the project, users require a Python interpreter.

VII. Test Plan

A test plan has been created to guarantee the project's functionality and dependability. This strategy includes a number of test cases that examine several facets of the code, including the construction of the deck, distribution of the cards, determination of the hand values, player decisions, and winner determination. Each project function and feature is rigorously evaluated to ensure its accuracy and compliance with each Blackjack version's standards. The initiative seeks to provide consumers with a fluid and satisfying gaming experience.

These are the different test cases for the functions that can have one:

display_hand(hand):

INPUTS	OUTPUTS
All the deck of cards	3 of Diamonds Q of Hearts

calculate_hand_value(hand):

INPUTS	OUTPUTS
A of Diamonds Q of Hearts	20

play_blackjack():

INPUT	PROCESS	OUTPUTS
3 of Diamonds Q of Hearts 13 Hit or stand ?	4 of Clubs 17 Hit or stand?	Dealers hand Win/loose

Or:

INPUT	PROCESS	OUTPUTS
J of Diamonds Q of Hearts 20 Hit or stand ?	Hit 4 of Clubs	Busted Loose

calculate_hand_value_blackjackTwist(hand):

INPUTS	OUTPUTS
A of Diamonds Q of Hearts	11

play_blackjack_with_twist():

INPUT	PROCESS	OUTPUTS
A of Diamonds Q of Hearts 11 Hit or stand ?	J of Clubs 21	Blackjack You win !

Or:

INPUT	PROCESS	OUTPUTS
A of Diamonds Q of Hearts 11 Hit or stand ?	Hit 9 of Clubs 20 Hit or stand ?	Dealers Hand Win/Loose

calculate_spanish_hand_value(hand):

INPUTS	OUTPUTS
9 of Spades 7 of Clubs	16

play_spanish_21():

INPUT	PROCESS	OUTPUTS
9 of Spades 7 of Clubs 16 Hit or stand ?	J of Clubs 26	Busted! You loose

Or:

INPUT	PROCESS	OUTPUTS
9 of Spades 7 of Clubs 16 Hit or stand ?	Hit 4 of Clubs 20 Hit or stand ?	Dealers Hand Win/Loose

main():

INPUTS	OUTPUTS
1	You selected Standard Blackjack.
2	You selected Blackjack with a Twist.
3	You selected Spanish 21.
4	rules.txt
5	Goodbye!