Project 12 -- SEYI ABRAHAM

TA Help: John Smith, Alice Jones

- Help with figuring out how to write a function.

Collaboration: Friend1, Friend2

- Helped figuring out how to load the dataset.
- Helped debug error with my plot.

Question 1

In [1]:
```
# A class can be considered a outline made by the user for creating objects
# An object is an instance of the class with actual values.

# Objects consist of:

# State: attributes/properties of an object

# Behavior: the methods of the object and also its response to other objects

# Identity: gives a unique name to the object and that allows for interaction between
```

In [2]:
```
# How to declare an object from the class
# We can't really do anything with that class,
# we can't look up the attributes, run the functions or methods inside that class
# unless we make an object of that class
```

In [1]:
```
class CAR:
    #attibute
    attr1 = "red"
    attr2 = "fast"
    coolfeature = "Apple CarPlay"
    year = "2022"

    #sample method
    #self refers to the class of which the method belongs
    def fun(self):
        print("The car is", self.attr1)
        print("The car is", self.attr2)
```

In [4]:
```
# Let us make a specific object of the class car and call it Miata
```

In [2]:
```
# object instantiaton

Miata = CAR()
```

In [3]:
```
# this display that Miata is an object of that class CAR

print(Miata)
```

```
<__main__.CAR object at 0x7fc5941277f0>
```

In [4]:
```
# We can run the function side and see what it does
# it print out what differnt attributes are

Miata.fun()
```

```
The car is red
The car is fast
```

In [5]:
```
Miata.attr1
```

Out[5]:
```
'red'
```

In [6]:
```
Miata.attr2
```

Out[6]:
```
'fast'
```

In [7]:
```
Miata.coolfeature
```

Out[7]:
```
'Apple CarPlay'
```

In [8]:
```
Miata.year
```

Out[8]:
```
'2022'
```

In [9]:
```
class KITCHEN:
    # attribute
    attr1="neat"
    attr2="lot of pantries"
    attr3="good illumination"
    available="cooking gas"
    year="2023"

    def fun(self):
        print("The kitchen is",self.attr1)
        print("The kitchen has",self.attr2)
        print("The kitchen has",self.attr3)
```

In [10]:
```
COOK = KITCHEN()
```

In [11]:
```
print(COOK)
```

```
<__main__.KITCHEN object at 0x7fc59413b910>
```

In [12]:
```
COOK.fun()
```

```
The kitchen is neat
The kitchen has lot of pantries
The kitchen has good illumination
```

In [13]:
```
COOK.attr1
```

Out[13]:
```
'neat'
```

In [14]:
```
COOK.attr2
```

Out[14]:
```
'lot of pantries'
```

In [15]:
```
COOK.attr3
```

Out[15]:
```
'good illumination'
```

In [16]:
```
COOK.available
```

Out[16]:
```
'cooking gas'
```

In [17]:
```
COOK.year
```

Out[17]:
```
'2023'
```

What happens when you type: print(Miata)

What happens when you type: Miata.fun()

Now create and declare your own object. We encourage you to make a class of your own and try it out, e.g., make a class for a BOOK or a KITCHEN or a UNIVERSITY or a DOG.

Question 2

In [21]:
```
# Python function inside a class is called a method
```

In [18]:
```
class Card:
#mapping each possible card number(2-10,J,Q,K,A) to a numerical value
    _value_dict = {"2": 2, "3": 3, "4": 4, "5": 5, "6": 6, "7": 7, "8":8, "9":9, "10"
    def __init__(self, number, suit):
        if str(number).lower() not in [str(num) for num in range(2, 11)] + list("jqka"
            raise Exception("Number wasn't 2-10 or J, Q, K, or A.")
        else:
            self.number = str(number).lower()
        if suit.lower() not in ["clubs", "hearts", "diamonds", "spades"]:
            raise Exception("Suit wasn't one of: clubs, hearts, spades, or diamonds.")
        else:
            self.suit = suit.lower()

    def __str__(self):
        return(f'{self.number} of {self.suit.lower()}')

    def __repr__(self):
        return(f'Card(str({self.number}), "{self.suit}")')

    def __eq__(self, other):
        if self.number == other.number:
            return True
        else:
            return False

    def __lt__(self, other):
        if self._value_dict[self.number] < self._value_dict[other.number]:
            return True
        else:
```

```
            return False

    def __gt__(self, other):
        if self._value_dict[self.number] > self._value_dict[other.number]:
            return True
        else:
            return False

    def __hash__(self):
        return hash(self.number)
```

In [19]:
```
class Deck:
    brand = "Bicycle"
    _suits = ["clubs", "hearts", "diamonds", "spades"]
    _numbers = [str(num) for num in range(2, 11)] + list("jqka")

    def __init__(self):
        self.cards = [Card(number, suit) for suit in self._suits for number in self._r

    def __len__(self):
        return len(self.cards)

    def __getitem__(self, key):
        return self.cards[key]

    def __setitem__(self, key, value):
        self.cards[key] = value
```

In [20]:
```
my_card = Card("10", "spades")
```

In [21]:
```
# It displays a card with a string 10 and then a spades when object is printed by itse

my_card
```

Out[21]:
```
Card(str(10), "spades")
```

In [22]:
```
# This code just give the output as 10 of spades

print(my_card)
```

```
10 of spades
```

In [23]:
```
my_deck = Deck()
```

In [24]:
```
len(my_deck)
```

Out[24]:
```
52
```

Create an instance of the class Card and call it my_card. Then run: print(my_card) and also run: my_card

What is the difference in the output?

Create an instance of the class Deck and call it my_deck. Now what is the number of items you will find in the object my_deck

Question 3

In [25]:
```
class Deck:
    brand = "Bicycle"
    _suits = ["clubs", "hearts", "diamonds", "spades"]
    _numbers = [str(num) for num in range(2, 11)] + list("jqka")

    def __init__(self):
        self.cards = [Card(number, suit) for suit in self._suits for number in self._r

    def __str__(self):
        return(f' A {self.brand} deck with {len(self)} card.')


    def __len__(self):
        return len(self.cards)

    def __getitem__(self, key):
        return self.cards[key]

    def __setitem__(self, key, value):
        self.cards[key] = value
```

In [26]:
```
my_deck1 = Deck()
```

In [27]:
```
print(my_deck1)
```

```
 A Bicycle deck with 52 card.
```

Modify the Class Deck to return a string that says "a bicycle deck with 52 cards".

Question 4

In [32]:
```
# we want to make a Player class and then make an object of
# type player that can start drawing some cards out of a deck.
# so our class called player needs to have something like:
# A deck to draw the card from
# A hand of card of its own
# A name for the player
# A method by which it can draw some cards from the deck
```

In [33]:
```
# The class will be called a player
# We will initiallize a player
# We want it name to be whatever we pass in
# And we want to have a hand that start out empty
# And we want to have a deck that is whatever deck we passed in as well
# Also we need to have a method by which we can draw some cards
# So all i need to do is tell it which player we are using to draw the cards
# Then we go off of that players' deck and get the last card
# -1 means getting the very last card
# Then off of that deck, we will pop a card
# Because we have stored the last one, then we will pop it off, so that it is not on t
# Then append the card to our hand
# Then we will return that card
```

In [28]:
```
class Player:

    def __init__(self, name, deck):
        self.name = name
        self.hand = []
        self.deck = deck

    def draw(self):
        card = self.deck.cards[-1]
        self.deck.cards.pop()
        self.hand.append(card)
        return card
```

Let's create a new class called Player We will use this to represent a player in a game. The following features must be included:

A deck to draw from

A hand of cards

The name of the player

A draw method that draws a card from the deck and adds it to the hand.

Question 5

In [29]:
```
my_deck1 = Deck()
```

In [30]:
```
len(my_deck1)
```

Out[30]:
```
52
```

In [31]:
```
player1 = Player("Liz", my_deck1)
```

In [32]:
```
card = player1.draw()
```

In [33]:
```
print(card)
```

```
a of spades
```

In [34]:
```
# a card has been taken off

len(my_deck1)
```

Out[34]:
```
51
```

In [35]:
```
anothercard=player1.draw()
```

In [36]:
```
print(card)
```

```
a of spades
```

In [37]:
```
len(my_deck1)
```

Out[37]:
```
50
```

In [38]:
```
anothercard1=player1.draw()
```

In [39]:
```
print(card)
```

```
a of spades
```

In [40]:
```
len(my_deck1)
```

Out[40]:
```
49
```

In [ ]:
```
# and we have not shuffle our deck
# if Liz keep going there will be reduction in the number
```

What card does Liz draw? Create a Deck and a Player, and draw a card from the deck. Print the value on the card that is drawn.

Pledge

By submitting this work I hereby pledge that this is my own, personal work. I've acknowledged in the designated place at the top of this file all sources that I used to complete said work, including but not limited to: online resources, books, and electronic communications. I've noted all collaboration with fellow students and/or TA's. I did not copy or plagiarize another's work.

> As a Boilermaker pursuing academic excellence, I pledge to be honest and true in all that I do. Accountable together – We are Purdue.