FIT9136: Assignment 1 - Card Game

Name: Alexandra Goh Jia Xin

Student ID: 29796431

Creation Date: 14th March 2023

Last modified: 30th March 2023

Description of program: This python script enables a user (player) to play a Card Game against a computer (robot).

Task 1. Game Menu function

In [ ]:
```
def game_menu():
    print("\n*****************************************************************************************************************")
    print("\nWelcome to the Monash card game! To access the following game options, enter its number as listed below: ")
    print("1 : start game")
    print("2 : shuffle deck")
    print("3 : pick a card")
    print("4 : show my cards")
    print("5 : check win or lose")
    print("6 : exit")

    print("\nFirst, choose one suit type from the following below: ")
    print("Deck 1 = ['♥', '♦', '♣', '♠']")
    print("Deck 2 = ['😀', '😈', '😵', '🤢', '😨']")
    print("Deck 3 = ['🤡', '👹', '👺', '👻', '👽', '👾', '🤖']")

    print("\nEnter '1 1' for Deck 1, '1 2' for Deck 2 or '1 3' for Deck 3.")
    print("\nThe first number would be your game option, and your second number would be your selected deck. For instance, '1 2'
    print("\nYou must ensure there is a space between your first and second number, when you are selecting a deck. Example: '12'
    print("\n*****************************************************************************************************************
```

In [ ]:
```
# Testing Game Menu function
game_menu()
```

Task 2. Create Deck function

In [ ]:
```
def create_deck(deck, suits, values):

    for suit in suits:
        new_deck = []
        for suit_type in suit:
            for value in values:
                new_deck.append("{} of {}".format(value, suit_type))
        deck.extend(new_deck)

    print(f"\nChosen deck: {deck}")
```

In [ ]:
```
# Testing Create Deck function with valid input/output:
"""
suits = ['♥', '♦', '♣', '♠']

values = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"]

deck = []

create_deck(deck, suits, values)

Returns correct output:
Each element of each suit printed only once in the deck.

"""

# Testing Create Deck function with invalid input/output:
"""
suits = ['♥', '♦', '♣', '♠']

values = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"]

create_deck(deck, suits, values)

Returns incorrect output:
Each element of each suit printed twice when deck is not specified as deck []

"""
```

Task 3. Shuffle Deck function

In [ ]:
```python
import random

def shuffle_deck(deck, suits):

    n = len(deck)

    # Find the index of the specific cards in the unshuffled deck
    first_suit = suits[0]
    second_suit = suits[1]
    last_suit = suits[-1]

    # Perform multiple shuffles until specific cards are in their desired positions
    while True:
        random.shuffle(deck)

        # Find index of the specific cards in the shuffled deck
        first_card = deck.index("A of " + first_suit) if "A of " + first_suit in deck else None
        middle_card = deck.index("Q of " + second_suit) if "Q of " + second_suit in deck else None
        last_card = deck.index("K of " + last_suit) if "K of " + last_suit in deck else None

        # Check if specific cards are in their desired positions
        if (first_card == 0 or first_card is None) and (middle_card == round((n+1)/2,0) or middle_card is None) and (last_card ==
            break

    print("\nDeck shuffled:", deck)
```

In [ ]:
```python
# Testing Shuffle Deck function with valid input/output:
"""
suits = ['♥', '♦', '♣', '♠']
deck = ['2 of ♥', '3 of ♥', '4 of ♥', '5 of ♥', '6 of ♥', '7 of ♥', '8 of ♥', '9 of ♥', '10 of ♥', 'J of ♥', 'Q of ♥', 'K of ♥',

shuffle_deck(deck, suits)

Returns correct output:
Everytime deck is shuffled: 'A of ♥' is at first position, 'Q of ♦' is at middle position and 'K of ♠' is at last position.

"""
```

```
"""
suits = ['♥', '♦', '♣', '♠']
deck = ['2 of ♥', '3 of ♥', '4 of ♥', '5 of ♥', '6 of ♥', '7 of ♥', '8 of ♥', '9 of ♥', '10 of ♥', 'J of ♥', 'Q of ♥', 'K of ♥',

shuffle_deck(deck, suits)

Returns correct output:
Even if one of the specific cards are removed from the deck (e.g. K of ♠): the other two cards (A of ♥) and (Q of ♦) still retair

"""

# Testing Shuffle Deck function with invalid input/output:
"""
suits = ['♥', '♦', '♣', '♠']
deck = []

shuffle_deck(deck, suits)

Returns incorrect output:
If deck is empty, nothing is shuffled inside and deck remains empty.

"""
```

Task 4. Pick Card function

In [ ]:
```
import random

def pick_card(deck):
    selected_index = random.randint(0, len(deck)-1)
    selected_card = deck[selected_index]
    deck = deck[:selected_index] + deck[selected_index+1:]
    return selected_card, deck
```

In [ ]:
```
# Testing Pick Card function with valid input/output:
"""
deck = ['2 of ♥', '3 of ♥', '4 of ♥', '5 of ♥', '6 of ♥', '7 of ♥', '8 of ♥', '9 of ♥', '10 of ♥', 'J of ♥', 'Q of ♥', 'K of ♥',

selected_card, deck = pick_card(deck)

print("Picked card:", selected_card)
print("Remaining deck:", deck)
```

```
Returns correct output:
Example: 8 of ♠ is selected and remaining deck does not consists of '8 of ♠'.

"""

# Testing Pick Card function with invalid input/output:
"""
deck = ['2 of ♥', '3 of ♥', '4 of ♥', '5 of ♥', '6 of ♥', '7 of ♥', '8 of ♥', '9 of ♥', '10 of ♥', 'J of ♥', 'Q of ♥', 'K of ♥',

deck = pick_card(deck)

print("Picked card:", selected_card)
print("Remaining deck:", deck)

Returns incorrect output:
Example: 3 of ♠ is selected but remaining deck still consists of '3 of ♠' if selected_card is not passed as a return value.

"""

"""
deck = ['2 of ♥', '3 of ♥', '4 of ♥', '5 of ♥', '6 of ♥', '7 of ♥', '8 of ♥', '9 of ♥', '10 of ♥', 'J of ♥', 'Q of ♥', 'K of ♥',

selected_card = pick_card(deck)

print("Picked card:", selected_card)
print("Remaining deck:", deck)

Returns incorrect output:
Example: If deck is not passed as a return value, every card is picked instead of just one.

"""
```

Task 5. Show Cards function

In [ ]:
```
def show_cards(player_cards):
    print("Your total cards are:", player_cards)
```

In [ ]:
```
#Testing Show Cards function with valid input/output:
"""
player_cards = ['2 of ♦', 'A of ♥', 'A of ♦', 'K of ♠', 'J of ♣']
```

```
show_cards(player_cards)

Returns correct output:
All cards in player_cards list are shown one after another.

"""

#Testing Show Cards function with valid input/output:
"""
player_cards = []
show_cards(player_cards)

Returns correct output:
If player has not chosen any cards, total cards are shown as empty.

"""

#Testing Show Cards function with invalid input/output:
"""
player_cards = '2 of ♦' 'A of ♥' 'A of ♦' 'K of ♠' 'J of ♣'
show_cards(player_cards)

Returns incorrect output:
If player_cards not defined as a list of cards or as a tuple, show_cards function will print out player_cards in a non-readable f

"""
```

Task 6. Check Result function

In [ ]:

```
def check_result(player_cards, robot_cards, suits):

    player_wins = None
    robot_wins = None

    # Rule 1
    for value in ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']:
        count_player = 0
        count_robot = 0
        for suit in suits:
            if "{} of {}".format(value, suit) in player_cards:
                count_player += 1
```

```
            if "{} of {}".format(value, suit) in robot_cards:
                count_robot += 1
        if count_player == len(suits) and count_robot != len(suits):
            player_wins = True
            robot_wins = False
        elif count_robot == len(suits) and count_player != len(suits):
            player_wins = False
            robot_wins = True

    # Rule 2

    if player_wins is None:
        for value in ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']:
            count_player = 0
            count_robot = 0
            for suit in suits:
                if "{} of {}".format(value, suit) in player_cards:
                    count_player += 1
                if "{} of {}".format(value, suit) in robot_cards:
                    count_robot += 1

            if count_player == (len(suits)-1) and count_robot != (len(suits)-1):
                player_wins = True
                robot_wins = False
            elif count_robot == (len(suits)-1) and count_player != (len(suits)-1):
                player_wins = False
                robot_wins = True


    # Rule 3

    if player_wins is None and robot_wins is None:

        if len(suits) == 2:
            card_suits = {"first": suits[0], "second":suits[1]}
        elif len(suits) == 3:
            card_suits = {"first": suits[0], "second":suits[1], "third": suits[2]}
        elif len(suits) == 4:
            card_suits = {"first": suits[0], "second":suits[1], "third": suits[2], "fourth":suits[3]}
        elif len(suits) == 5:
            card_suits = {"first": suits[0], "second":suits[1], "third": suits[2], "fourth":suits[3], "fifth": suits[4]}
        elif len(suits) == 6:
            card_suits = {"first": suits[0], "second":suits[1], "third": suits[2], "fourth":suits[3], "fifth": suits[4], "sixth"
        else:
```

```
            card_suits = {"first": suits[0], "second":suits[1], "third": suits[2], "fourth":suits[3], "fifth": suits[4], "sixth"

        player_count = 0
        robot_count = 0

        for card in player_cards:
            if card.endswith(card_suits["second"]):
                player_count +=1
        for card in robot_cards:
            if card.endswith(card_suits["second"]):
                robot_count +=1

        if player_count > robot_count:
            player_wins = True
            robot_wins = False
        elif robot_count > player_count:
            player_wins = False
            robot_wins = True
        else:
            player_wins = None
            robot_wins = None


    # Rule 4

    if player_wins is None and robot_wins is None:
        card_values = {'A': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9, '10': 10, 'J': 11, 'Q': 12, 'K':

        player_values = [card_values[card.split()[0]] for card in player_cards]
        player_average = sum(player_values) / len(player_cards)

        if len(robot_cards) == 0:
            robot_average = 0
        else:
            robot_values = [card_values[card.split()[0]] for card in robot_cards]
            robot_average = sum(robot_values) / len(robot_cards)

        if player_average > robot_average:
            player_wins = True
            robot_wins = False
        elif robot_average >= player_average:
            player_wins = False
            robot_wins = True
```

```
    if player_wins is True and robot_wins is False:
        print("🏆 Congratulations! You have won the game 🏆")
        print("\nYour cards:", player_cards)
        print("\nRobot's cards:", robot_cards)
    elif player_wins is False and robot_wins is True:
        print("😨 Sorry! Better luck next time 😨")
        print("\nYour cards:", player_cards)
        print("\nRobot's cards:", robot_cards)

    return player_wins
    return robot_wins
```

In [ ]:

```
# Testing Check Result function - Rule 1:
"""
player_cards = ["K of ♥", "K of ♦", "K of ♣", "K of ♠"]
robot_cards = ["A of ♥", "4 of ♦", "10 of ♣", "Q of ♠"]
suits = ["♥", "♦", "♣", "♠"]

check_result(player_cards, robot_cards, suits)

Returns correct output: player wins
"""

# Testing Check Result function - Rule 2:
"""
player_cards = ["K of ♥", "Q of ♦", "10 of ♣", "5 of ♠"]
robot_cards = ["J of ♥", "J of ♦", "J of ♣", "A of ♠"]
suits = ["♥", "♦", "♣", "♠"]

check_result(player_cards, robot_cards, suits)

Returns correct output: robot wins
"""

# Testing Check Result function - Rule 3:

"""
player_cards = ["2 of ♥", "4 of ♦", "2 of ♦", "3 of ♦"]
robot_cards = ["5 of ♥", "10 of ♦", "5 of ♣", "Q of ♦"]
suits = ["♥", "♦", "♣", "♠"]

check_result(player_cards, robot_cards, suits)
```

```
Returns correct output: player wins

"""

# Testing Check Result function - Rule 4:

"""
player_cards = ["2 of ♥", "4 of ♦", "2 of ♣", "3 of ♠"]
robot_cards = ["5 of ♥", "10 of ♦", "J of ♣", "Q of ♥"]
suits = ["♥", "♦", "♣", "♠"]

check_result(player_cards, robot_cards, suits)

Returns correct output: robot wins

"""
```

Task 7. Play Game function

In [ ]:

```
def play_game():

    def elements_1(suits):
        while True:
            elements = input("\nChoose the number of elements you wish to play with by entering a number between 2 to 4: ")
            if elements.isdigit() and int(elements) in [2, 3, 4]:
                new_suits = random.sample(suits1, int(elements))
                print(f"\nChosen elements: {new_suits}")
                return new_suits
            else:
                print("\n ❗ Invalid input. Enter a number only between 2 to 4.")

    def elements_2(suits):
        while True:
            elements = input("\nChoose the number of elements you wish to play with by entering a number between 2 to 5: ")
            if elements.isdigit() and int(elements) in [2, 3, 4, 5]:
                new_suits = random.sample(suits2, int(elements))
                print(f"\nChosen elements: {new_suits}")
                return new_suits
            else:
                print("\n ❗ Invalid input. Enter a number only between 2 to 5.")
```

```python
def elements_3(suits):
    while True:
        elements = input("\nChoose the number of elements you wish to play with by entering a number between 2 to 7: ")
        if elements.isdigit() and int(elements) in [2, 3, 4, 5, 6, 7]:
            new_suits = random.sample(suits3, int(elements))
            print(f"\nChosen elements: {new_suits}")
            return new_suits
        else:
            print("\n❗ Invalid input. Enter a number only between 2 to 7.")


deck_selected = None
deck = []
player_cards = []
robot_cards = []
num_player_cards = 0
num_robot_cards = 0

values = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']

suits1 = ['♥', '♦', '♣', '♠']
suits2 = ['😀', '😈', '😵', '🤢', '😨']
suits3 = ['🤡', '👺', '👹', '👻', '🐸', '👾', '🤖']
suits = [suits1, suits2, suits3]

exit_game = False

while exit_game is False:
    game_menu()

    input_selection = input("\nPlease enter your selection: ")

    if input_selection == "1":
        if not deck_selected:
            suits = elements_1(suits)
            create_deck(deck, suits, values)
            shuffle_deck(deck, suits)
            deck_selected = True
            print ("\nStarting game with Deck 1... continue to pick your cards by entering '3', or reshuffle the deck by ente
        else:
            print("\n❗ Deck already selected. Proceed to pick your cards by entering '3', or reshuffle the deck by entering

    elif input_selection.startswith("1") and input_selection != "1":
```

```
            if not deck_selected:
                split_input = input_selection.split()
                if len(split_input) == 2:
                    suit_type = split_input[1]

                    if suit_type in ["1", "2", "3"]:
                        if suit_type == "1":
                            suits = elements_1(suits)
                            create_deck(deck, suits, values)
                            shuffle_deck(deck, suits)
                            deck_selected = True
                        elif suit_type == "2":
                            suits = elements_2(suits)
                            create_deck(deck, suits, values)
                            shuffle_deck(deck, suits)
                            deck_selected = True
                        else:
                            suits = elements_3(suits)
                            create_deck(deck, suits, values)
                            shuffle_deck(deck, suits)
                            deck_selected = True

                        print("\nStarting game with Deck", suit_type, "... continue to pick your cards by entering '3', or reshu

                    else:
                        print("\n ❗ Invalid input. Enter a valid suit type as your second number, which is either '1', '2' or '3'
                else:
                    print("\n ❗ Invalid input. Separate your two numbers when starting game and selecting a deck. Example: '1 2'
            else:
                print("\n ❗ Deck already selected. Proceed to pick your cards by entering '3', or reshuffle the deck by entering


        elif input_selection.startswith("2 ") or input_selection == "2" and len(input_selection) in [1,2]:
            if deck_selected is True:
                print("\nShuffling deck...please be patient")
                shuffle_deck(deck, suits)
            else:
                print("\n ❗ Deck not selected. Select a deck first before shuffling the deck.")


        elif input_selection.startswith("3 ") or input_selection == "3" and len(input_selection) in [1,2]:
            if deck_selected is True:
                if num_player_cards <6 and exit_game is False:
                    print("\nPicking a card...")
```

```
                selected_card, deck = pick_card(deck)
                player_cards.append(selected_card)

                num_player_cards += 1
                if num_player_cards == 6:
                    print("\n You have reached the card limit. Now let's see your results!")
                    check_result(player_cards, robot_cards, suits)

                robot_card_probability = 50
                if random.randint(1, 100) <= robot_card_probability and len(deck) > 0:
                    robot_selected_index = random.randint(0, len(deck)-1)
                    robot_selected_card = deck[robot_selected_index]
                    deck = deck[:robot_selected_index] + deck[robot_selected_index+1:]
                    robot_cards.append(robot_selected_card)

                    num_robot_cards += 1
                else:
                    robot_selected_card = None

                print("\nPicked card:", selected_card)
                print("\nRobot picked card:", robot_selected_card)
                print("\nRemaining deck:", deck)

            else:
                print("\n ❗ You have already picked the maximum of 6 cards.")
        else:
            print("\n ❗ Deck not selected. Select a deck first before picking a card.")

    if num_player_cards == 6:
        while True:
            play_again = input("\nDo you wish to play again? Enter 'Y' for Yes or 'N' for No:")
            if play_again.upper() == "N":
                print("\nThanks for playing and have a nice day!")
                print("\nExiting game...")
                exit_game = True
                break
            elif play_again.upper() not in ['Y', 'N']:
                print("\n ❗ Invalid input. Enter either 'Y' for Yes or 'N' for No.")
                continue
            else:
                print("\nRestarting game...")
                deck_selected = None
                deck.clear()
```

```
                player_cards.clear()
                robot_cards.clear()
                num_player_cards = 0
                num_robot_cards = 0
                break


        elif input_selection.startswith("4 ") or input_selection == "4" and len(input_selection) in [1,2]:
            if deck_selected is True:
                print("\nShowing your cards...")
                show_cards(player_cards)
            else:
                print("\n ❗ Deck not selected. Select a deck first before picking a card.")


        elif input_selection.startswith("5 ") or input_selection == "5" and len(input_selection) in [1,2]:
            if deck_selected is True and len(player_cards)>=1:
                print("\nChecking win or lose...")
                check_result(player_cards, robot_cards, suits)
                print("\nYou may continue to pick your cards (maximum cards you can hold in total is 6) by entering '3' or exit t
            else:
                print("\n ❗ Deck or cards not selected. Select a deck first before picking a card; otherwise, select a card first


        elif not input_selection.isnumeric():
            print("\n ❗ Invalid input. Enter only numbers between 1-6 when choosing your options.")


        elif int(input_selection)<1 or int(input_selection)>6:
            print("\n ❗ Invalid input. Enter only numbers between 1-6 when choosing your options.")


        elif input_selection.startswith("6 ") or input_selection == "6" and len(input_selection) in [1,2]:
            print("\nThanks for playing and have a nice day!")
            print("\nExiting game...")
            break
```

In [ ]:

```
# Testing Play Game function:
"""
When starting game:
1) If player hasn't picked a deck yet when first entering input_selection, message will print "Deck not selected. Select a deck t
2) If player has already picked a deck but selects '1' again, message will print "Deck already selected. Proceed to pick your car
```

```
When choosing suits & elements:
1) If player enters invalid number (not '1', '2', '3'), message will print "Invalid input. Enter a valid suit type as your second
2) If player trying to enter two numbers and not having space between, message will print "Invalid input. Separate your two numbe
3) If player enters invalid number not within the range when choosing elements, message will print "Invalid input. Enter a number

When choosing to play again or not:
1) If player does not enter "Y, y, N, or n": message will print "Invalid input. Enter either 'Y' for Yes or 'N' for No."

If player puts characters, multiple integers or any number not within the game options menu (1-6): message will print "Invalid in
"""
```

In [ ]:
```
play_game()
```