CMPT120: Introduction to Computing Science and Programming I

Instructor: Hassan Khosravi

Summer 2012

Assignment 3

Due: July 30th

This assignment is to be done individually.

The university policy on academic dishonesty (cheating) will be taken very seriously in this course. You may not discuss the specific questions nor their solutions with any other student. You are encouraged to discuss the general concepts involved in the questions. Please take advantage of office hours offered by the instructor and the TAs if you are having difficulties

---

For this assignment, your job is to create a program that plays (a simplified version of) blackjack. Name your program blackjack.py.

The Rules

For simplicity, we won't be implementing the full set of blackjack rules, but will get the core of the game working:

1. A regular deck of 52 cards is used. The cards are every combination of 13 "ranks" (two, three, four, five, six, seven, eight, nine, ten, jack, queen, king, ace) and 4 "suits" (spades, clubs, diamonds, hearts). The deck is shuffled (put into a random order) before the game begins.
2. In our game, each player starts with $1000. At the start of the game, ask how many people wish to play.
3. Before each turn, each player makes a bet. They can risk any amount of their money on each turn.
4. Two cards are dealt (given) to each player and placed face-up on the table. The dealer also receives two cards, but one is kept face-down (the "hole card"). The rest of the cards form the "shoe".
5. The object of the game is to get cards that have value as close as possible to 21 without going over ("busting"). The cards in a players hand have point values as follows.

Card Rank	Value
2–10	the card's face value (2–10)
J, Q, K	10
A	1 or 11, whichever is better

6. Each player gets a turn to take additional cards from the shoe. A player can take another card ("hit") as many times as they like. When they stop taking new cards ("stand"), their turn is over. The cards in the player's hand will not change after their turn is over.
7. Once the players have taken their turns, the dealer reveals his/her hole card. The dealer also has the chance to hit and stand, but they have a more rigid rule to follow. The dealer must hit (take another card) if their hand contains 16 points or less, and stand (stop taking cards) if they have 17 points or more.
8. Once all of the cards have been played, each player either wins or loses depending on what is in their hand and the dealer's hand:

		Player's outcomes	
		Busts (> 21 points)	**Doesn't bust (≤ 21 points)**
Dealer's outcomes	**Busts**	player loses	player wins
	points < player	player loses	player wins
	points = player	player loses	push
	points > player	player loses	player loses

If the player wins, they get their bet back plus an equal amount from the dealer. If they lose, they lose the amount of money they bet.

9. A "push" is a tie. In that case, the player takes his/her money back and doesn't win or lose anything. [In true blackjack, there is an additional set of outcomes that occur when either the player or dealer has "blackjack": 21 points with two cards. We won't worry about those rules here.]
10. After the turn is played and the bets are settled, they play continues with another round (go back to step 3).
11. In a true game of blackjack, players can join or leave the game between hands. We won't worry about that. For this game, all of the players will play *three hands*, and then the game will stop.

Implementing and suggested plan

To help you along, a cards module and its documentation has been provided. You'll have to download the cards.pyc file. Save this file in the same directory as your blackjack.py file and you should be able to import cards.

This is a suggested list of steps that you can follow to get your program working. Remember to test your code frequently, at least after each of these steps.

The hints don't say much about breaking your code into functions. Deciding how to do that is part of the assignment.

1. Import cards. Create a deck of cards using cards.deck().
2. Use cards.hand_display() to print out the contents of the deck to check it. Add code to shuffle and test again.
3. Get the number of players in the game.
4. Create a loop for the turns (you want to loop three times). In it, call a function that plays a full round (or will eventually, at least).
5. For each round, create an empty list for the hands. For each player, create their hand (a list of card objects) and append it to the list of hands. Use hand_display to print out the contents of each hand to test. Do the same for the dealer. (Don't worry about hiding the dealer's hole card yet.)
6. Create a function to count the total number of points in a hand (list of cards). Use it in the loop to calculate the total for each initial hand.
7. Add code to let a player hit or stand. Check for the player busting. Re-test your point totaling function to make sure it handles hands with aces that should count as 1 properly. (See hints below.)
8. Add code to play the dealer's hand: keep hitting until the total is ≥ 17.
9. Add code to determine and print the result for each player (win/loss/push). You will add the betting/money code later: for now, just indicate the result of the hand.
10. Create your own version of hand_display that displays ** instead of the first card. Use that instead of hand_display when initially displaying the dealer's hand. This will take care of hiding the hole card.
11. Work back through you program and add code for the player's banks (starting at $1000 each). This should probably be a list of integers: one for each player.
12. Add code that asks for each player's bet before each turn.
13. In your win/loss/push code, add logic to update each player's bank.
14. Make sure you have done the error checking on all of the input

Functions and Pseudocode

Prepare a detailed plan for the assignment. This plan will be submitted.

The following are outlines of logic for this assignment.

The game

Code corresponding to this logic would likely form the main part of your program. It should be broken up into appropriate functions, but at a high level, your program should do this:

1. Create and shuffle the deck.
2. Ask how many players there are and put $1000 in each player's “bank”.
3. For each round you're going to play:
    1. Ask each player for their bet.
    2. Give two cards to each player and the dealer (display all but the dealer's hole card).
    3. Let each player take their turn:
        1. Ask the player whether they want to hit or stand, until they either stand or bust.
        2. Each card the player is given should be added to their hand, and their new total calculated appropriately.
    4. Play the dealer's hand. (Hit on 16, stand on 17.)
    5. Settle the bets: give/take the bet for each player.
4. Game over.

Counting a Hand’s Value

This is the general idea for determining the number of points in a hand:

1. For each card in the hand:
    1. Add the card's value to the total.
    2. Count aces as 11, but keep track of the number of aces you've seen.
2. If the total is over 21, but there were aces in the hand, subtract 10. Repeat as necessary.

Notes

- The random module has a function shuffle that will shuffle the deck for you.
- Lists have a pop() method that removes an element from the end of the list and returns it. This would be handy for removing a card from the deck and adding it to a player's hand.

```
c = deck.pop()
hand.append(c)
```

- Remember that every time a player takes a card from the deck, it is *removed* from the deck.
- An ace can count for either 11 or 1 in a hand. When calculating the value of a hand, make sure you use the best one.
- The user should be able to enter their choice in lower- or uppercase. So, entering `s` and `S` should do the same thing. Hint: convert their input to uppercase immediately with the string's upper method.
- You won't want to play the full game all of the time while testing. Press control-C to stop the program.
- Break up the program into subtasks placed into functions as appropriate. You should do that whenever you think it's useful.
- Your code should be easy to read and understand. Style will be marked more closely on this assignment.

Sample turns

This is the first round of the game. Notice that Player 1 has an initial total of 16, with the ace counting for 11 points. When he picks up another ace, a total of 27 would be a bust; it counts for 1 point, for a total of 17 points.

When Player 1 draws an eight, their total would be 25. Since there is still another ace counting 11 points (they have a “soft hand”), that ace can count for 1 and their total becomes 15. Player 1 decides he has pushed his luck far enough and stands.

Player 2 makes some input errors, but eventually hits on 16 to get a total of 20. After another typo, she stands. The dealer ends up with 19 points, so player 1 loses his $100 bet. Player 2 wins hers. A this point, they should have $900 and $1100, respectively.

```
How many players are there? 2

Player #1:
You have $1000.
How much do you want to bet? 100
Player #2:
You have $1000.
How much do you want to bet? 100

Hands:
  Dealer:     ** 7C
  Player #1: AH 5S
  Player #2: 6D QD

Player #1
Cards: AH 5S, total: 16
Hit (h) or stand (s)? h
Cards: AH 5S AC, total: 17
Hit (h) or stand (s)? h
Cards: AH 5S AC 8S, total: 15
Hit (h) or stand (s)? s

Player #2
Cards: 6D QD, total: 16
Hit (h) or stand (s)? A
Please enter "h" or "s".
Cards: 6D QD, total: 16
Hit (h) or stand (s)? HIT
Please enter "h" or "s".
Cards: 6D QD, total: 16
Hit (h) or stand (s)? H
Cards: 6D QD 4D, total: 20
Hit (h) or stand (s)? stand
Please enter "h" or "s".
Cards: 6D QD 4D, total: 20
Hit (h) or stand (s)? S

Dealer draws 2D.
Dealer draws 3S.
Dealer draws 4H.
Dealer's hand: 3C 7C 2D 3S 4H, total: 19
Player #1 loses $100.
Player #2 wins $100.
Press enter to continue.
```

In the next example, Player 3 has some initial betting problems.

Once that's done, Player 1 gets very lucky with a soft hand and makes 21 before standing. Player 2 wisely stands with the initial 20. Player 3 busts with 22 points.

When the dealer gets 20 as well, it means that we have all three possible outcomes of the hand: Player 1 wins, Player 2 pushes, Player 3 loses. Their totals should now be $1200, $1000, and $700.

```
How many players are there? 3

Player #1:
You have $1000.
How much do you want to bet? 200
Player #2:
You have $1000.
How much do you want to bet? 100
Player #3:
You have $1000.
How much do you want to bet? 0
You have $1000.
How much do you want to bet? 1234
You have $1000.
How much do you want to bet? -123
You have $1000.
How much do you want to bet? 300

Hands:
  Dealer:    ** 6H
  Player #1: AD AS
  Player #2: QC TC
  Player #3: 5D JD

Player #1
Cards: AD AS, total: 12
Hit (h) or stand (s)? h
Cards: AD AS 6S, total: 18
Hit (h) or stand (s)? h
Cards: AD AS 6S 3S, total: 21
Hit (h) or stand (s)? s

Player #2
Cards: QC TC, total: 20
Hit (h) or stand (s)? s

Player #3
Cards: 5D JD, total: 15
Hit (h) or stand (s)? h
Cards: 5D JD 7H, total: 22
Bust!

Dealer draws 6C.
Dealer's hand: 8S 6H 6C, total: 20
Player #1 wins $200.
Player #2 pushes.
Player #3 loses $200.
Press enter to continue.
```

The next three pages are an example of a full run

```
How many players are there? 3

Player #1:
You have $1000.
How much do you want to bet? 100
Player #2:
You have $1000.
How much do you want to bet? 200
Player #3:
You have $1000.
How much do you want to bet? 300

Hands:
  Dealer:    ** 3D
  Player #1: 7C 5D
  Player #2: 6D 5S
  Player #3: 2S KD

Player #1
Cards: 7C 5D, total: 12
Hit (h) or stand (s)? h
Cards: 7C 5D KS, total: 22
Bust!

Player #2
Cards: 6D 5S, total: 11
Hit (h) or stand (s)? h
Cards: 6D 5S QS, total: 21
Hit (h) or stand (s)? s

Player #3
Cards: 2S KD, total: 12
Hit (h) or stand (s)? h
Cards: 2S KD 4H, total: 16
Hit (h) or stand (s)? s

Dealer draws 8C.
Dealer draws 9D.
Dealer's hand: AS 3D 8C 9D, total: 21
Player #1 loses $100.
Player #2 pushes.
Player #3 loses $300.
Press enter to continue.
```

```
Player #1:
You have $900.
How much do you want to bet? 100
Player #2:
You have $1000.
How much do you want to bet? 200
Player #3:
You have $700.
How much do you want to bet? 300

Hands:
  Dealer:    ** KC
  Player #1: 7S JC
  Player #2: TH 6C
  Player #3: 7H TC

Player #1
Cards: 7S JC, total: 17
Hit (h) or stand (s)? s

Player #2
Cards: TH 6C, total: 16
Hit (h) or stand (s)? h
Cards: TH 6C 3S, total: 19
Hit (h) or stand (s)? s

Player #3
Cards: 7H TC, total: 17
Hit (h) or stand (s)? h
Cards: 7H TC JD, total: 27
Bust!

Dealer's hand: QH KC, total: 20
Player #1 loses $100.
Player #2 loses $200.
Player #3 loses $300.
Press enter to continue.
```

```
Player #1:
You have $800.
How much do you want to bet? 200
Player #2:
You have $800.
How much do you want to bet? 400
Player #3:
You have $400.
How much do you want to bet? 600
You have $400.
How much do you want to bet? 399

Hands:
  Dealer:    ** TS
  Player #1: 8H JH
  Player #2: 6H 4D
  Player #3: 9C 4C

Player #1
Cards: 8H JH, total: 18
Hit (h) or stand (s)? s

Player #2
Cards: 6H 4D, total: 10
Hit (h) or stand (s)? h
Cards: 6H 4D 3H, total: 13
Hit (h) or stand (s)? h
Cards: 6H 4D 3H 9S, total: 22
Bust!

Player #3
Cards: 9C 4C, total: 13
Hit (h) or stand (s)? h
Cards: 9C 4C 7D, total: 20
Hit (h) or stand (s)? s

Dealer draws KH.
Dealer's hand: 2H TS KH, total: 22
Player #1 wins $200.
Player #2 loses $400.
Player #3 wins $399.
Press enter to continue.

Player #1: $1000
Player #2: $400
Player #3: $799
```

Submitting Your Assignment

You must create your assignment in electronic form and submit it online.

Put your implementation (blackjack.py), the card module (cards.pyc) and your plan (could be a doc, pdf or jpg file) in a zip file. Plan the whole program before starting your implementation. It would help you greatly. Make sure your plan is fully and easily understandable for the TA. There are some instructions in the website if you have trouble with zipping your files. Call the zip file assignment3 (so it would be assignment3.rar or assignment3.zip).

Login to your account in https://courses.cs.sfu.ca/. You should be able to upload your file when you go into assignment3.