In [137…
```python
# Author Name: Wilson Wong
# Student ID: 29704154
# Creation Date: 11 Sep 2023
# Date Last Updated: 15 Sep 2023
# Decription:
    # This program is a card game with one player against a robot.
    # Cards are drawn from a customised deck.
    # Cards held are then compared based on specific rules to
    # determine a winner of the round.
```

In [138…
```python
# Video Link: https://youtu.be/TdY81e2GwWs
```

In [139…
```python
import random
```

Task 1. Game menu function

In [140…
```python
def game_menu(bool_flag, suit_types):
    """
    Display the game menu.

    Parameters:
    - bool_flag (boolean): A flag indicating whether the game has started
                        (a suit has been chosen).
    - suit_types (list): A list of the suit types.

    Returns:
    - None: No value is returned.

    If the game has started, it will only display the menu options.
    If the game has not started, it will additionally display a welcome
    message and suit types to choose from as well as the menu options.
    """
    if bool_flag == False:
        count = 1
        print("Welcome to the game!\n")
        print("The following are avaliable types of suits to select from:")
        for suit in suit_types:
            print(" " + str(count) + " -" + str(suit))
            count = count + 1
        print("\nTo select the suit type in 1 (to start game) followed"
              + " by a space and the suit desired (e.g. 1 1)\n")
    print("Please enter your option choice:")
    print(" 1) Start Game \n 2) Pick a Card \n 3) Shuffle Deck"
          + "\n 4) Show My Cards \n 5) Check Win/Lose \n 6) Exit")
```

Task 2. Create Deck function

In [141…
```python
def create_deck(deck, suits, values):
    """
    Create a deck of cards by combining suits and values.

    Parameters:
    - deck (list): A list of cards.
    - suits (list): A list of card suits.
    - values (list): A list of card values.

    Returns:
    - None: No value is returned.

    This function combines suits and values together to make a card which is
    then populated into the deck.
    """
    for suit in suits:
        for value in values:
            deck.append(str(suit)+str(value))
    print("*** Deck has been created! ***")
```

Task 3. Shuffle Deck function

In [142…
```python
def shuffle_deck(deck, suits):
    """
    Shuffle the deck of cards with specific rules.

    Parameters:
    - deck (list): A list of cards.
    - suits (list): A list of card suits.

    Returns:
    - None: No value is returned.

    This function shuffles the cards within the deck.
    As certain defined cards need to be in a specific position, it first
    removes them from the list before shuffling.'A' of the first suit in
    'suits' is then inserted as the first card, 'Q' of the second suit in
    'suits' as the second card and 'K' of the last suit in 'suits' as the
    last card.
    """
    global global_first
    global global_middle
    global global_last

    global_first = suits[0] + "A"
    global_middle = suits[1] + "Q"
    global_last = suits[len(suits)-1] + "K"
    middle_position = round((len(deck))//2)

    deck.remove(global_first)
    deck.remove(global_middle)
    deck.remove(global_last)

    random.shuffle(deck)

    deck.insert(0, global_first)
    deck.insert(middle_position, global_middle)
    deck.insert(len(deck), global_last)

    print("*** Deck has been shuffled! ***")
    print(deck)
```

Task 4. Pick Card function

In [143…
```python
def pick_card(deck):
    """
    Pick a random card from the deck.

    Parameters:
    - deck (list): A list of cards.

    Returns:
    - str: A string for the randomly selected card.

    This function picks a random card from the deck and returns it. The card chosen
    can be any card but the first, middle and last card of the deck originally
    specified in the shuffle_deck function.
    """
    flag = False
    while flag == False:
        chosen = random.randint(0,(len(deck))-1)
        if (chosen != global_first
                and chosen != global_middle
                and chosen != global_last):
            return deck[chosen]
```

Task 5. Show Cards function

In [144…
```python
def show_cards(player_cards):

    """
    Display the cards held by a player.

    Parameters:
    - player_cards (list): A list of cards held by the player.

    Returns:
    - None: No value is returned.

    This function turns the list of player's cards and turns them
    into a string to be displayed.
    """
    cards_view = "Cards held: "
    for card in player_cards:
        if cards_view == "Cards held: ":
            cards_view = cards_view + card
        else:
            cards_view = cards_view + ", " + card
    print(cards_view)
```

Task 6. Check Result function

In [151…
```python
def check_results(player_cards, robot_cards, suits):

    """
    Compare the cards held by the player and the robot
    to determine the game result.

    Parameters:
    - player_cards (list): A list of cards held by the player.
    - robot_cards (list): A list of cards held by the robot.
    - suits (list): A list of card suits.

    Returns:
    - bool: True if the player wins, False if the robot wins.

    This function takes the player's cards, robot's cards and the list of
    suits as inputs and compares the two hands to determine the game result
    based on the specified rules.
    """
    def num_occurance(cards, start_end):
        """
        Count the occurrences of values at the start or end of a list
        of cards.

        Parameters:
        - cards (list): A list of cards.
        - start_end (str): A string to specify if the start or end of the
                           string is looked at

        Returns:
        - list: If start_end is 'end',
                returns a list of counts for each unique value.
        - int: If start_end is 'start',
               returns the count of the second suit's value.

        This function counts the occurrences of values in the list of cards
        either at the start or end of each card depending on the specified
        in the start_end parameter.
        """
        val = []
        unique = set()
        occurance = []

        for card in cards:
            if start_end == "start":
                val.append(card[0:1])
            elif start_end == "end":
                val.append(card[1:])

        for item in val:
            unique.add(item)

        if start_end == "start":
            return val.count(suits[1])
        elif start_end == "end":
            for single in unique:
                occurance.append(val.count(single))
            return occurance

    def average(cards):
        """
        Calculates the average of the cards.

        Parameters:
        - cards (list): A list of cards.

        Returns:
        - float: If card is not empty, return the average.
        - int: If card is empty, return 0

        This function calculates the average of all the cards if cards
        is not empty. Average is calculated by totaling the numeric value
        of the cards and dividing by the count of how many there are.
        """
        val = []
        total = 0
        count = len(cards)

        for card in cards:
            item = card[1:]

            if item == 'A':
                item = '1'
            if item == 'J':
                item = '11'
            if item == 'Q':
                item = '12'
            if item == 'K':
                item = '13'
            total = total + int(item)

        if count > 0:
            return float(total)/float(count)
        else:
            return 0

    suits_num = len(suits)

    if player_cards and not robot_cards:
        print("*** Robot has no cards. ***")
        return True
    elif player_cards and robot_cards:
        if (suits_num in num_occurance(player_cards, "end")
                and suits_num in num_occurance(robot_cards, "end")):
            print("*** Both have all the same value card for all suits. ***")
            return True
        elif suits_num in num_occurance(robot_cards, "end"):
            print("*** Robot has all the same value card for all suits. ***")
            return False
        elif suits_num in num_occurance(player_cards, "end"):
            print("*** Player has all the same value card for all suits. ***")
            return True
        elif ((suits_num - 1) in num_occurance(player_cards, "end")
                and (suits_num - 1) in num_occurance(robot_cards, "end")):
            print("*** Both have all but one of the "
                  + "same value card for all suits. ***")
            return True
        elif (suits_num - 1) in num_occurance(robot_cards, "end"):
            print("*** Robot has all but one of the "
                  + "same value card for all suits. ***")
            return False
        elif (suits_num - 1) in num_occurance(player_cards, "end"):
            print("*** Player has all but one of the "
                  + "same value card for all suits. ***")
            return True
        elif (num_occurance(player_cards, "start")
                > num_occurance(robot_cards, "start")):
            print("*** Player has more occurance of the second suit "
                  + suits[1] + ". ***")
            return True
        elif (num_occurance(player_cards, "start")
                < num_occurance(robot_cards, "start")):
            print("*** Robot has more occurance of the second suit "
                  + suits[1] + ". ***")
            return False
        elif average(player_cards) > average(robot_cards):
            print("*** Player has a higher average. ***")
            return True
        elif average(player_cards) < average(robot_cards):
            print("*** Robot has a higher average. ***")
            return False
        else:
            print("*** No conditions satisfied. ***")
            return False
    else:
        print("*** Not enough cards to play. ***")
        return False
```

Task 7. Play Game function

In [152…

```
def play_game():
    """
    Runs the game.

    Parameters:
    - No parameters are required.

    Returns:
    - None: No value is returned.

    This function calls all other functions to create a card game.
    Functions include game_menu, create_deck, shuffle_deck,
    pick_card, show_card and check_results. User inputs are
    captured to decide on the next action until the game is
    exited by choice of the player.
    """
    deck = []
    values = ["2", "3", "4", "5", "6", "7", "8",
              "9", "10", "J", "Q", "K", "A"]
    suit_types = [("♥", "♦", "♣", "♠"),
                  ("😃", "😈", "😵", "🤢", "😨"),
                  ("🤡", "👿", "👹", "👻", "👽", "👾", "🤖")]
    suits = []
    player_cards = []
    robot_cards = []
    round_count = 1
    global_first = ""
    global_middle = ""
    global_last = ""
    flag_start = False
    flag_exit = False

    def round_end(player_cards, robot_cards, suits):
        """
        Displays the end of the round results.

        Parameters:
        - player_cards (list): A list of cards held by the player.
        - robot_cards (list): A list of cards held by the robot.
        - suits (list): A list of card suits.

        Returns:
        - None: No value is returned.

        This function displays the player's cards, the robot's cards,
        and the result of winning or losing to the player.
        """
        print("\n*** Player ***")
        show_cards(player_cards)
        print("*** Robot ***")
        show_cards(robot_cards)
        print("")
        if check_results(player_cards, robot_cards, suits) == True:
            print("\n*** Congratulations! You Won! ***")
        else:
            print("\n*** You Lose! ***")
        print("--------------------------------------------------------------")

    print("Round: " + str(round_count) +"\n")
    while flag_exit == False:

        game_menu(flag_start, suit_types)
        selection = input("\nPlease enter your selection: ")

        if flag_start == False:
            # if else for deck creation
            if len(selection) == 3 and selection[:2] == '1 ':
                if selection[2:3] == '1':
                    suits = suit_types[0]
                    create_deck(deck, suits, values)
                elif selection[2:3] == '2':
                    suits = suit_types[1]
                    create_deck(deck, suits, values)
                elif selection[2:3] == '3':
                    suits = suit_types[2]
                    create_deck(deck, suits, values)
                else:
                    print("\n!Error! Invalid input. "
                          + "Creating deck with the first suit type!")
                    suits = suit_types[0]
                    create_deck(deck, suits, values)
                shuffle_deck(deck, suits)
                flag_start = True
            elif selection == '1':
                print("\n*** Creating deck with the first suit type! ***")
                suits = suit_types[0]
                create_deck(deck, suits, values)
                shuffle_deck(deck, suits)
                flag_start = True
            elif selection[:1] == '1':
                print("\n!Error! Invalid input."
                      + "Creating deck with the first suit type!")
                suits = suit_types[0]
                create_deck(deck, suits, values)
                shuffle_deck(deck, suits)
                flag_start = True
            elif (selection[:1] == '2' or selection[:1] == '3'
                    or selection[:1] == '4' or selection[:1] == '5'):
                print("!Error! Deck has not been created yet. "
                      + "Creating deck with the first suit type!")
```

```python
                suits = suit_types[0]
                create_deck(deck, suits, values)
                shuffle_deck(deck, suits)
                flag_start = True
            elif selection[:1] == '6':
                print("*** Player has exited the game! ***")
                flag_exit = True
            else:
                print("\n!Error! Enter a valid input!\n")
        elif selection[:1] == '1':
            print("\n!Error! Game has already started!\n")
        elif selection[:1] == '2':
            # for player
            card = pick_card(deck)
            deck.remove(card)
            player_cards.append(card)
            print("\n*** Card has been picked. ***\n")
            # for robot
            random_num = random.randint(0,3)
            if random_num != 0:
                card = pick_card(deck)
                deck.remove(card)
                robot_cards.append(card)
        elif selection[:1] == '3':
            shuffle_deck(deck, suits)
        elif selection[:1] == '4':
            show_cards(player_cards)
        elif selection[:1] == '5':
            round_end(player_cards, robot_cards, suits)
            round_count = round_count + 1
            print("Round: " + str(round_count) +"\n")
            flag_start = False
            deck = []
            player_cards = []
            robot_cards = []
        elif selection[:1] == '6':
            print("\n*** Player has exited the game! ***")
            flag_exit = True
        else:
            print("\n!Error! Enter a valid input!\n")

        # for six card limit
        if len(player_cards) == 6:
            print("\n*** Player has 6 cards! Calculating Result! ***")
            round_end(player_cards, robot_cards, suits)
            round_count = round_count + 1
            print("Round: " + str(round_count) +"\n")
            flag_start = False
            deck = []
            player_cards = []
            robot_cards = []

play_game()
```

```
Round: 1

Welcome to the game!

The following are avaliable types of suits to select from:
 1 -('♥', '♦', '♣', '♠')
 2 -('😃', '😈', '😵', '🤢', '😨')
 3 -('🤡', '👿', '👹', '👻', '👽', '👾', '🤖')

To select the suit type in 1 (to start game) followed by a space and the suit desired (e.g. 1 1)

Please enter your option choice:
 1) Start Game
 2) Pick a Card
 3) Shuffle Deck
 4) Show My Cards
 5) Check Win/Lose
 6) Exit

Please enter your selection: 6
*** Player has exited the game! ***
```