Homework #4 - Make a two-player card game

For this assignment, you will be writing constructors and methods to create a two-player card game.

The three classes are *Card, Deck, Player*.

The *Card* class can be used to create a card object. For example, a “King of Spades”. The constructor and the methods for the Card class are already provided

The *Deck* class can be used to create a standard deck of 52 cards. The constructor and methods have been provided.

The *Player* class is used to create players. You will be writing the constructor and all of the methods for the *Player* class.

NOTE :
We have provided five unit tests (lines 96-117). If these give you an error, that means that you still need to write code for those functions or fix bugs.

You should not edit the unit tests. If these fail after you have written the code then it is because of bugs. Fixing the bugs will *pass* the tests. You can comment out unit tests while you are working on them, but be sure to uncomment them before you submit.

All four tests will say “Ok” if your code is complete and there are no errors.

Player Class

Write the following:

- Create the constructor: It will initialize a player object
    - A player object should have 3 instance variables: name (str); hand (list); score (int)
    - It takes 3 inputs - card deck, the name of the player, and the number of cards to be dealt. By default, a player should be given 5 cards.
    - It also sets the player's score to zero.
    - It should draw the number of cards specified in the input from a shuffled deck and append the cards to a list called hand.

- Create the __str__ method: It returns the player's name, number of cards remaining in the player's hand and the current score. For example, "Leo : 3 cards in hand. 0 points"

```
Leo : 4 cards in hand. 1 points.
```

- Create the play_card method
    - It should play the last card in the player's hand

- Create the update_score method
    - Update the player's score based on how their card compared to the other player's card
    - It should take an integer as an input (-1, 0, +1) and add it to the player's score

- Create the final_score method
    - It takes as input another Player object
    - It compares the scores of both players
    - It returns the name of the player who won. For example, "Leo won by 2 points"
    - If both players have equal scores, it returns "No winners!"

```
Leo won by 6 points!
```

- Create the exchange_card method
    - Replace the last card in the player's hand with a card drawn from the deck

---

Main Method

Complete the main function to:

- create the two player objects
- In the loop:
    - play the players' cards one after the other
    - print out the player's name and what card they played
    - compare and update scores
    - print out each player's information
    - before the last cards are played both players need to exchange the final card in their hand with one from the deck

---

Example Output From HW4

NOTE: Your output will not look *exactly* like this because we are using *random* and can't predict what it will return.

```
Leo played Jack of Hearts
John played 3 of Clubs
Leo : 6 cards in hand. 1 points.
John : 6 cards in hand. -1 points.


Leo played Jack of Clubs
John played 8 of Clubs
Leo : 5 cards in hand. 2 points.
John : 5 cards in hand. -2 points.


Leo played 7 of Hearts
John played Queen of Spades
Leo : 4 cards in hand. 1 points.
John : 4 cards in hand. -1 points.


Leo played 8 of Hearts
John played 8 of Spades
Leo : 3 cards in hand. 2 points.
John : 3 cards in hand. -2 points.


Leo played 5 of Clubs
John played 7 of Clubs
Leo : 2 cards in hand. 1 points.
John : 2 cards in hand. -1 points.


Leo played 10 of Diamonds
John played 9 of Hearts
Leo : 1 cards in hand. 2 points.
John : 1 cards in hand. -2 points.


Leo played King of Clubs
John played 10 of Clubs
Leo : 0 cards in hand. 3 points.
John : 0 cards in hand. -3 points.


Leo won by 6 points!
```

Grading Rubric - total of 60 points

10 points - the __init__ method in the Player class – Initializes all the instance variables, and deals a hand to the player
10 points - the *__str__* method in the Player class – Returns the player's name, number of cards remaining in the player's hand and the current score
5 points - the *play_card* method in the Player class – Plays a card from the player's hand (i.e. removes the last card from the player's hand and returns it).
5 points - the *update_score* method in the Player class – Updates the player's score
10 points - the *final_score* method in the Player class – Compares the score of the player with another player and returns the winner's name and the margin they won by (Ex. "Leo won by 6 points!"). If there is a tie, returns "No winners!"
5 points - the *exchange_card* method in the Player class – Exchanges the last card in the player's hand with a random card from the deck.
10 points - Complete the main function to 1) create the two player objects, then in the loop 1) play the players' cards in alternation, 2) print out the player's name and what card they played, 3) compare and update scores, and 4) print out each player's name, number of cards remaining, and current score.
5 points - Before the last hand, each player should exchange a card in the player's hand with a card from the deck.

This grading rubric shows how you will gain points, but not all the ways you could lose points.

Git Commits - 15 points

Make at least 3 git commits before the deadline. Each commit is worth 5 points. Please upload a link to your github repository URL to canvas.

Extra Credit - 3 points

Update the *compare_card* method in the Card class to consider the card suit when two cards have the same rank:

Suits are in this order : Clubs < Spades < Hearts < Diamonds

Therefore, a "2 of Clubs" is lower than a "2 of Spades" but a "9 of Hearts" is higher than a "8 of Diamonds"

Extra Credit Example Output:

Leo has 4 cards in hand and 1 point, John has 4 cards in hand -1 point.

Next time when they both played, their cards had the same rank. But Leo played a Hearts card, which is higher than John's Spades card. So he gets another point, and John loses another point.

```
Leo played 7 of Hearts
John played Queen of Spades
Leo : 4 cards in hand. 1 points.
John : 4 cards in hand. -1 points.


Leo played 8 of Hearts
John played 8 of Spades
Leo : 3 cards in hand. 2 points.
John : 3 cards in hand. -2 points.
```