پروژه مرحله 2 - کتاب کار گام به گام

در زیر مجموعه‌ای از مراحل برای شما آورده شده است تا سعی کنید بازی پروژه مرحله 2 بلک جک را ایجاد کنید!

بازی

برای بازی یک دست بلک جک، باید مراحل زیر را دنبال کنید:

1. یک دسته کارت 52 تایی ایجاد کنید.
2. دسته کارت را مخلوط کنید.
3. از بازیکن شرط خود را بپرسید.
4. اطمینان حاصل کنید که شرط بازیکن از تعداد تراشه‌های در دسترسش بیشتر نشود.
5. دو کارت به دلال و دو کارت به بازیکن بدهید.
6. فقط یکی از کارت‌های دلال را نشان دهید و دیگری مخفی بماند.
7. هر دو کارت بازیکن را نشان دهید.
8. از بازیکن بپرسید که آیا می‌خواهند کارت بگیرند یا خیر و کارت دیگری بگیرند.
9. اگر دست بازیکن زیادی بشود (بیش از 21)، بپرسید که آیا می‌خواهند دوباره کارت بگیرند یا خیر.
10. اگر بازیکن ایستاد، دست دلال را بازی کنید. دلال همیشه کارت بگیرد تا مقدار دلال برابر یا بیشتر از 17 شود.
11. برنده را تعیین کرده و تعدیل chips بازیکن را مطابق با آن انجام دهید.
12. از بازیکن بپرسید که آیا می‌خواهند دوباره بازی کنند یا خیر.

کارت‌های بازی

یک دسته استاندارد کارت بازی شامل چهار نوع (قلب، دیموند، پیک و تپه) و سیزده رتبه (از 2 تا 10 و سپس کارت‌های صورتی جک، کوین و کینگ و آس) است که در مجموع 52 کارت

دارد. جک‌ها، کوین‌ها و کینگ‌ها همه رتبه 10 دارند. آس‌ها رتبه 11 یا 1 دارند تا به 21 برسند بدون اینکه بیشتر از این مقدار شوند. به عنوان نقطه شروع برنامه، می‌توانید متغیرهایی را اختصاص دهید تا یک لیست از نوع‌ها، رتبه‌ها را ذخیره کنید و سپس از یک دیکشینری برای نگاشت رتبه‌ها به ارزش‌ها استفاده کنید.

بازی

Imports and Global Variables

گام ۱: ماژول random را وارد کنید. این ماژول برای مخلوط کردن دسته کارت قبل از توزیع استفاده خواهد شد. سپس متغیرهایی را برای ذخیره کردن انواع کارت‌ها، رتبه‌ها و ارزش‌ها تعریف کنید. شما می‌توانید سیستم خودتان را توسعه دهید یا می‌توانید سیستم ما را کپی کنید که در زیر آمده است. در نهایت، یک مقدار بولین تعریف کنید که برای کنترل حلقه‌های while استفاده می‌شود. این یک شیوه رایج است که برای کنترل جریان بازی استفاده می‌شود.

```
suits = ('Hearts', 'Diamonds', 'Spades', 'Clubs')
ranks = ('Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine', 'Te
```

In [4]:

```
import random

suits = ('Hearts', 'Diamonds', 'Spades', 'Clubs')
ranks = ('Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine', 'Ten', 'Jac
values = {'Two':2, 'Three':3, 'Four':4, 'Five':5, 'Six':6, 'Seven':7, 'Eight':8, 'Nir
          'Queen':10, 'King':10, 'Ace':11}

playing = True
```

تعاریف کلاس

در نظر بگیرید یک کلاس Card ایجاد کنید که هر شیء Card دارای نوع و رتبه است، سپس یک کلاس Deck برای نگهداری تمامی 52 شیء Card و مخلوط کردن آن‌ها و در نهایت یک کلاس Hand که شامل کارت‌هایی است که از دسته به هر بازیکن داده شده است.

گام ۲: ایجاد کلاس Card
یک شیء Card واقعاً تنها به دو ویژگی نیاز دارد: نوع و رتبه. شما می‌توانید یک ویژگی برای "ارزش" اضافه کنید - ما تصمیم گرفتیم ارزش را در زمان توسعه کلاس Hand مدیریت کنیم.
علاوه بر روش __init__ کارت، در نظر بگیرید یک روش __str__ اضافه کنید که هنگام درخواست چاپ یک کارت، یک رشته به شکل "Two of Hearts" را برمی‌گرداند.

In [1]:

```
class Card:

    def __init__(self, suit, rank):
        self.suit = suit
        self.rank = rank

    def __str__(self):
        return self.rank + ' of ' + self.suit
```

گام ۳: ایجاد کلاس Deck
در این قسمت می‌توانیم 52 شیء کارت را در یک لیست ذخیره کنیم که در آینده بتوانیم آنها را مخلوط کنیم. اما در ابتدا، باید تمامی 52 شیء کارت یکتایی را ایجاد کرده و آن‌ها را به لیستمان اضافه کنیم. تا زمانی که تعریف کلاس Card در کد ما وجود دارد، ما می‌توانیم شیء کارت را درون متد __init__ کلاس Deck بسازیم. در نظر بگیرید که با استفاده از حلقه‌ها بر روی دنباله‌های انواع و رتبه‌ها، هر کارت را بسازیم. این ممکن است درون متد __init__ کلاس Deck ظاهر شود:

```
for suit in suits:
    for rank in ranks:
```

علاوه بر متد __init__، ما می‌خواهیم متدهایی را اضافه کنیم تا دسته کارت‌هایمان را مخلوط کنیم و کارت‌ها را در طول بازی توزیع کنیم.

اختیاری: ممکن است هرگز نیازی به چاپ محتویات دسته کارت‌ها در طول بازی نداشته باشیم، اما داشتن امکان مشاهده کارت‌ها درون آن می‌تواند به رفع مشکلاتی که در طول توسعه به وجود می‌آید کمک کند. با این در نظر گرفته شود، در نظر بگیرید یک متد __str__ را به تعریف کلاس اضافه کنید.

In [2]:

```
class Deck:

    def __init__(self):
        self.deck = []  # start with an empty list
        for suit in suits:
            for rank in ranks:
                self.deck.append(Card(suit, rank))

    def __str__(self):
        deck_comp = ''
        for card in self.deck:
            deck_comp += '\n' + card.__str__()

        return 'The deck has: ' + deck_comp

    def shuffle(self):
        random.shuffle(self.deck)

    def deal(self):
        signle_card = self.deck.pop()
        return signle_card
```

آزمایش: فقط برای دیدن اینکه تا الان همه چیز کار می‌کند چه شکلی دسته کارت‌هایمان است، بیایید ببینیم!

In [5]:

```
test_deck = Deck()
print(test_deck)
```

```
The deck has:
Two of Hearts
Three of Hearts
Four of Hearts
Five of Hearts
Six of Hearts
Seven of Hearts
Eight of Hearts
Nine of Hearts
Ten of Hearts
Jack of Hearts
Queen of Hearts
King of Hearts
Ace of Hearts
Two of Diamonds
Three of Diamonds
Four of Diamonds
Five of Diamonds
Six of Diamonds
Seven of Diamonds
Eight of Diamonds
Nine of Diamonds
Ten of Diamonds
Jack of Diamonds
Queen of Diamonds
King of Diamonds
Ace of Diamonds
Two of Spades
Three of Spades
Four of Spades
Five of Spades
Six of Spades
Seven of Spades
Eight of Spades
Nine of Spades
Ten of Spades
Jack of Spades
Queen of Spades
King of Spades
Ace of Spades
Two of Clubs
Three of Clubs
Four of Clubs
Five of Clubs
Six of Clubs
Seven of Clubs
Eight of Clubs
Nine of Clubs
Ten of Clubs
Jack of Clubs
Queen of Clubs
King of Clubs
Ace of Clubs
```

عالی! حالا بیایید به کلاس Hand برویم.

گام ۴: ایجاد کلاس Hand

علاوه بر نگهداری از شیءهای کارتی که از دسته داده می‌شوند، کلاس Hand می‌تواند برای محاسبه ارزش این کارت‌ها با استفاده از فهرست ارزش‌هایی که در بالا تعریف شده‌اند، استفاده شود. همچنین در صورت لزوم باید برای ارزش آس‌ها تعدیلی انجام دهد.

In [6]:

```
class Hand:
    def __init__(self):
        self.cards = []  # start with an empty list as we did in the Deck class
        self.value = 0   # start with zero value
        self.aces = 0    # add an attribute to keep track of aces

    def add_card(self,card):
        self.cards.append(card)
        self.value += values[card.rank]

    def adjust_for_ace(self):
        pass
```

In [7]:

```
test_deck = Deck()
test_deck.shuffle()

test_player = Hand()
test_player.add_card(test_deck.deal())
test_player.add_card(test_deck.deal())

test_player.value
```

Out[7]:

```
18
```

In [8]:

```
for card in test_player.cards:
    print(card)
```

```
Ten of Clubs
Eight of Diamonds
```

In [9]:

```
class Hand:
    def __init__(self):
        self.cards = []  # start with an empty list as we did in the Deck class
        self.value = 0   # start with zero value
        self.aces = 0    # add an attribute to keep track of aces

    def add_card(self,card):
        self.cards.append(card)
        self.value += values[card.rank]

    def adjust_for_ace(self):
        while self.value > 21 and self.aces:
            self.value -= 10
            self.aces  -= 1
```

گام ۵: ایجاد کلاس Chips

علاوه بر دسته‌های کارت و دست‌ها، ما باید موجودی اولیه بازیکن، شرط‌ها و برنده‌شدن‌های مداوم یک بازیکن را نیز نگهداری کنیم. این کار می‌تواند با استفاده از متغیرهای عمومی انجام شود، اما با توجه به روح برنامه‌نویسی شیءگرا، به جای آن یک کلاس Chips بسازیم!

In [10]:

```
class Chips:

    def __init__(self, total = 100):
        self.total = total  # This can be set to a default value or supplied by a use
        self.bet = 0

    def win_bet(self):
        self.total += self.bet

    def lose_bet(self):
        self.total -= self.bet
```

تعریف توابع

بسیاری از مراحل تکراری خواهند بود. اینجاست که توابع به کار می‌آیند! مراحل زیر راهنمایی‌هایی هستند - توابع را بر اساس نیاز خود در برنامه خود اضافه یا حذف کنید.

گام ۶: نوشتن تابع برای شرط‌ها

از آنجا که از کاربر مقدار عددی می‌خواهیم، اینجا مکان خوبی است برای استفاده از try/except است. به خاطر داشته باشید که چک کنید که شرط بازیکن توسط تعداد تراشه‌های موجود تامین شود.

In [11]:

```
def take_bet(chips):

    while True:
        try:
            chips.bet = int(input("How many chips would like to bet? "))
        except ValueError:
            print("Sorry, a bet must be an integer!")
        else:
            if chips.bet > chips.total:
                print("Sorry, your bet can't exceed",chips.total)
            else:
                break
```

گام ۷: نوشتن تابع برای گرفتن کارت

هر یک از بازیکنان می‌توانند تا زمانی که تابع bust را بدهند، کارت بگیرند. این تابع هنگام بازی فراخوانی می‌شود هر زمانی که یک بازیکن درخواست کارت بدهد یا دست یک دیلر کمتر از 17 باشد. باید شیءهای دسته و دست را به عنوان آرگومان دریافت کند و یک کارت از دسته را برداشت کند و به دست اضافه کند. ممکن است بخواهید در صورتی که دست یک بازیکن بیش از 21 باشد، برای مشخص کردن آس‌ها بررسی کنید.

In [12]:

```
def hit(deck,hand):

    hand.add_card(deck.deal())
    hand.adjust_for_ace()
```

گام ۸: نوشتن تابع برای درخواست بازیکن برای گرفتن کارت یا ایستادن

این تابع باید دسته و دست بازیکن را به عنوان آرگومان دریافت کند و بازی را به عنوان یک متغیر عمومی تعیین کند.
اگر بازیکن کارت می‌گیرد (Hit)، از تابع hit() استفاده کنید. اگر بازیکن ایستاد (Stand)، متغیر playing را به False تنظیم کنید - این متغیر عملکرد یک حلقه while در بخشی دیگر از کد را کنترل خواهد کرد.

In [13]:

```
def hit_or_stand(deck,hand):
    global playing  # to control an upcoming while loop

    while True:
        x = input("Would you like to Hit or Stand? enter 'h' or 's'")

        if x[0].lower() == 'h':
            hit(deck, hand)
        elif x[0].lower() == 's':
            print("Player stands. Dealer is playing.")
            playing = False

        else:
            print("Sorry, Please try again!")
            continue
        break
```

گام ۹: نوشتن توابع برای نمایش کارت‌ها

هنگام شروع بازی و پس از هر باری که بازیکن یک کارت می‌گیرد، کارت اول دیلر مخفی است و تمام کارت‌های بازیکن قابل مشاهده است. در پایان دست، تمام کارت‌ها نمایش داده می‌شوند و ممکن است بخواهید مجموع ارزش هر دست را نشان

دهید. برای هر یک از این حالات، یک تابع بنویسید.

In [14]:

```
def show_some(player,dealer):
    print("\nDealer's Hand:")
    print(" <card hidden>")
    print('',dealer.cards[1])
    print("\nPlayer's Hand:", *player.cards, sep='\n ')

def show_all(player,dealer):
    print("\nDealer's Hand:", *dealer.cards, sep='\n ')
    print("Dealer's Hand =",dealer.value)
    print("\nPlayer's Hand:", *player.cards, sep='\n ')
    print("Player's Hand =",player.value)
```

گام ۱۰: نوشتن توابع برای مدیریت حالات پایانی بازی

به یاد داشته باشید که در صورت نیاز، دست بازیکن، دست دیلر و تراشه‌ها را به عنوان آرگومان‌ها منتقل کنید.

In [21]:

```
def player_busts(player,dealer,chips):
    print("Player busts!")
    chips.lose_bet()

def player_wins(player,dealer,chips):
    print("Player wins!")
    chips.win_bet()

def dealer_busts(player,dealer,chips):
    print("Dealer busts!")
    chips.win_bet()

def dealer_wins(player,dealer,chips):
    print("Dealer wins!")
    chips.lose_bet()

def push(player,dealer):
    print("Dealer and Player tie! It's a push.")
```

و حالا به بازی می‌پردازیم.!!

In [22]:

```
while True:
    # Print an opening statement
    print('Welcome to BlackJack! Get as close to 21 as you can without going over!\n
    Dealer hits until she reaches 17. Aces count as 1 or 11.')

    # Create & shuffle the deck, deal two cards to each player
    deck = Deck()
    deck.shuffle()

    player_hand = Hand()
    player_hand.add_card(deck.deal())
    player_hand.add_card(deck.deal())

    dealer_hand = Hand()
    dealer_hand.add_card(deck.deal())
    dealer_hand.add_card(deck.deal())

    # Set up the Player's chips
    player_chips = Chips()  # remember the default value is 100

    # Prompt the Player for their bet
    take_bet(player_chips)

    # Show cards (but keep one dealer card hidden)
    show_some(player_hand,dealer_hand)

    while playing:  # recall this variable from our hit_or_stand function

        # Prompt for Player to Hit or Stand
        hit_or_stand(deck,player_hand)

        # Show cards (but keep one dealer card hidden)
        show_some(player_hand,dealer_hand)

        # If player's hand exceeds 21, run player_busts() and break out of loop
        if player_hand.value > 21:
            player_busts(player_hand,dealer_hand,player_chips)
            break


    # If Player hasn't busted, play Dealer's hand until Dealer reaches 17
    if player_hand.value <= 21:

        while dealer_hand.value < 17:
            hit(deck,dealer_hand)

        # Show all cards
        show_all(player_hand,dealer_hand)

        # Run different winning scenarios
        if dealer_hand.value > 21:
            dealer_busts(player_hand,dealer_hand,player_chips)

        elif dealer_hand.value > player_hand.value:
            dealer_wins(player_hand,dealer_hand,player_chips)

        elif dealer_hand.value < player_hand.value:
            player_wins(player_hand,dealer_hand,player_chips)

```

```
        else:
            push(player_hand,dealer_hand)

    # Inform Player of their chips total
    print("\nPlayer's winnings stand at",player_chips.total)

    # Ask to play again
    new_game = input("Would you like to play another hand? Enter 'y' or 'n' ")

    if new_game[0].lower()=='y':
        playing=True
        continue
    else:
        print("Thanks for playing!")
        break
```

```
Welcome to BlackJack! Get as close to 21 as you can without going over!
    Dealer hits until she reaches 17. Aces count as 1 or 11.
How many chips would like to bet? 42

Dealer's Hand:
 <card hidden>
 Five of Clubs

Player's Hand:
 King of Clubs
 Ace of Hearts

Dealer's Hand:
 Ten of Hearts
 Five of Clubs
 Queen of Diamonds
Dealer's Hand = 25

Player's Hand:
 King of Clubs
 Ace of Hearts
Player's Hand = 21
Dealer busts!

Player's winnings stand at 142
Would you like to play another hand? Enter 'y' or 'n' y
Welcome to BlackJack! Get as close to 21 as you can without going over!
    Dealer hits until she reaches 17. Aces count as 1 or 11.
How many chips would like to bet? 40

Dealer's Hand:
 <card hidden>
 Three of Hearts

Player's Hand:
 Jack of Spades
 King of Spades
Would you like to Hit or Stand? enter 'h' or 's's
Player stands. Dealer is playing.

Dealer's Hand:
 <card hidden>
 Three of Hearts

Player's Hand:
 Jack of Spades
 King of Spades

Dealer's Hand:
 Queen of Hearts
 Three of Hearts
 Jack of Diamonds
Dealer's Hand = 23

Player's Hand:
 Jack of Spades
 King of Spades
Player's Hand = 20
Dealer busts!

Player's winnings stand at 140
```

```
Would you like to play another hand? Enter 'y' or 'n' n
Thanks for playing!
```

تمام شد! به یاد داشته باشید که این مراحل ممکن است با راه‌حل خودتان به طور قابل توجهی متفاوت باشد. این مهم نیست! تا زمانی که نتیجه مورد نظر را بدست نیاورید، روی بخش‌های مختلف برنامه‌ی خود کار کنید. این مسئله نیازمند زمان و صبر زیادی است! همیشه از پرسش‌ها و نظرات خود در انجمن‌های پرسش و پاسخ استفاده کنید.

خدا قوت!