CIS 41A - Lab assignment 2: if statements, loops, functions, programming logic

Write a program that plays a simple version of the blackjack or twenty-one card game.

**How the game is played**

* The blackjack game is played between the computer and the user, using standard playing cards with values A (Ace), 2-10, J (Jack), Q (Queen), K (King).
* The dealer is the computer and has multiple decks of cards to deal out. (This means there can be more than 4 aces, for example. So no card counting when you play!)
* At the start of the game, each player gets 2 random cards.
* The dealer or computer shows (prints) 1 of its 2 cards, and it shows both of the user's cards.
* The user adds up the total values of all the user's cards.  
  For this simplified version, A, J, Q, K all get a value of 10 each, and the numbered cards' values are the numbers themselves.
* Then the dealer keeps asking the user if they want another card, and gives one card to the user each time the user says yes. The object of the game is for the user to stop getting more cards before their total goes above 21, but try to get as close to 21 as possible.
* When the user stops getting cards, then the dealer starts getting more cards until the dealer's total is *17* or above.
* At any time, if the dealer's or user’s total goes above 21, then that person automatically loses.   
  Similarly, at any time if the dealer's or user’s total is exactly 21, then that person automatically wins.  
  If both players' totals are below 21 when both players stop getting cards, then the player with the higher total wins.   
  If the 2 players tie, then the dealer loses and the user wins.

**Requirements**

* The program must have the 5 functions as described below:
* A **getACard** function that returns one random playing card:
  1. Use the randint method in the random module to create a random number between 1 and 13. The number represents one of the playing cards, in order from 1-13: A, 2-10, J, Q, K
  2. Return this *number*, which is a card.
* A **showCard** function that converts the card number to a printable value:
  1. Accept a card (a number between 1 and 13)
  2. Return the corresponding card value to be printed, which is:
     + the number itself, for the numbered cards between 2 and 10
     + 'A' for 1, 'J' for 11, 'Q' for 12, 'K' for 13
* An **updateTotal** function that updates the total of one player:
  1. Accept the current total and a card (a number between 1 and 13)
  2. Add the card value to the total. For the simple version of the game,
     + the A, J, Q, K cards each have the value 10
     + the numbered cards each have the value of their corresponding number
  3. Return the new total
* A **oneGame** function that plays one game by calling the previous 3 functions and coordinating them.   
  Here is the pseudocode for one game:
  1. get the first card for the computer
  2. update the computer total
  3. get the second card for the computer
  4. update the computer total
  5. get the first card for the user
  6. update the user total
  7. get the second card for the user
  8. update the user total
  9. print only 1 card for the computer (see sample output)
  10. print both cards for the user (see sample output)
  11. loop to ask the user whether they want another card, and as long as the answer is 'y':  
       a. get a card for the user  
       b. print the card (see sample output)  
       c. update the user total  
       d. determine if the user has won (total is 21) or if the user has lost (total above 21) or go back to   
       the top of the loop (step 11)
  12. when the user is done getting cards and if the user hasn't lost or won yet:  
       a. print the 2nd computer card (see sample output)  
       b. loop until the computer has won (total is 21) or until the computer has lost (total above 21) or   
       until the total is above 17. In the loop:  
       - get another card  
       - print the new card (see sample output)  
       - update the computer total
  13. when the computer is done getting cards and if the computer hasn't lost or won yet, compare the user total and the computer total
  14. print the winner. Note that the winner can be determined from step 11d, 12b, or 13.
* A **main** function:

Loop as long as the user answers 'y':

* + play one game
  + ask the user whether to play again and accepts 'y' to continue, stop for anything else entered.

**Additional requirements**

* Make sure to have your name and lab number at the top of the file
* Have a *docstring* for all functions except main
* Read the sample test case so you know what to print when there's a winner and what to do when the user doesn't enter the correct input.
* Do not use lists or any other container for this lab. There's no need to use them.

**Test**

Run your program continuously (user answers 'y' when asked to continue the game) until you've seen a player going above 21, a player at 21, and both players are below 21.

Sample output

My cards: Q and ? # only one dealer card is shown at first

Your cards: 7 5

one more card? y/n: y

Your card: 9

You won! # user automatically wins

Your total is 21 # no need to show dealer total

Play another game? y/n: y

My cards: 4 and ?

Your cards: 3 10

one more card? y/n: y

Your card: 4

one more card? y/n: k # accept 'y' or 'n' only or keep prompting

one more card? y/n: y

Your card: A

I won!

Your total is 27 # user automatically loses, no need to print dealer total

Play another game? y/n: y

My cards: K and ?

Your cards: 6 4

one more card? y/n: y

Your card: 10

one more card? y/n: n # user stops getting cards

My other card is 5 # show dealer's 2nd card

My card: 4 # dealer stops getting card because total >= 17

You won!

Your total: 20 # user wins by comparing totals

My total: 19 # in this case print both user and dealer totals

Play another game? y/n: n # user can also enter anything other than 'y' to stop