Welcome to the card game by @sshe0094

-- Developer: Samrudh Rajesh Shetty-- -- Card Game for FIT9136 -- -- Monash ID number: 33354448 --

In [1]:
```python
import random
```

In [2]:
```python
values = ["2","3","4","5","6","7","8","9","10","J","Q","K","A"]
suit_1 = ["♥","♦","♣","♠"]
suit_2 = ["😎 ","😉 ","😇 ","🤓"]
suit_3 = ["🐶 ","🐲 ","🐯 ","🐼 "]
deck = []
```

Task 1. Game Menu function

In [3]:
```python
def game_menu():
 """
 Running this function gives us menu for the game

 Parameters
 No parameters are given

 Developer
 samrudhShetty

 Output
 No return values

 Examples
 -game_menu()
Welcome to the card game, Let's play:
 0. How to win?
 1. Start game
 2. Pick a card
 3. Shuffle deck
 4. Show me my cards
 5. Check win or lose
 6. Exit
 """
 print('''
Welcome to card game, you have the following options:
 0. Show suits available.
 1. Start game
 2. Pick a card
 3. Shuffle deck
 4. Show me my cards
 5. Check win or lose
 6. Exit ''')

#Test code block
#game_menu()
```

In [ ]:

Task 2. Create Deck function

In [4]:
```python
def create_deck(deck, suits, values)->None:
 """
 Running this function with the user based parameters, creates a deck of cards

 Parameters
 deck- a list of strings with the deck of all cards
 suits- a list with the suits of all the deck
 values- is a fixed list of values for the cards

 Developer
 samrudhShetty

 Output
 No return values

 Examples
 -create_deck(["🐶 ","🐲 ","🐯 "], ["2","3","4","5","6","7","8","9","10","J","Q","K","A"])
 ['2 of 🐶 ', '3 of 🐶 ', '4 of 🐶 ', '5 of 🐶 ', '6 of 🐶 ', '7 of 🐶 ', '8 of 🐶 ', '9 of 🐶 ', '10 of 🐶 ', 'J of 🐶 ', 'Q of 🐶 ', 'K of 🐶 ', 'A of 🐶 ', '2 of 🐲 ', '3 of 🐲 ',
 '4 of 🐲 ', '5 of 🐲 ', '6 of 🐲 ', '7 of 🐲 ', '8 of 🐲 ',
 '9 of 🐲 ', '10 of 🐲 ', 'J of 🐲 ', 'Q of 🐲 ', 'K of 🐲 ', 'A of 🐲 ', '2 of 🐯 ', '3 of 🐯 ', '4 of 🐯 ', '5 of 🐯 ',
 '6 of 🐯 ', '7 of 🐯 ', '8 of 🐯 ', '9 of 🐯 ', '10 of 🐯 ', 'J of 🐯 ', 'Q of 🐯 ', 'K of 🐯 ', 'A of 🐯 ']
 """
 for suit in suits:
    for value in values:
        card = f"{value} of {suit}"
        deck.append(card)

#Test code block
# create_deck(["🐶 ","🐲 ","🐯 "], ["2","3","4","5","6","7","8","9","10","J","Q","K","A"])
# print(deck)
```

In [5]:
```python
create_deck([],["🐶 ","🐲 ","🐯 "], ["2","3","4","5","6","7","8","9","10","J","Q","K","A"])
print(deck)
```

[]

Task 3. Shuffle Deck function

In [6]:
```python
def shuffle_deck(deck, suits)-> None:
 """
 Running this function shuffles the deck of cards of selected suit, within the given specifications

 Parameters
 deck- a list of strings with the deck of all cards
 suits- a list with the suits of all the deck
 values- is a fixed list of values for the cards

 Developer
 samrudhShetty

 Output
 No return values

 Examples
 - shuffle_deck(['4 of 🐯 ', '8 of 🐯 ', '3 of 🐲 ', 'Q of 🐶 ', 'A of 🐲 ', '6 of 🐲 ', '8 of 🐲 ', '7 of 🐲 ', '2 of 🐶 ',
 '10 of 🐶 ', '9 of 🐯 ', '9 of 🐶 ', 'J of 🐶 ', 'A of 🐶 ', '9 of 🐲 ', '5 of 🐲 ', 'A of 🐯 ', 'J of 🐲 ', '7 of 🐶 ',
 '10 of 🐲 ', 'Q of 🐲 ', '4 of 🐶 ', '2 of 🐯 ', 'Q of 🐯 ', '8 of 🐶 ', 'K of 🐲 ', '5 of 🐯 ', 'K of 🐯 ', '2 of 🐲 ',
 '5 of 🐶 ', '3 of 🐯 ', 'J of 🐯 ', 'K of 🐶 ', '6 of 🐯 ', '3 of 🐶 ', '4 of 🐲 ', '7 of 🐯 ', '6 of 🐶 ', '10 of 🐯 ']
 ,["🐶 ","🐲 ","🐯 "])
 --['A of 🐶 ', '2 of 🐲 ', '8 of 🐯 ', '3 of 🐯 ','10 of 🐶 ', '5 of 🐲 ', '3 of 🐶 ', '6 of 🐲 ', '2 of 🐯 ', '10 of 🐲 ', '5 of 🐯 ',
 '7 of 🐶 ', 'J of 🐯 ',  'J of 🐶 ', '3 of 🐲 ',  'Q of 🐲 ', '5 of 🐶 ', '8 of 🐲 ', '9 of 🐲 ', '8 of 🐶 ', '4 of 🐲 ',
 '9 of 🐯 ', '4 of 🐶 ', '2 of 🐶 ', '10 of 🐯 ',  '9 of 🐶 ', 'J of 🐲 ', '7 of 🐯 ', '6 of 🐯 ', '7 of 🐲 ', '4 of 🐯 ',
 '6 of 🐶 ', 'K of 🐯 ']
 """
 for cd in ['A', 'Q', 'K']:
#         cd is the card
   if cd + ' of ' + suits[0] in deck:
            deck.remove(cd + ' of ' + suits[0])
   if cd + ' of ' + suits[len(suits)//2] in deck:
            deck.remove(cd + ' of ' + suits[len(suits)//2])
   if cd + ' of ' + suits[-1] in deck:
            deck.remove(cd + ' of ' + suits[-1])
 random.shuffle(deck)
    # Insert the first, middle, and last cards back into the deck
 deck.insert(0, 'A' + ' of ' + suits[0])
 deck.insert(len(deck)//2, 'Q' + ' of ' + suits[len(suits)//2])
 deck.append('K' + ' of ' + suits[-1])
 return(deck)
#Test code block
#print(shuffle_deck(['4 of 🐯 ', '8 of 🐯 ', '3 of 🐲 ', 'Q of 🐶 ', 'A of 🐲 ', '6 of 🐲 ', '8 of 🐲 ', '7 of 🐲 ', '2 of 🐶 ', '10 of 🐶 ', '9 of 🐯 ', '9 of 🐶 ', 'J of 🐶 ', 'A of 🐶 ', '9 of 🐲 ', '5
```

In [7]:
```python
print(shuffle_deck(['4 of 🐯 ', '8 of 🐯 ', '3 of 🐲 ', 'Q of 🐶 ', 'A of 🐲 ', '6 of 🐲 ', '8 of 🐲 ', '7 of 🐲 ', '2 of 🐶 ', '10 of 🐶 ', '9 of 🐯 ', '9 of 🐶 ', 'J of 🐶 ', 'A of 🐶 ', '9 of 🐲 ', '5 o
```

['A of 🐶 ', '3 of 🐶 ', '6 of 🐯 ', 'J of 🐯 ', '5 of 🐯 ', '5 of 🐶 ', '10 of 🐯 ', '6 of 🐶 ', '5 of 🐲 ', '3 of 🐲 ', '2 of 🐶 ', '4 of 🐯 ', '4 of 🐲 ', '7 of 🐶 ', '9 of 🐶 ', 'Q of 🐲 ', '4 of 🐶
', 'J of 🐲 ', '6 of 🐲 ', '2 of 🐯 ', '9 of 🐯 ', '7 of 🐯 ', '10 of 🐲 ', '10 of 🐶 ', '8 of 🐲 ', '2 of 🐲 ', '8 of 🐯 ', '7 of 🐲 ', '9 of 🐲 ', '8 of 🐶 ', 'J of 🐶 ', '3 of 🐯 ', 'K of 🐯 ']

In [8]:
```python
print(shuffle_deck(deck,["🐶 ","🐲 ","🐯 "]))
```

['Q of 🐲 ', 'A of 🐯 ', 'K of 🐯 ']

Task 4. Pick Card function

In [9]:
```python
def pick_a_card(deck) -> str:
 """
 Running this function picks a random card from the deck of cards of selected suit, within the given specifications.

 Parameters
 deck- a list of strings with the deck of all cards

 Developer
 samrudhShetty

 Output
 string value- function picks a random card from the deck and shows it

 Examples
 - pick_a_card(['A of 🐶 ', '3 of 🐯 ', '2 of 🐲 ', '7 of 🐯 ', '5 of 🐯 ', '4 of 🐲 ', '6 of 🐯 ', '8 of 🐶 '])
 3 of 🐯
 """
 card = random.choice(deck)
 deck.remove(card)
 return card
# Test code
# print(pick_a_card(deck))
```

In [10]:
```python
# Test code
# print(pick_a_card(deck))
```

Task 5. Show Card function

In [11]:
```python
def show_cards(p_cards):
 """
 Running this function shows all the cards the player holds

 Parameters
 p_cards- Takes an list of string with the player card

 Developer
 samrudhShetty

 Output
 None
 No return values

 Examples
 print(show_cards(['A of 🐶 ', '3 of 🐯 ', '2 of 🐲 ', '7 of 🐯 ']))
 """
 if(len(p_cards)>0):
  print("Your hand:")
  print(p_cards)
 else:
  print("You haven't picked a card yet.")
 #Test code block
#show_cards(['A of 🐶 ', '3 of 🐯 ', '2 of 🐲 ', '7 of 🐯 '])
#show_cards([])
```

In [12]:
```python
# show_cards(['A of 🐶 ', '3 of 🐯 ', '2 of 🐲 ', '7 of 🐯 '])
```

Task 6. Check Result function

In [13]:
```python
def check_result(p_cards, r_cards, suits):
    """
    Running this function checks the rules of the game with the given card combinations

    Parameters:
    p_cards - A list of string with the player's card
    r_cards - A list of string with the robot's card
    suits - A list of string with the available suits

    Developer
    samrudhShetty

    Output:
    None. No return values.

    Examples:
    print(check_win(['A of 🐶 ', '3 of 🐯 ', '2 of 🐲 ', '7 of 🐯 '], [], ['🐶', '🐯', '🐲']))
    """

    # Rule 0a: If the player's card is empty, the game cannot proceed
    if len(p_cards) == 0:
        print("The player's card is empty")
        return

    # Rule 0b: If the robot picks no cards and the player picks a card, the player wins
    if len(p_cards) > 0 and len(r_cards) == 0:
        print("Your opponent didn't pick a card, so you win")
        return True

        # Rule 1: Check if the player has the same value card for all defined suits
    pv = list({card.split()[0] for card in p_cards})
    if len(pv) == 1:
        return True
    rv = list({card.split()[0] for card in r_cards})
    if len(rv) == 1:
        return False

    # Rule 2: If the player has a (suit length-1)-of-a-kind, the player wins
    p_val = [card.split()[0] for card in p_cards]
    p_val_ct = dict()

    for value in p_val:
        if value not in p_val_ct.keys():
            p_val_ct.update({value: 1})
            continue
        p_val_ct.update({value: p_val_ct[value] + 1})

    for key, value in p_val_ct.items():
        if value == 3:
            return True

        return False

    # Rule 3: If one suit has more cards than the others, the player with the most cards of that suit wins
    if len(suits) >= 2:
        suit = suits[0]
        player_count = len([card.split()[1] for card in p_cards if suit in card])
        robot_count = len([card.split()[1] for card in r_cards if suit in card])
        if player_count > robot_count:
            return True
        elif robot_count > player_count:
            return False

    # Rule 4: If the average value of the player's cards is higher than the robot's, the player wins
    player_sum = 0
    robot_sum = 0
    for card in p_cards:
        value = card.split()[0]
        if value == 'A':
            player_sum += 1
        elif value == 'J':
            player_sum += 11
        elif value == 'Q':
            player_sum += 12
        elif value == 'K':
            player_sum += 13
        elif value.isnumeric():
            player_sum += int(value)

    for card in r_cards:
        value = card.split()[0]
        if value == 'A':
            robot_sum += 1
        elif value == 'J':
            robot_sum += 11
        elif value == 'Q':
            robot_sum += 12
        elif value == 'K':
            robot_sum += 13
        elif value.isnumeric():
            robot_sum += int(value)

    player_avg = player_sum / len(p_cards)
    robot_avg = robot_sum / len(r_cards) if len(r_cards) > 0 else 0

    if player_avg > robot_avg:
        return True
    elif robot_avg > player_avg:
        return False

    # Player loses
    # return False
```

In [14]:
```python
# Rule 1 Check
# check_result(['A of 🐶 ', 'A of 🐯 ', 'A of 🐲 '],['2 of 🐶 ', 'K of 🐯 ', '4 of 🐲 '], ['🐶', '🐯', '🐲'] )
```

In [15]:
```python
#Rule 2
#check_result(['A of 🐶 ', '1 of 🐯 ', '5 of 🐲 '],['A of 🐶 ', 'A of 🐯 ', '10 of 🐲 '], ['🐶', '🐯', '🐲'] )
```

In [16]:
```python
#Rule 3
#check_result(['7 of 🐶 ', '10 of 🐯 ', '9 of 🐯 '],['1 of 🐶 ', '11 of 🐯 ', '10 of 🐲 '], ['🐶', '🐯', '🐲'] )
```

In [17]:
```python
#Rule 3
#check_result(['7 of 🐶 ', '10 of 🐯 ', '9 of 🐯 '],['1 of 🐶 ', '11 of 🐯 ', '10 of 🐲 '], ['🐶', '🐯', '🐲'] )
```

Task 7. Play Game function

In [ ]:
```python
def play_game():
 """
    Running this function launches the game.

    Parameters:
    None

    Developer
    samrudhShetty

    Output:
    None. No return values.
    """
print(""""
Welcome to card game, you have the following options:
 0. What suits do we have to play with?
 1. Start game
 2. Pick a card
 3. Shuffle deck
 4. Show me my cards
 5. Check win or lose
 6. Exit """)
print("""
The available suits in the game are
-> Suit1 = ["♥","♦","♣","♠"]
-> Suit2 = ["😎 ","😉 ","😇 ","🤓"]
-> Suit3 = ["🐶 ","🐲 ","🐯 ","🐼 "]

To select any of the 3 available suits press 1 to choose option 1.Start Game
followed by the suit you wish to choose
("1 2" to start the game and choose Suit2)
If you go "1" the game automatically chooses Suit1

The possible values of the cards are as follows-
-> Values = ["2","3","4","5","6","7","8","9","10","J","Q","K","A"]""")

# Game function variables
on_game = True
suit1 = ["♥","♦","♣","♠"]
suit2 = ["😎 ","😉 ","😇 ","🤓"]
suit3 = ["🐶 ","🐲 ","🐯 ","🐼 "]
values = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"]

# Variables used between various functions and segments of the game
renew_game = False
play_suit = []
play_deck = []
p_cards = []
r_cards = []

# while loop that will execute game until the user chooses to terminate it
while (on_game):
    p_choice = input("Choose an option:")
    p_choices = p_choice.split(" ")
    if (len(p_choices) == 1 and len(p_choice) > 2):
        print("You can only enter multiple values if they are seperated by a space")
        game_menu()
        continue

    if (len(p_choices) > 2):
        print("You can only enter 2 characters")
        game_menu()
        continue

    if (len(p_choices) == 2):
        if (p_choices[0] == '1'):
            # start game
            if (len(play_deck) > 1):
             print("Game's already in progress, You must pick an option from 2 to 6 to continue")
             game_menu()
             continue
            if (p_choices[1] == '1'):
                suit_len = input("How many suit element types do you want? (Max4)")
                if (suit_len.isnumeric() and int(suit_len) >= 2):
                    play_suit = suit1[:int(suit_len)]
                else:
                    print("Invalid suit count given. Choosing suit for you...")
                    play_suit = suit1
            elif (p_choices[1] == '2'):
                print(suit2)
                suit_len = input("How many suit element types do you want?")
                if (suit_len.isnumeric() and int(suit_len) >= 2):
                    play_suit = suit2[:int(suit_len)]
                else:
                    print("Invalid suit count given. Choosing suit for you...")
                    play_suit = suit2
            elif (p_choices[1] == '3'):
                print(suit3)
                suit_len = input("How many suit element types do you want?")
                if (suit_len.isnumeric() and int(suit_len) >= 2):
                    play_suit = suit3[:int(suit_len)]
                else:
                    print("Invalid suit count given. Choosing suit for you...")
                    play_suit = suit3
            else:
                print("Valid input required.")
                game_menu()
                continue
            print("Your suit subset is:", play_suit)
            create_deck(play_deck, play_suit, values)
            shuffle_deck(play_deck, play_suit)
        elif (p_choices[0] != '1'):
            p_choice = p_choices[0]
        else:
            print("Valid input required.")
            game_menu()
            continue
    if (len(p_choice) >= 1):
        if (p_choice.isnumeric()):
#             if (p_choice == '0'):
#                 print(f"""The rules to win are as follows-
#                         -The player holds the same value card for all the defined suits.
#                         -The player has the same values for at least the total defined suits
#                           minus one.
#                         -The player holds more cards from the suit in position 2 than the robot. Note: if the
#                           suits=["♥", "♦", "♣", "♠"], the second suit is "♦".
#                         -The player holds a higher average of the card's value than robot.
#                                 """)
            if (p_choice == '0'):
                print(f"""
                    The suits available-
             suit1 = ["♥","♦","♣","♠"]
             suit2= ["😎 ","😉 ","😇 ","🤓"]
             suit3 = ["🐶 ","🐲 ","🐯 ","🐼 "]
             Suit picked by you-
                    chosen suit = {play_suit}
                    """)
            elif (p_choice == '1'):
                # start a game
                if (len(play_deck) > 1):
                    print("Game's already in progress, You must pick an option from 1 to 6 to continue")
                    game_menu()
                    continue
                suit_len = input("How many suit element types do you want? (Max4)")
                if (suit_len.isnumeric() and int(suit_len) >= 2):
                    play_suit = suit1[:int(suit_len)]
#                     print(play_suit)
                else:
                    print("Invalid suit count given. Choosing suit for you...")
                    play_suit = suit1
                create_deck(play_deck, play_suit, values)
                shuffle_deck(play_deck, play_suit)
                print("Game in progress! Do you want to continue to pick card or view cards?")
            elif (p_choice == '2'):
                # pick a card
                print(play_deck)
                if (len(play_deck) == 0):
                    print("You have to choose option 1 to be able to start to pick a card from the deck")
                    game_menu()
                    continue
                p_pic_card = pick_a_card(play_deck)
#                 print(len(play_deck))
                print("The card you picked is ", p_pic_card)
                p_cards.append(p_pic_card)
                if( len(play_deck)>0):
                  if (random.randint(0,1)):
                      r_pic_card = pick_a_card(play_deck)
                      r_cards.append(r_pic_card)
                      #print(r_pic_card)
            elif (p_choice == '3'):
                # shuffle the deck
                if (len(play_deck) == 0):
                 print("You cant access this option before choosing option 1")
                 game_menu()
                 continue
                shuffle_deck(play_deck, play_suit)
                print("\n")
            elif (p_choice == '4'):
                # show my cards
                if (len(play_deck) == 0):
                    print("You cant access this option before choosing option 1")
                    game_menu()
                    continue
                show_cards(p_cards)
            elif (p_choice == '5'):
                # check win or lose
                if (len(play_deck) == 0):
                    print("You cant access this option before choosing option 1")
                    game_menu()
                    continue
                player_win_lose = check_result(p_cards, r_cards, play_suit)
                if (player_win_lose):
                    print("You win!!")
                    renew_game = True
                else:
                    print("You lost to your opponent.")
                    renew_game = True
            elif (p_choice == '6'):  # exit game
                print("Come back soon!")
                break
            else:
                print("Please enter a valid input from 0 through 6")

    ## The game is to be restarted when the player gets 6 cards or the player enters check_result function
    if (len(p_cards) == 6 or renew_game):
        renew_game = False
        if (len(p_cards) == 6):
            print("You have reached your card limit displaying result....")
            player_win_lose = check_result(p_cards, r_cards, play_suit)
            if (player_win_lose):
                print("You win!!")
            else:
                print("You lost to your opponent.")
        play_deck = []
        play_suit = []
        p_cards = []
        r_cards = []
        print("|The game is being restarted |")


    game_menu()
play_game()
```

```
"
Welcome to card game, you have the following options:
 0. What suits do we have to play with?
 1. Start game
 2. Pick a card
 3. Shuffle deck
 4. Show me my cards
 5. Check win or lose
 6. Exit

The available suits in the game are
-> Suit1 = ["♥","♦","♣","♠"]
-> Suit2 = ["😎 ","😉 ","😇 ","🤓"]
-> Suit3 = ["🐶 ","🐲 ","🐯 ","🐼 "]

To select any of the 3 available suits press 1 to choose option 1.Start Game
followed by the suit you wish to choose
("1 2" to start the game and choose Suit2)
If you "1" the game automatically chooses Suit1

The possible values of the cards are as follows-
-> Values = ["2","3","4","5","6","7","8","9","10","J","Q","K","A"]
```