blackjack.py

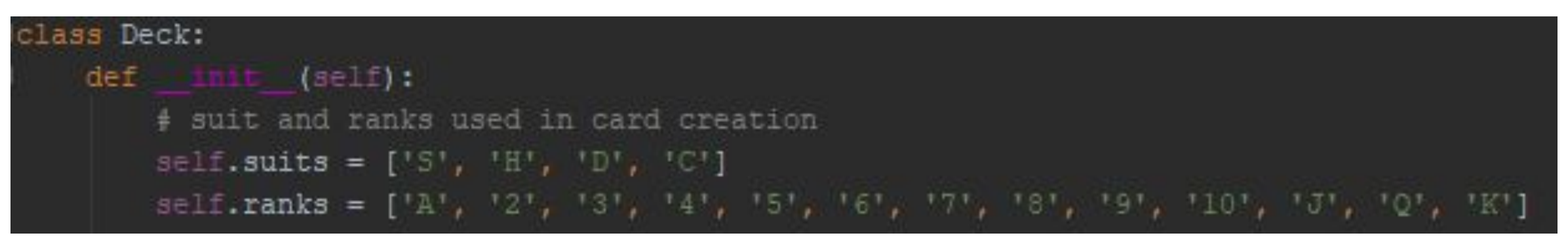
River Hill

MAIN CLASSES RUNNING THE GAME

The main classes that make this game are the Card, Hand, and Deck classes.

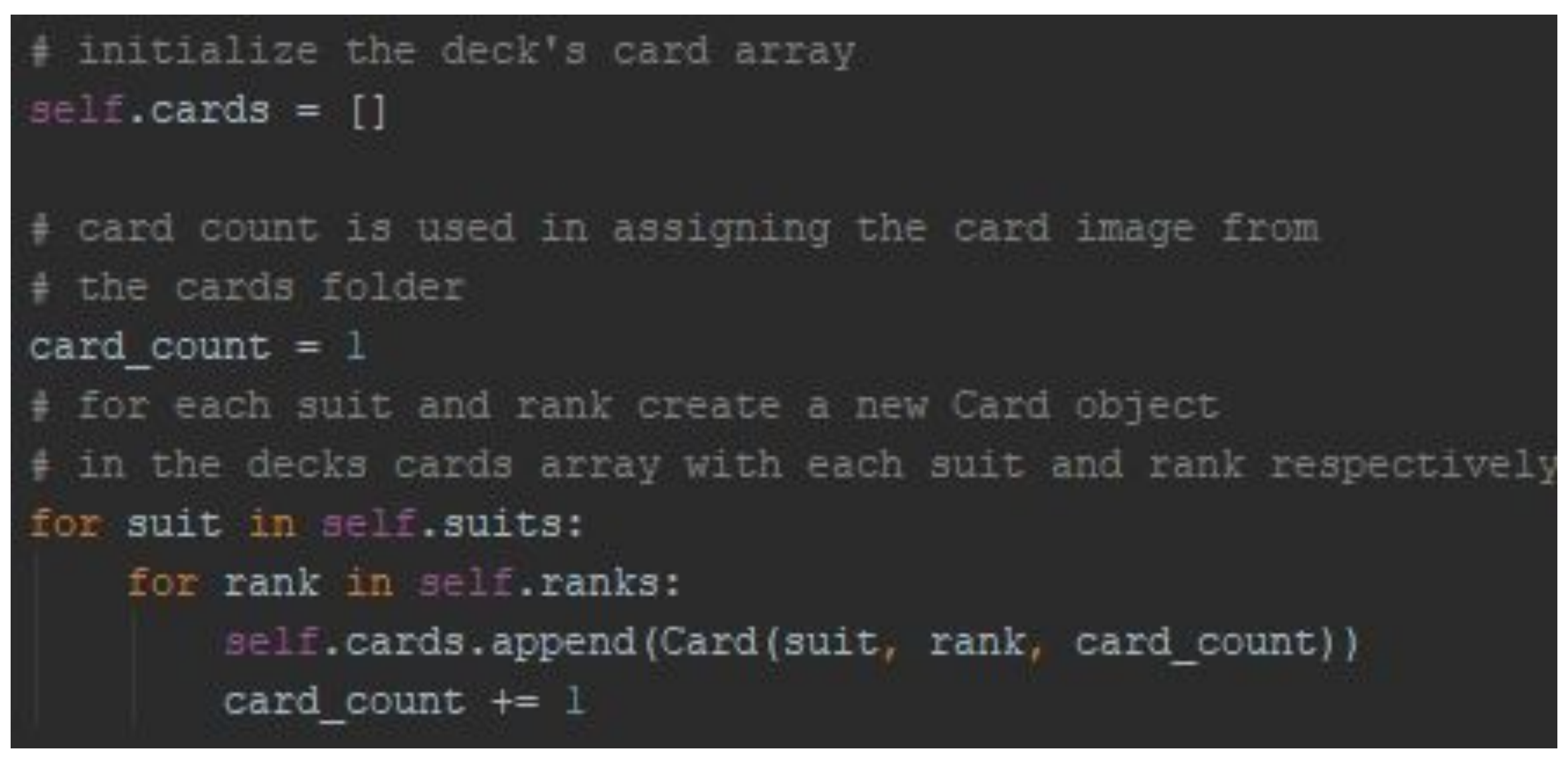
The Blackjack class keeps track of all the functions used in making the game possible. It begins by creating a new Deck class which consists of 52 Card class objects which keep track of their own assigned suit and rank.

This was accomplished with two arrays in the Deck class which held each suit and rank.



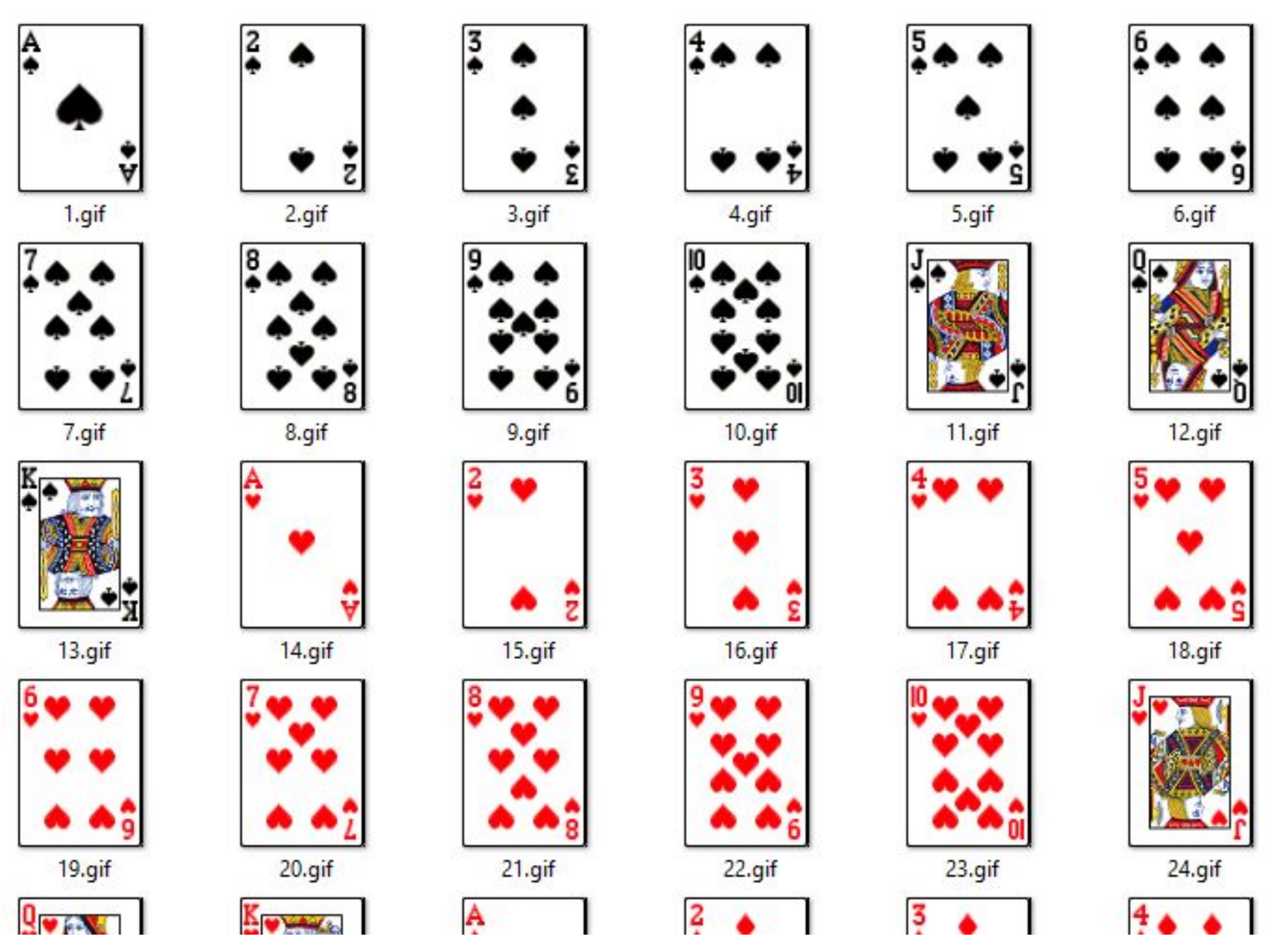
```
class Deck:
    def __init__(self):
        # suit and ranks used in card creation
        self.suits = ['S', 'H', 'D', 'C']
        self.ranks = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
```

An empty array is created to hold the new Cards and two nested loops go through each suit and rank and assigns it to the newly created Card. The card_count array is used in keeping track of what number a given card is within the deck and is used in finding the appropriate card image to display on screen.

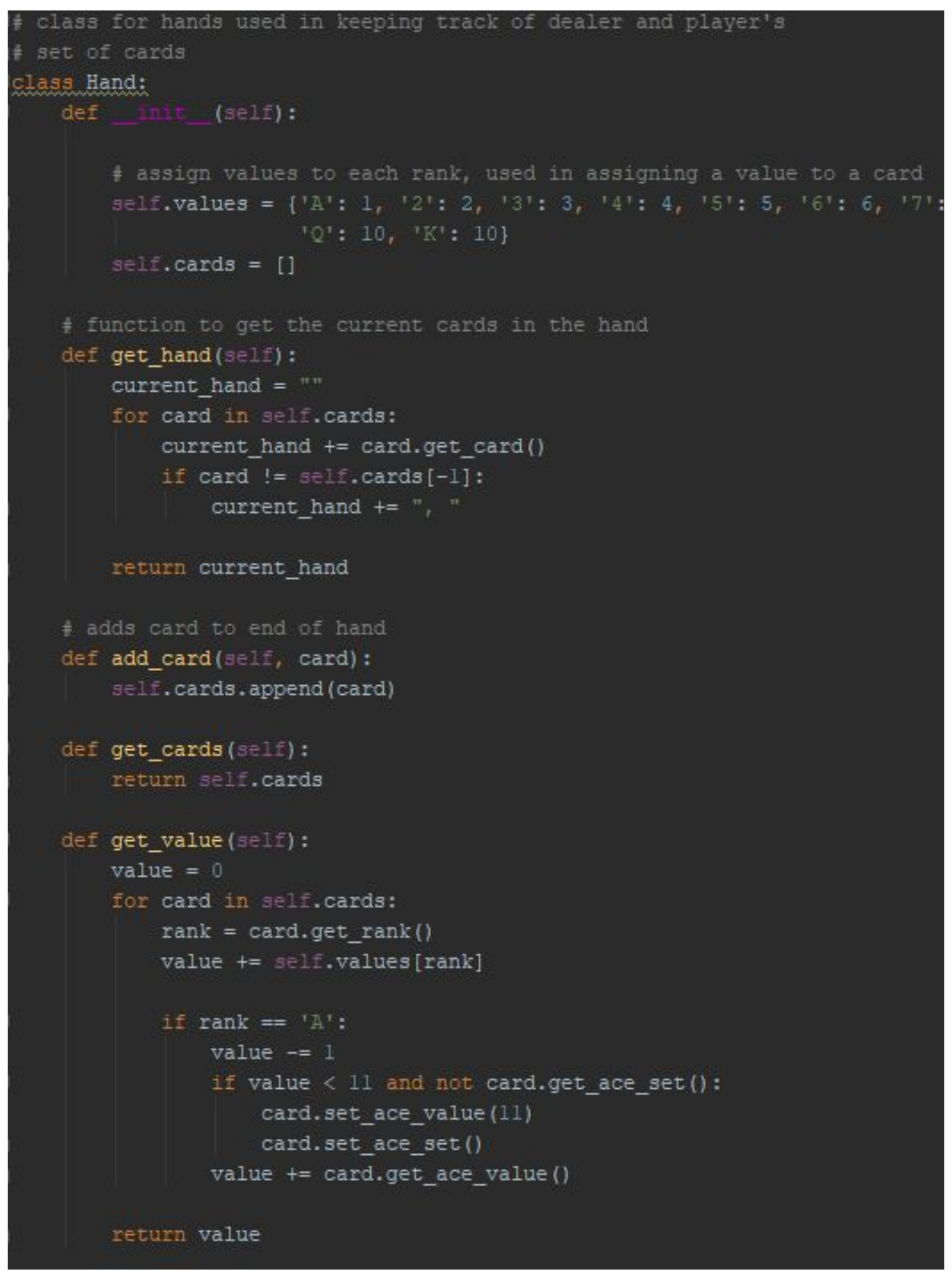


```
# initialize the deck's card array
self.cards = []

# card count is used in assigning the card image from
# the cards folder
card_count = 1
# for each suit and rank create a new Card object
# in the decks cards array with each suit and rank respectively
for suit in self.suits:
    for rank in self.ranks:
        self.cards.append(Card(suit, rank, card_count))
        card_count += 1
```

Using the card images provided within the python book examples, we found that they are in the order of Spades, Hearts, Diamonds and Clubs. Knowing this, we arranged my self.suit rank to reflect that which allows the card_count variable represent the card's image number



The Blackjack class then deals two cards from the Deck class to two new Hand classes for the dealer and player.
To find the value of a hand which is used in presenting the dealer and players hand value at the end of a turn we use the get_value function. This will go through each card, find it's rank, and compare it with the values array which keeps track of each ranks value,

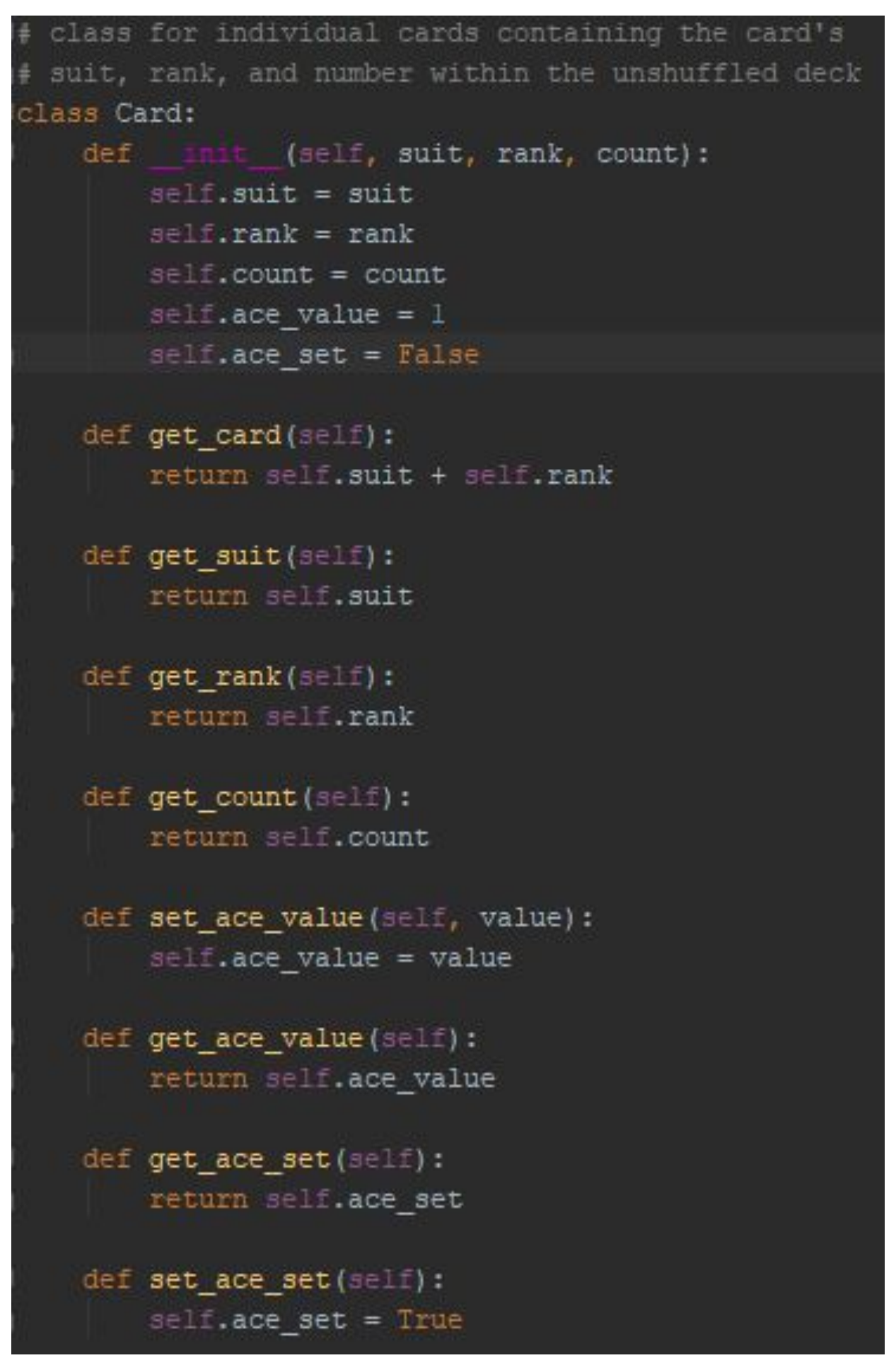


```
# class for hands used in keeping track of dealer and player's
# set of cards
class Hand:
    def __init__(self):

        # assign values to each rank, used in assigning a value to a card
        self.values = {'A': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7':
                       'Q': 10, 'K': 10}
        self.cards = []

    # function to get the current cards in the hand
    def get_hand(self):
        current_hand = ""
        for card in self.cards:
            current_hand += card.get_card()
            if card != self.cards[-1]:
                current_hand += ", "

        return current_hand

    # adds card to end of hand
    def add_card(self, card):
        self.cards.append(card)

    def get_cards(self):
        return self.cards

    def get_value(self):
        value = 0
        for card in self.cards:
            rank = card.get_rank()
            value += self.values[rank]

            if rank == 'A':
                value -= 1
                if value < 11 and not card.get_ace_set():
                    card.set_ace_value(11)
                    card.set_ace_set()
                value += card.get_ace_value()

        return value
```

PROBLEMS WE RAN INTO

A problem we faced right away was with the get_value function and aces. Currently we get the value of the player at the end of every turn and display it. By default if the player or dealer's total score is under 10 then we count aces as 11 otherwise we add 1. But since the get_values function is called at the end of every turn, it would change previous aces which were counted as 11 to 1 if the player's score got above 10. To fix this we had to add separate function in the card class which kept track if the Card had already had it's ace counted and the value it held.

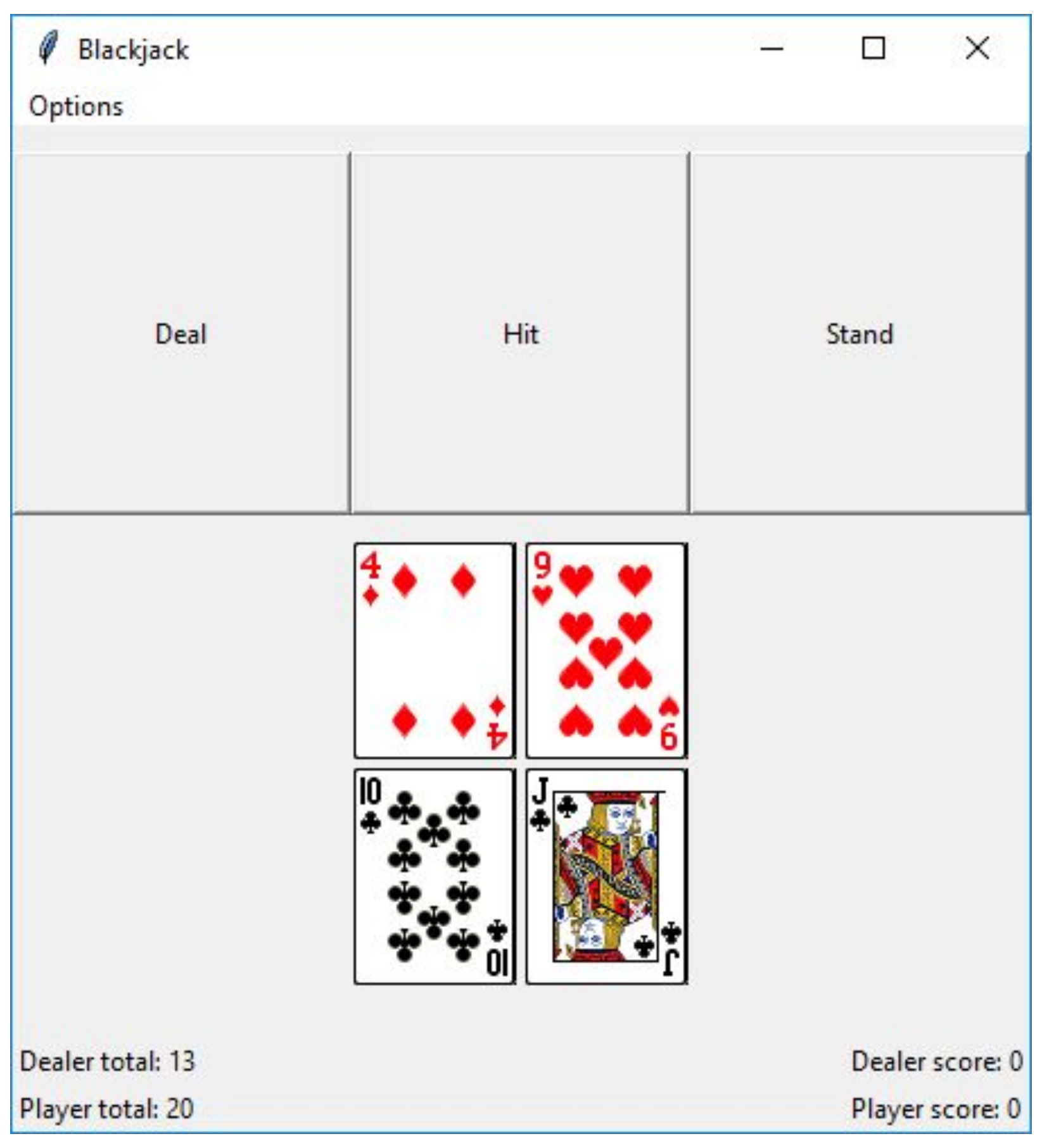
To compare this image with the last image in the left panel, the get_value function checks to see if the card it's analyzing is an ace. If it is, it will check if the card has been set before and if the hand value is below 11, if not it will run the cards functions which will set it as checked and set the cards ace value to 11 which is added to the hands total.



```
# class for individual cards containing the card's
# suit, rank, and number within the unshuffled deck
class Card:
    def __init__(self, suit, rank, count):
        self.suit = suit
        self.rank = rank
        self.count = count
        self.ace_value = 1
        self.ace_set = False

    def get_card(self):
        return self.suit + self.rank

    def get_suit(self):
        return self.suit

    def get_rank(self):
        return self.rank

    def get_count(self):
        return self.count

    def set_ace_value(self, value):
        self.ace_value = value

    def get_ace_value(self):
        return self.ace_value

    def get_ace_set(self):
        return self.ace_set

    def set_ace_set(self):
        self.ace_set = True
```

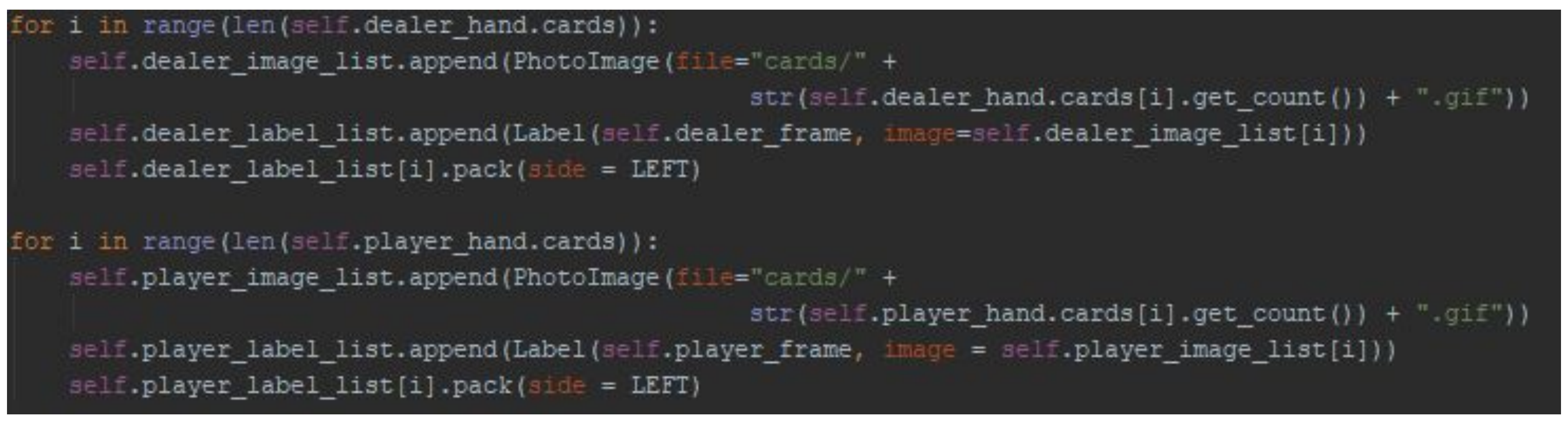
Another problem faced was figuring out how to display the dealer and player's cards on the screen. Using the example from the python book which showed how to display cards and shuffle them we modified it to show a single card using the Cards count variable.

We used a loop to go through the cards in a hand and create an image list which uses the Cards count variable to find the numbered image representing the card. Then we add the image to a label list which consists of the Tkinter label images for the cards which are then drawn to the screen.



```
Dealer total: 13                                Dealer score: 0
Player total: 20                                Player score: 0
```

The image below shows how images were displayed for the dealer's hand



```
for i in range(len(self.dealer_hand.cards)):
    self.dealer_image_list.append(PhotoImage(file="cards/" +
                                             str(self.dealer_hand.cards[i].get_count()) + ".gif"))
    self.dealer_label_list.append(Label(self.dealer_frame, image=self.dealer_image_list[i]))
    self.dealer_label_list[i].pack(side = LEFT)

for i in range(len(self.player_hand.cards)):
    self.player_image_list.append(PhotoImage(file="cards/" +
                                             str(self.player_hand.cards[i].get_count()) + ".gif"))
    self.player_label_list.append(Label(self.player_frame, image = self.player_image_list[i]))
    self.player_label_list[i].pack(side = LEFT)
```

Icons made by Freepik from www.flaticon.com