**blackjack\_module.py**

import random

full\_deck = {"Two of clubs": 2, "Three of clubs": 3, "Four of clubs": 4, "Five of clubs": 5, "Six of clubs": 6,

"Seven of clubs": 7, "Eight of clubs": 8, "Nine of clubs": 9, "Ten of clubs": 10,

"Jack of clubs": 10, "Queen of clubs": 10, "King of clubs": 10, "Ace of clubs": 11,

"Two of diamonds": 2, "Three of diamonds": 3, "Four of diamonds": 4, "Five of diamonds": 5,

"Six of diamonds": 6, "Seven of diamonds": 7, "Eight of diamonds": 8, "Nine of diamonds": 9,

"Ten of diamonds": 10, "Jack of diamonds": 10, "Queen of diamonds": 10, "King of diamonds": 10,

"Ace of diamonds": 11, "Two of hearts": 2, "Three of hearts": 3, "Four of hearts":

4, "Five of hearts": 5, "Six of hearts": 6,

"Seven of hearts": 7, "Eight of hearts": 8, "Nine of hearts": 9, "Ten of hearts": 10,

"Jack of hearts": 10, "Queen of hearts": 10, "King of hearts": 10, "Ace of hearts": 11,

"Two of spades": 2, "Three of spades": 3, "Four of spades": 4, "Five of spades": 5, "Six of spades": 6,

"Seven of spades": 7, "Eight of spades": 8, "Nine of spades": 9, "Ten of spades": 10,

"Jack of spades": 10, "Queen of spades": 10, "King of spades": 10, "Ace of spades": 11,

}

def get\_new\_shuffled\_deck():

deck = list(full\_deck.keys())

random.shuffle(deck)

return deck

def get\_card\_value(card):

return full\_deck[card]

def calculate\_hand\_value(hand):

hand\_value = 0

for card in hand:

hand\_value += get\_card\_value(card)

return hand\_value

**Exercise 1.1. - Deck and hand**

using blackjack\_module.py above create a new python file, this is where the bulk of your code will be written. Import the module you have added to your project and feel free to give it an alias, like this:

import blackjack\_module as bjm

Use the get\_new\_shuffled\_deck () function when creating a variable that holds a list of cards (representing a shuffled deck of cards). This is used when you hand out cards later.  
Make two lists, one representing the player's hand and one for the dealer.  
Fill the player's and dealer's hand with two cards from the deck. (can make this as a function?)  
Write a printout of what both the player has on hand, as well as the value of these, and the first card the dealer has on hand. For example. something like this:

"The cards have been dealt. You have an Eight of clubs and a Three of hearts, with a total value of 11.  
The dealers visible card is a Five of hearts, with a value of 5. "

**Exercise 1.2 - BlackJack**  
A)  
Check if the player's hand is BlackJack (has a value of 21). In that case, write a suitable printout.  
You can use the calculate\_hand\_value function to get the value of a hand.

B)  
If it is not BlackJack, give the player two choices:  
1 - Hit  
2 - Stand

Print the choice the user made  
Initially, create a simple implementation of "Stand":

C)  
Create an implementation if the user selects "Hit":  
Give the player a new card from the deck  
Print the drawn card, and the updated hand and its value  
Check if the player is "busted" (has over 21 in total value)  
D)  
Put all this logic in a loop that ends when the user selects "Stand" or the player is busted.

When all these sub-tasks are finished, you have a fully implemented (albeit somewhat simpler) round of BlackJack! :)

**Task 1.3 - Reset and play several rounds**

Allows you to play multiple rounds. After a round, you ask if the user wants to play another round or not. If another round is to be played, you must: empty your hands and get a new deck of cards.

**Task 1.4 - Dealers round**

We will expand the program to also address that the dealer can get more cards than two. Now that the user has selected "Stand", the dealer will receive a new card until the hand has a value of 17 or more.

Print the entire dealer's hand and the value of the hand  
Prints the player's hand and the value of the hand  
Create a function "print\_result ()" which takes the value of the player's hand and the value of the dealer's hand as parameters. This should compare these and make different prints based on the result:  
The dealer has over 21 and "busted", the player has won  
The player has higher than the dealer, the player has won  
The dealer has higher than the player, the player has lost (the house won)  
The player and the dealer have liked, the player has neither won nor lost

**Exercise 1.5 - Aces as 1 or 11**

Make changes to the function calculate\_hand\_value (hand) and take into account that an ace can have the value 1 instead of 11 (if the total value is over 21, you want the ace to count as 1).  
For its part, it is sufficient to do this for one ace. If you feel like an extra challenge, you can try to handle the situation where a player potentially has several aces.

**Exercise 1.6 - Chips**

A)

Create a variable that holds the number of chips. Set the initial value to 5.

Before the hands are dealt, print out how many chips the user has, and take the player's bet as input.

Bring updates and prints for the number of chips places it is natural. It may make sense to change the "print\_result ()" function so that it also takes the number of chips and bets as parameters to achieve this.

The rules for the chips are as follows:

The player wins -> The player earns as many chips as specified in the bet  
The dealer wins -> The player loses as many chips as specified in the bet  
No winner -> The player chips remain the same as before the bet  
Blackjack -> The player earns 2 x the number of chips specified in the bet  
B)

Add handling so that the user cannot enter a bet that is not a number (hint try-except)

Check that the player's bet is valid (not less than 1 or greater than available chips).

If what the user enters is invalid in one way or another, the user must be allowed to enter the bet again.

C)

If the player runs out of chips, the game must end.

**Task 1.7 - Improve the code**

Review the code you created to see if there is anything you can do to improve it. Is it e.g. repetitive code that may be suitable for distinguishing features?