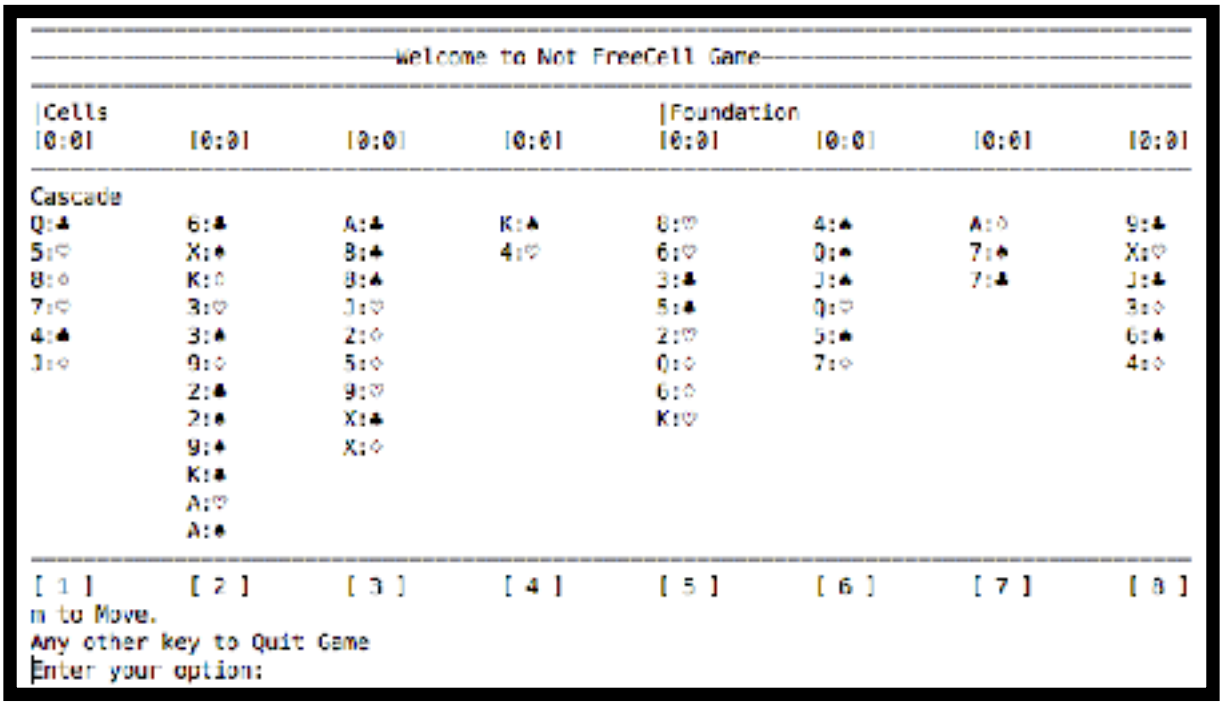
Not Freecell

A Python Project



]

Class Card

Card class is the first class which is created to create the card to be used for the deck. This

A Card is constructed by passing the face of the card (1 to 13) and the suit which it should belong to (Heart, Diamond, Club, Spade). In addition to the face and suit, symbol for the card is also generated based on the suit. To differentiate red and black suits, the symbols are used.

Club Black : ♣

Spade Black : ♠

Red Heart : ♡

Red Diamond : ♢

If other suits are required, the symbols in the Card class can be updated as required. By default the symbol assigned would be 'NA' is the suit is unknown. The character for the cards A, J, Q, K are generated if the face value is 1, 11, 12, 13 respectively. If the face value is 10, the character is generated as X so simply the screen output with all single characters. For any other face value, the character would be same as it's face value,

repr method :
This method returns the string object by concatenating the character of the card, colon ":" and the symbol of the card.

get_face method :
This method accepts the card object and returns the face value of the card object.

get_suit method :
This method accepts the card object and returns the suit of the card object.

get_symbol method :
This method accepts the card object and returns the symbol of the card object.

Deck Class

Deck class is used to construct the deck using the cards and setting up the cascade, foundation and cells.

init Method :

The Deck class is initiated by passing the start value, end value and number of suits required for the deck. From the start value and the end value, the number of card face values for each suit is calculated and made into a list. The number of suits required randomly from the available number of suits. By using the number of cards and the number of suits, a deck is created by calling the Card class. This would create a cards_deck list which would not have an ordered list of the cards needed for the game. The list needs to shuffled to randomise the distribution of the cards in the cascade. To shuffle the method shuffle() is used. After shuffling the list the deck should be split randomly and assigned to the cascade of size 8.

shuffle Method :

The cards_deck would be a sorted list having cards in the ascending order for each suit. To have the random distribution of the cards to the deck, the cards needs to be shuffled. random.shuffle is used by passing the cards_deck to shuffle the list to be used for the game.

split_list Method :

The shuffled cards_deck should be distributed to a list of size 8 having a random number of cards in each sublist. A loop runs until the cards_deck becomes empty. Inside the loop another loop is run 8 times, one for each list. A random number from 0 to 5 or the length of the cards_deck whichever is minimum is picked for each loop. The final_deck list for each position is picked and appended with the cards_number. This happens as many times as the size of the random number generated earlier. Once the loop is complete the list final_deck would have all the cards distributed randomly.

drawCard Method :

This method takes the parameters place and position and returns the face value and suit of the card at that place and position. If the position at the place is

empty, it returns 0, 0. Please can accept the values ‘foundation’, ‘cascade’ and ‘cell. Position can accept any integer value.

addCard Method :

This method takes 4 parameters as inputs - from place, from position, to place, to position. Based on the input the card is moved based on the place and position.

1. From cascade

    a. If the to place is also cascade then the card is removed from the from position of the cascade and appended to the to position of the cascade.

    b. If the to place is cell, then the card is removed from the cascade at the position and added to the cell at the position.

    c. If the to place is foundation, then the card is removed from the cascade at the position. The foundation is checked if it is empty. If empty the card removed from cascade is added at position 1, or else the card is appended to the to position of the foundation.

2. From Foundation

    a. If the to place is cascade then the card is removed from the foundation at the position and added to the cascade at the position. If the length of the foundation is one then an empty card object is added.

    b. If the to place is cell, then the card is removed from the foundation at the position and added the cell at the position. If the length of the foundation is one then an empty card object is added.

3. From Cell

    a. If the to place is foundation, then the card is removed from the cell at the position. The foundation is checked if it is empty. If empty the card removed from cell is added at position 1, or else the card is appended to the to position of the foundation. An empty class object is created at the from position of the cell.

    b. If the to place is another cell, then the card is moved from the from position to the to position. An empty class object is created at the from position of the cell.

c. If the to place is cascade then the card is removed from the cell at the position and added to the cascade at the position. An empty class object is created at the from position of the cell.

str method

This method is to build the output screen for the user using the cascade, foundation and cell, and creating a string object which would be returned to the class object.

NotFreecell Class

The NotFreecell class would have all the rules of the Freecell game implemented by calling the Deck class.

init Method

In the init method the Deck object is created using the Deck class. gameDeck would be the object.

str Method

This method would return the string object returned from the Deck class.

draw Method

This method is to call the drawCard method from the Deck class by passing the place and position of the card required. The return object would consist of the card and suit of the card at the place and position.

moveCard Method

This method has all the rules of the Freecell which is to be implemented while moving the card. The method takes parameters from place, to place, from position, to position

Initially the user input values for the from Place and to Place are connected into values like ‘cascade’, ‘foundation’ and ‘cell’.

The existing card details in the from position and to position are retrieved by calling the draw method by passing the parameters needed.

1. From Cascade

a. If the from position doesn't have any card then the cascade position entered by user is empty. Message is shows saying cascade is empty.

b. If the cascade has a card and to place is also a cascade then the to position of the cascade is checked. If the cascade at to position is empty then no need to check any rules. The card is moved from the from position to the to position by calling the addCard method from the Deck class.
If the to position has a card already then a check is done if the card is of opposite colour and also one face value more than the from card. If so, then call the addCard method. Or else display message saying card cannot be inserted.

c. If the cascade at from position has a card and to place is a cell, then a check is done if the cell already has a card at the mentioned position. If so then display message saying cell is not empty. Or else call the addCard method to move the card from the cascade to the cell.

d. If the cascade at from position has a card and to place is foundation, then a check is done if the foundation is empty. If it is empty then the card face is checked if it is having face value of 1 (Ace). Since first card in a foundation can only be an Ace this check is done. If success then call the addCard method. If fails then display message. If the foundation is not empty then a check is done if the card is of same suit of the existing foundation and exactly one value more then the card at the existing foundation. If both the condition are satisfied then the card is inserted. Or else message is displayed to the user.

2. From Cell

a. If the from position doesn't have any card then the cell is empty. Display message to the user saying the cell is empty at the position.

b. If the cell is not empty and the to place is cascade, then the to position of the cascade is checked. If the cascade at to position is empty then no need to check any rules. The card is moved from the from position to the to position by calling the addCard method from the Deck class.
If the to position has a card already then a check is done if the card is of opposite colour and also one face value more than the from card. If so, then call the addCard method. Or else display message saying card cannot be inserted.

c. If the cell is not empty and to place is also a cell, then a check is done if the cell already has a card at the mentioned position. If so then display message

saying cell is not empty. Or else call the addCard method to move the card from the cascade to the cell.

d. If the cell is not empty and to place is a foundation, then a check is done if the foundation is empty. If it is empty then the card face is checked if it is having face value of 1 (Ace). Since first card in a foundation can only be an Ace this check is done. If success then call the addCard method. If fails then display message. If the foundation is not empty then a check is done if the card is of same suit of the existing foundation and exactly one value more then the card at the existing foundation. If both the condition are satisfied then the card is inserted. Or else message is displayed to the user.

3. From Foundation

a. If the from position doesn't have card then it means that the foundation is empty. Display a message to the user saying that the foundation is empty at the position.

b. If the foundation is not empty and to place is a cascade, then the to position of the cascade is checked. If the cascade at to position is empty then no need to check any rules. The card is moved from the from position to the to position by calling the addCard method from the Deck class.
If the to position has a card already then a check is done if the card is of opposite colour and also one face value more than the from card. If so, then call the addCard method. Or else display message saying card cannot be inserted.

c. If the foundation is not empty and to place is a cell, then a check is done if the cell already has a card at the mentioned position. If so then display message saying cell is not empty. Or else call the addCard method to move the card from the cascade to the cell.

Main

The main method has the logic to run the game by taking user inputs and calling necessary functions from the NotFreecell Class.

check_ip Method

This method takes the parameters start, end and value is checks if the value is between the start and the end. If true then no action is taken. If it fails then a message is displayed saying Invalid input and program is quit.

User input is requested to get the start value, end value and number of suits needed for the deck. User input is validated by calling the check_ip method.
A check is done to see if the end value is less than the start value. If so then display message and quit the execution of the program. If not then continue execution.
Using the start, end and number of suits values received from the user the NotFreecell class object is initiated.

A loop is run until user enters any other key apart from 'm'. Inside the loop the game is checked if it is completed. User input is requested for the action to be done.
From place, to place, from position, to position inputs are taken from the user. moveCard method of the NotFreecell class is called by passing these parameters. The loop will until the game ends in a victory or the user wants to quit.