CS 1337 - Spring 2019

Final Project - Blackjack

Points: 400

Develop a python program for a card game called Blackjack. By the time it's finished, this program will let the users play Blackjack.

Note:

This is a group project. You should form teams of two or three. Create a contributions table using LibreOffice (team.odt) listing all group members' names and the percentage contributed to this effort. It should be obvious that these percentages should add up to 100%. Please note that if you are a slacker, your team is obligated to reveal your lack of contribution.

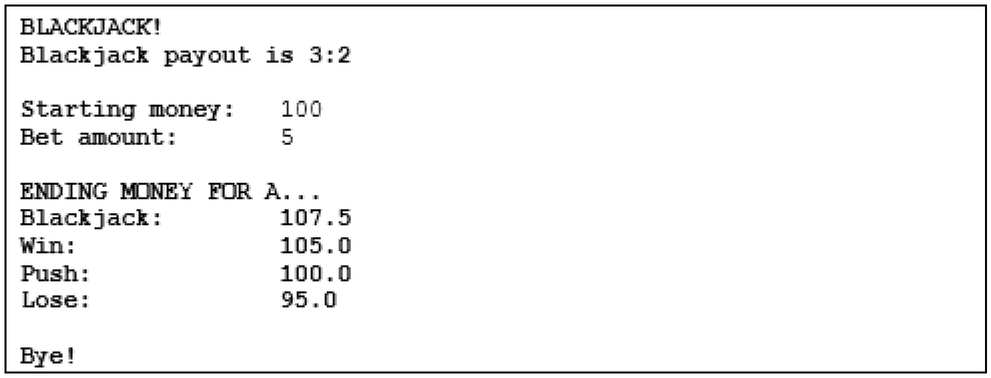
There are different levels to this project and you build one top of the similar to your lab exercises. You only get to upload the files that belong to the highest level you have reached. Each level describes the specifications along with the files to be uploaded at that level. You should upload your team.odt file along with your python files.

Level 1 (40 points)

Calculate the player's money

Create a program that calculates the amount of money that a player will have if the player gets a Blackjack, wins, loses, or pushes (ties).

Console



```
BLACKJACK!
Blackjack payout is 3:2

Starting money:    100
Bet amount:        5

ENDING MONEY FOR A...
Blackjack:         107.5
Win:               105.0
Push:              100.0
Lose:              95.0

Bye!
```

Specifications

- The program should accept integer or float entries.
- Assume the user will enter valid data.
- Getting a Blackjack pays out 3:2, which is 1.5 times the bet amount.
- The program should round the Blackjack payout to a maximum of two decimal place

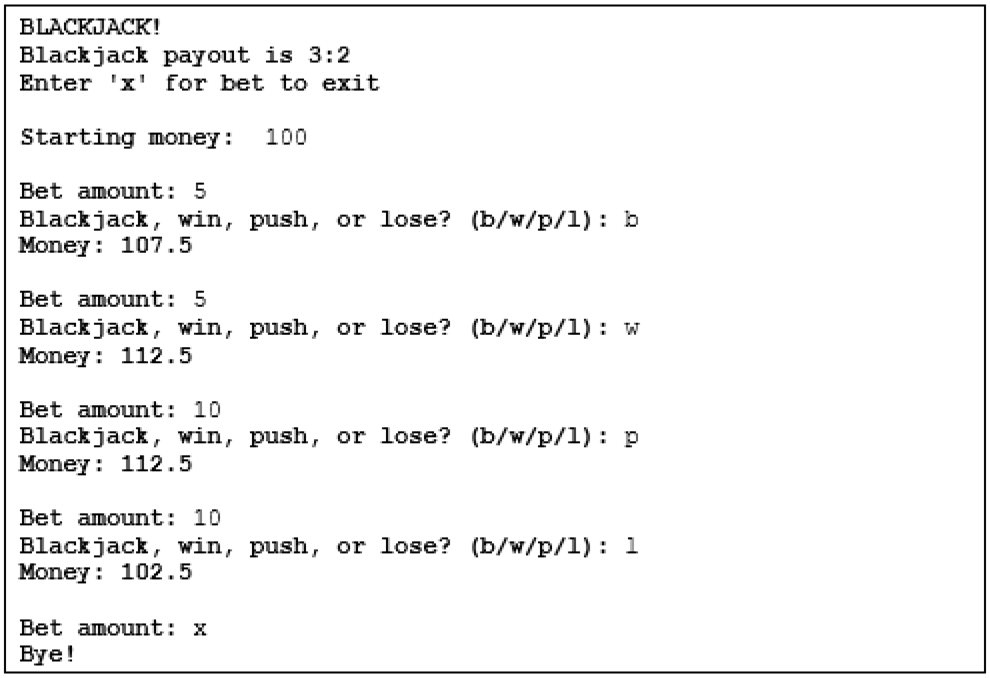
Files to upload: level1_blackjack.py

Level 2 (50 points)

Track the player's money

Update the program so that it keeps track of the money until the user exits the program.

Console



```
BLACKJACK!
Blackjack payout is 3:2
Enter 'x' for bet to exit

Starting money:  100

Bet amount: 5
Blackjack, win, push, or lose? (b/w/p/l): b
Money: 107.5

Bet amount: 5
Blackjack, win, push, or lose? (b/w/p/l): w
Money: 112.5

Bet amount: 10
Blackjack, win, push, or lose? (b/w/p/l): p
Money: 112.5

Bet amount: 10
Blackjack, win, push, or lose? (b/w/p/l): l
Money: 102.5

Bet amount: x
Bye!
```

Specifications

- Assume the user will enter valid data.

Files to upload: level2_blackjack.py

Level 3 (90 points)

Use functions to organize the program

Create a program that provides the same functionality as chapter 3, but use functions to organize the code for the program so it’s easier to reuse and maintain.

Console

```
BLACKJACK!
Blackjack payout is 3:2
Enter 'x' for bet to exit

Starting money:   100

Bet amount: 5
You won.
Money: 105.0

Bet amount: 5
You lost.
Money: 100.0

Bet amount: 10
You lost.
Money: 90.0

Bet amount: 10
You lost.
Money: 80.0

Bet amount: 20
You pushed.
Money: 80.0

Bet amount: 20
You got a blackjack!
Money: 110.0

Bet amount: x
Bye!
```

Specifications

- Assume the user will enter valid data.
- Use the random module to determine whether the player gets a Blackjack, wins, pushes, or loses using the following odds:

Blackjack	5%
Win	37%
Push	9%
Loss	49%

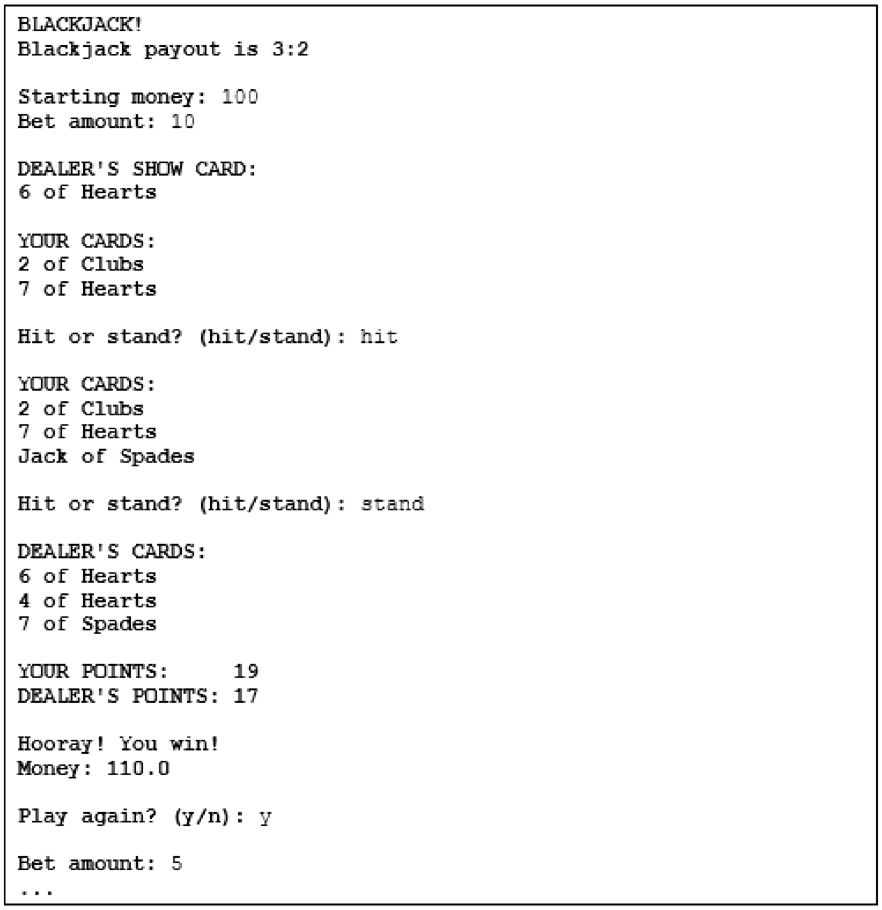
Files to upload: level3_blackjack.py

Level 4 (100 points)

Use lists to store cards

Modify the program so it allows the player to play Blackjack against a computer dealer. Then, the program can update the player's money depending on the result of each round.

Console



```
BLACKJACK!
Blackjack payout is 3:2

Starting money: 100
Bet amount: 10

DEALER'S SHOW CARD:
6 of Hearts

YOUR CARDS:
2 of Clubs
7 of Hearts

Hit or stand? (hit/stand): hit

YOUR CARDS:
2 of Clubs
7 of Hearts
Jack of Spades

Hit or stand? (hit/stand): stand

DEALER'S CARDS:
6 of Hearts
4 of Hearts
7 of Spades

YOUR POINTS:     19
DEALER'S POINTS: 17

Hooray! You win!
Money: 110.0

Play again? (y/n): y

Bet amount: 5
...
```

Specifications

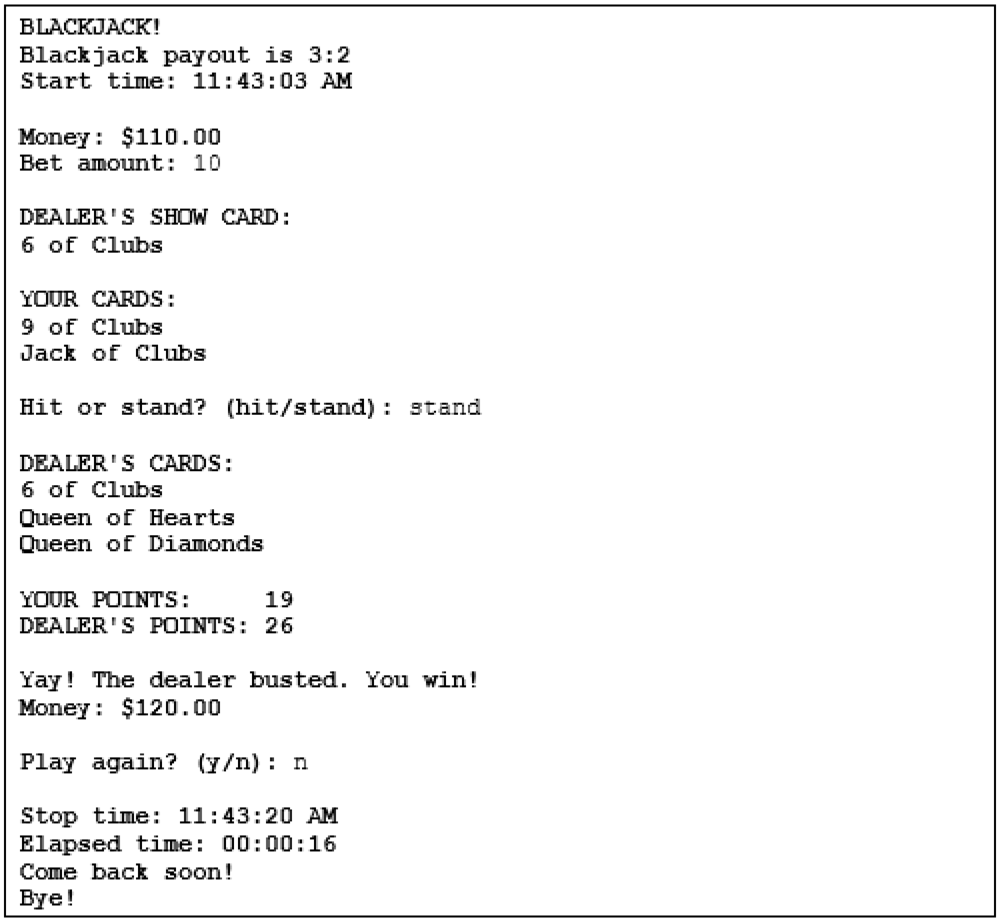
- Use a list to store the suit, rank, and point value for each card.
- Use a list of lists to store the cards in the deck. You can use two nested loops to create the deck of cards. Use a list of lists to store the dealer's hand and the player's hand.
- If necessary, learn the rules of Blackjack by researching it on the web.
- Use a standard 52-card deck of playing cards.
- The dealer must continue taking cards until the dealer has at least 17 points.
- Don't allow a player to "split" a hand or "double down."

Files to upload: level4_blackjack.py, cards.py (module that contains reusable code for get_deck(), shuffle(deck), deal_card(deck), get_empty_hand(), add_card(hand, card), get_points(hand)

Level 5 (120 points)

Use an object-oriented approach

Convert the program from procedural to object-oriented. This shouldn't change the functionality of the code, but it should make the code more modular, reusable, and easier to maintain.



```
BLACKJACK!
Blackjack payout is 3:2
Start time: 11:43:03 AM

Money: $110.00
Bet amount: 10

DEALER'S SHOW CARD:
6 of Clubs

YOUR CARDS:
9 of Clubs
Jack of Clubs

Hit or stand? (hit/stand): stand

DEALER'S CARDS:
6 of Clubs
Queen of Hearts
Queen of Diamonds

YOUR POINTS:     19
DEALER'S POINTS: 26

Yay! The dealer busted. You win!
Money: $120.00

Play again? (y/n): n

Stop time: 11:43:20 AM
Elapsed time: 00:00:16
Come back soon!
Bye!
```

Specifications

- Use a Card class that provides attributes for storing the rank, suit, and points for a card. This class should also provide a method that returns a string that includes the rank and suit of the card.
- Use a Deck class that provides for a standard 52-card playing deck. This class should include methods that allow you to shuffle the deck and to deal cards from it.
- Use a Hand class to store the dealer's hand and the player's hand. This class should include methods that allow you to add a card, get a card, count the number of cards, and get the total points for a hand of Blackjack.

Files to upload: level5_blackjack.py, objects.py (define classes Card, Deck, Hand in the objects file)