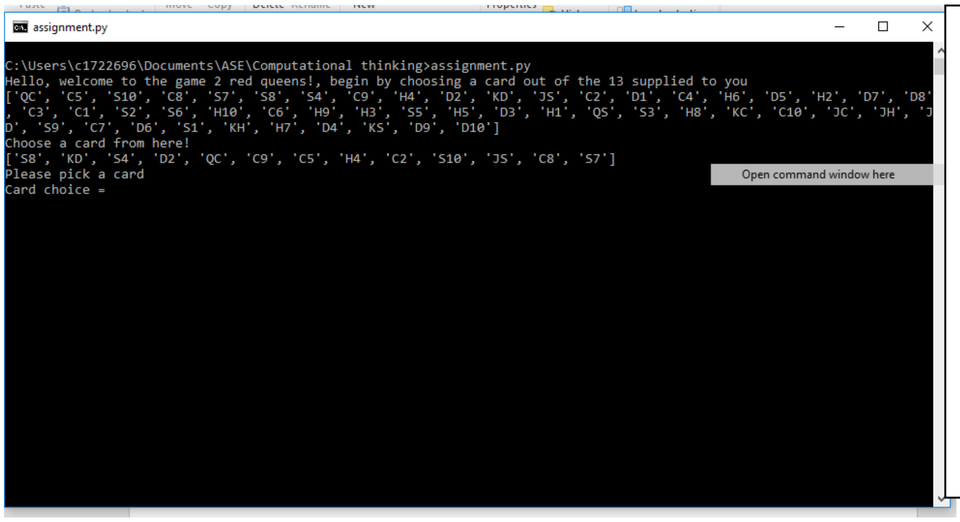
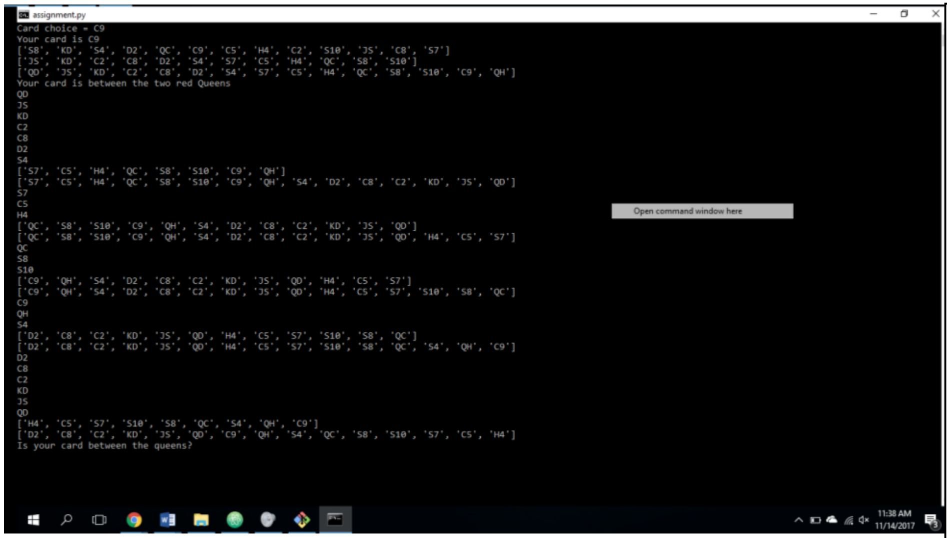


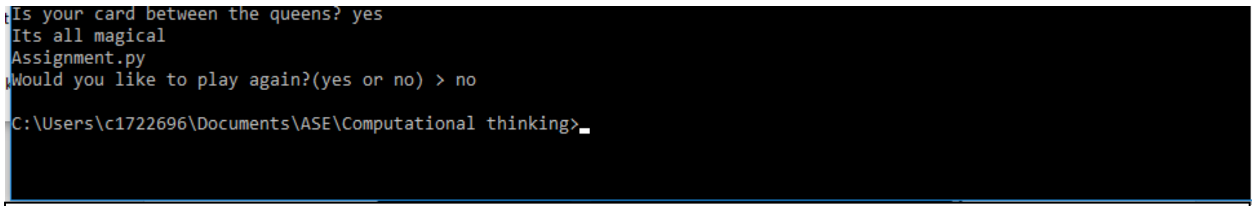
```
C:\Users\c1722696\Documents\ASE\Computational thinking>assignment.py
Hello, welcome to the game 2 red queens!, begin by choosing a card out of the 13 supplied to you
['QC', 'C5', 'S10', 'C8', 'S7', 'S8', 'S4', 'C9', 'H4', 'D2', 'KD', 'JS', 'C2', 'D1', 'C4', 'H6', 'D5', 'H2', 'D7', 'D8'
, 'C3', 'C1', 'S2', 'S6', 'H10', 'C6', 'H9', 'H3', 'S5', 'H5', 'D3', 'H1', 'QS', 'S3', 'H8', 'KC', 'C10', 'JC', 'JH', 'J
D', 'S9', 'C7', 'D6', 'S1', 'KH', 'H7', 'D4', 'KS', 'D9', 'D10']
Choose a card from here!
['S8', 'KD', 'S4', 'D2', 'QC', 'C9', 'C5', 'H4', 'C2', 'S10', 'JS', 'C8', 'S7']
Please pick a card
Card choice =
```

Here we can see the first part of the game in which the card deck is shuffled and the user is asked whether they would like to pick a card.



```
Card choice = C9
Your card is C9
['S8', 'KD', 'S4', 'D2', 'QC', 'C9', 'C5', 'H4', 'C2', 'S10', 'JS', 'C8', 'S7']
['JS', 'KD', 'C2', 'C8', 'D2', 'S4', 'S7', 'C5', 'H4', 'QC', 'S8', 'S10']
['QD', 'JS', 'KD', 'C2', 'C8', 'D2', 'S4', 'S7', 'C5', 'H4', 'QC', 'S8', 'S10', 'C9', 'QH']
Your card is between the two red Queens
QD
JS
KD
C2
C8
D2
S4
['S7', 'C5', 'H4', 'QC', 'S8', 'S10', 'C9', 'QH']
['S7', 'C5', 'H4', 'QC', 'S8', 'S10', 'C9', 'QH', 'S4', 'D2', 'C8', 'C2', 'KD', 'JS', 'QD']
S7
C5
H4
['QC', 'S8', 'S10', 'C9', 'QH', 'S4', 'D2', 'C8', 'C2', 'KD', 'JS', 'QD']
['QC', 'S8', 'S10', 'C9', 'QH', 'S4', 'D2', 'C8', 'C2', 'KD', 'JS', 'QD', 'H4', 'C5', 'S7']
QC
S8
S10
['C9', 'QH', 'S4', 'D2', 'C8', 'C2', 'KD', 'JS', 'QD', 'H4', 'C5', 'S7']
['C9', 'QH', 'S4', 'D2', 'C8', 'C2', 'KD', 'JS', 'QD', 'H4', 'C5', 'S7', 'S10', 'S8', 'QC']
C9
QH
S4
['D2', 'C8', 'C2', 'KD', 'JS', 'QD', 'H4', 'C5', 'S7', 'S10', 'S8', 'QC']
['D2', 'C8', 'C2', 'KD', 'JS', 'QD', 'H4', 'C5', 'S7', 'S10', 'S8', 'QC', 'S4', 'QH', 'C9']
D2
C8
C2
KD
JS
QD
['H4', 'C5', 'S7', 'S10', 'S8', 'QC', 'S4', 'QH', 'C9']
['D2', 'C8', 'C2', 'KD', 'JS', 'QD', 'C9', 'QH', 'S4', 'QC', 'S8', 'S10', 'S7', 'C5', 'H4']
Is your card between the queens?
```

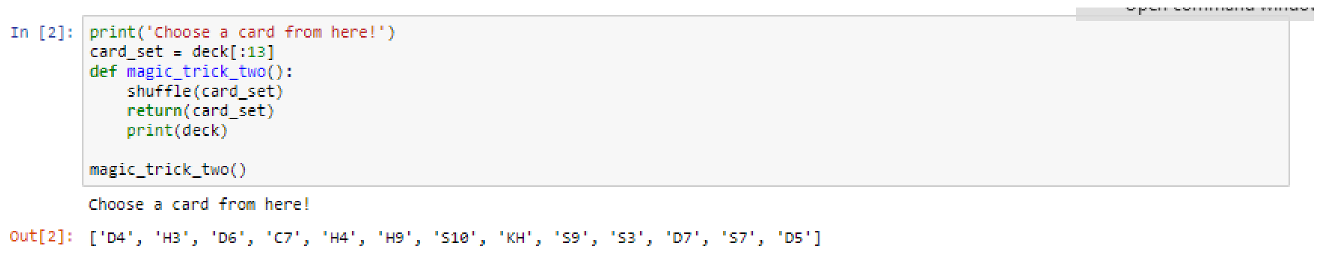
We then shuffle the 13 cards and then add the 2 red queens and the users card to their set positions. We then go through each card set the set amount of times being 7,3,3,3,6. The final result is then printed and the user is asked whether their card is within the two red queens.



```
Is your card between the queens? yes
Its all magical
Assignment.py
Would you like to play again?(yes or no) > no

C:\Users\c1722696\Documents\ASE\Computational thinking>
```

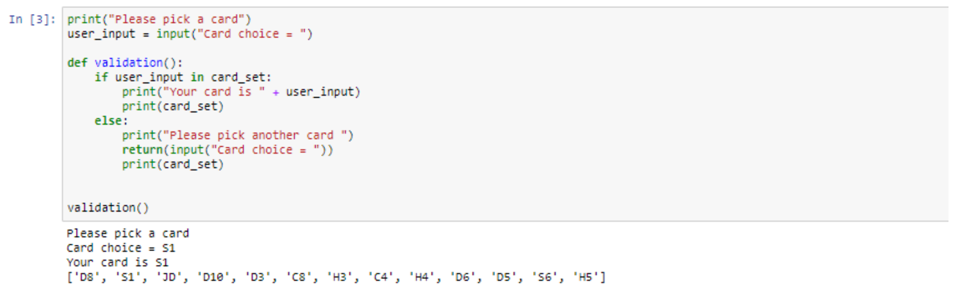
If the user replies yes (which they hopefully should) then the system should print 'it's all magical' and should ask whether the user wants to play again. If so then the game should restart (however this feature isn't 100% right now) but if the user says no as you see above the program is exited.

What	Type	What it’s used for
Deck	List/output	This was used to hold my deck of cards which I went on to use to shuffle and draw 13 cards from
Suits	List	This held the suits that `I used
Class	List	This held the Classes that I used
Card_set	List/output	The card set was one of my main lists. It was effectively an output of my deck list and give an output of the 13 cards which the user chooses.
user_input	Input/variable	This was used to store the user input which we could validate and used later on in the program when placing the users card back.
card_set_two	List	A card set list I created to store the new order of the values from card_set
card_set_three	List	A card set list I created to store the new order of the values from card_set_2
card_set_four	List	A card set list I created to store the new order of the values from card_set_3
card_set_five	List	A card set list I created to store the new order of the values from card_set_4
final_set	List	A card set list I created to store the new order of the values from card_set_5 and print the final deck of cards in which the card should be between the two red queens.
game	Variable	In this variable inside the function I held my game as a variable and so that if the user in the if statement said yes to wanting to play again the function would re-run my game.



```
In [2]: print('Choose a card from here!')
        card_set = deck[:13]
        def magic_trick_two():
            shuffle(card_set)
            return(card_set)
            print(deck)

        magic_trick_two()

        Choose a card from here!
Out[2]: ['D4', 'H3', 'D6', 'C7', 'H4', 'H9', 'S10', 'KH', 'S9', 'S3', 'D7', 'S7', 'D5']
```

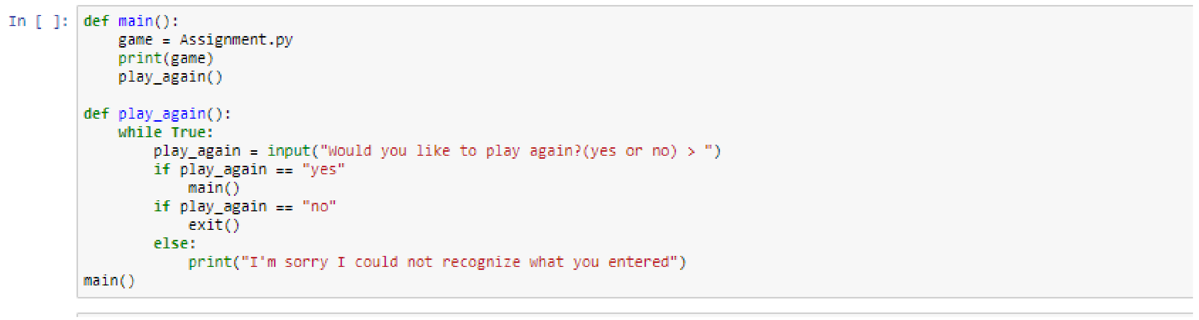
Here we see an example of a function which takes 13 cards from the deck. I defined it as *magic_trick_two* to help me keep track of my functions during my coding. The result of the function is 13 cards of which the user chooses one when called via *magic_trick_two() on the bottom line.*



```
In [3]: print("Please pick a card")
        user_input = input("Card choice = ")

        def validation():
            if user_input in card_set:
                print("Your card is " + user_input)
                print(card_set)
            else:
                print("Please pick another card ")
                return(input("Card choice = "))
                print(card_set)


        validation()

        Please pick a card
        Card choice = S1
        Your card is S1
        ['D8', 'S1', 'JD', 'D10', 'D3', 'C8', 'H3', 'C4', 'H4', 'D6', 'D5', 'S6', 'H5']
```

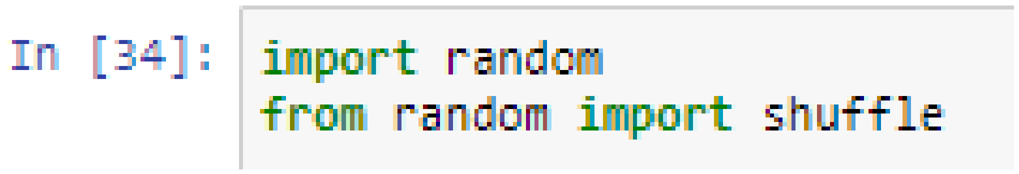
We see control flow here in the if statement that checks as to whether the user card is in the card set as it will go on to print *'Your card is ""'* and print the card set whereas if an incorrect card is named we will see the else statement come into effect as the user is asked to pick another card which will bring up their card choice and print the card set after.

I Didn't use run time error handling in my game as I wasn't aware of it while coding and I wasn't very confident in the area however I know this could be done by potentially implementing a Try catch of some sort or using a runtime error function.



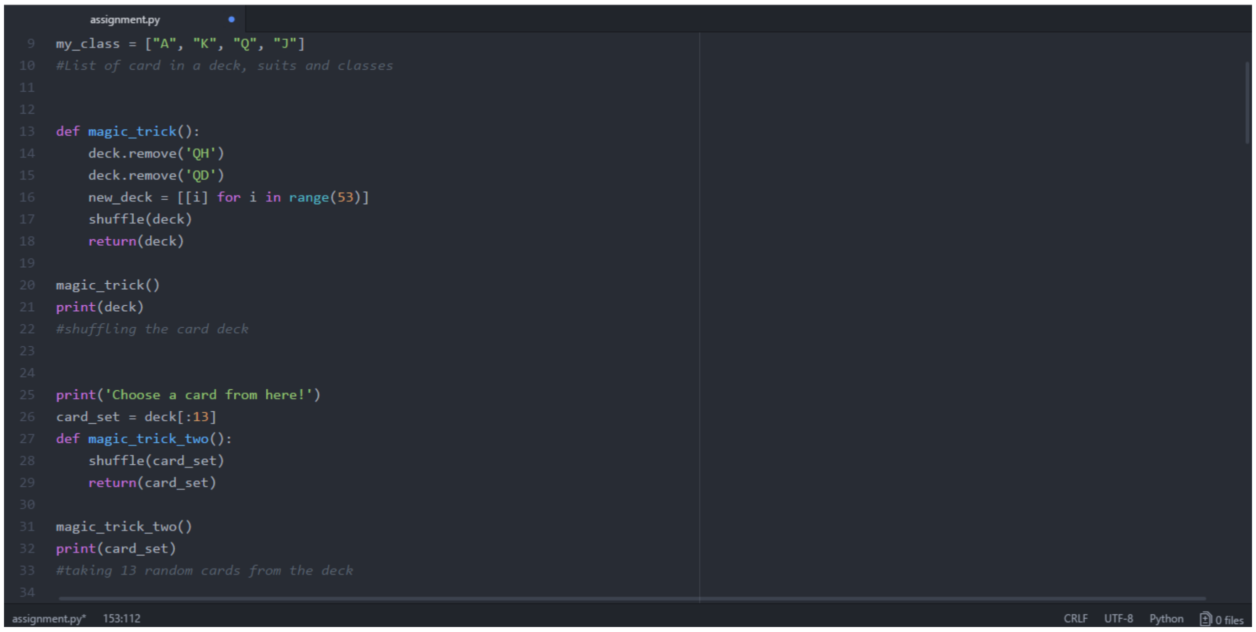
```
In [ ]: def main():
            game = Assignment.py
            print(game)
            play_again()

        def play_again():
            while True:
                play_again = input("Would you like to play again?(yes or no) > ")
                if play_again == "yes"
                    main()
                if play_again == "no"
                    exit()
                else:
                    print("I'm sorry I could not recognize what you entered")
        main()
```

Here we can see my use of loops in my final function as it says *while True*. This is followed by two if statements and an else statement of which the if statement will either restart the game if yes is typed by the user or the game will simply exit if no is typed by the user. The else statement makes sure a valid input is inserted.



```
In [34]: import random
         from random import shuffle
```

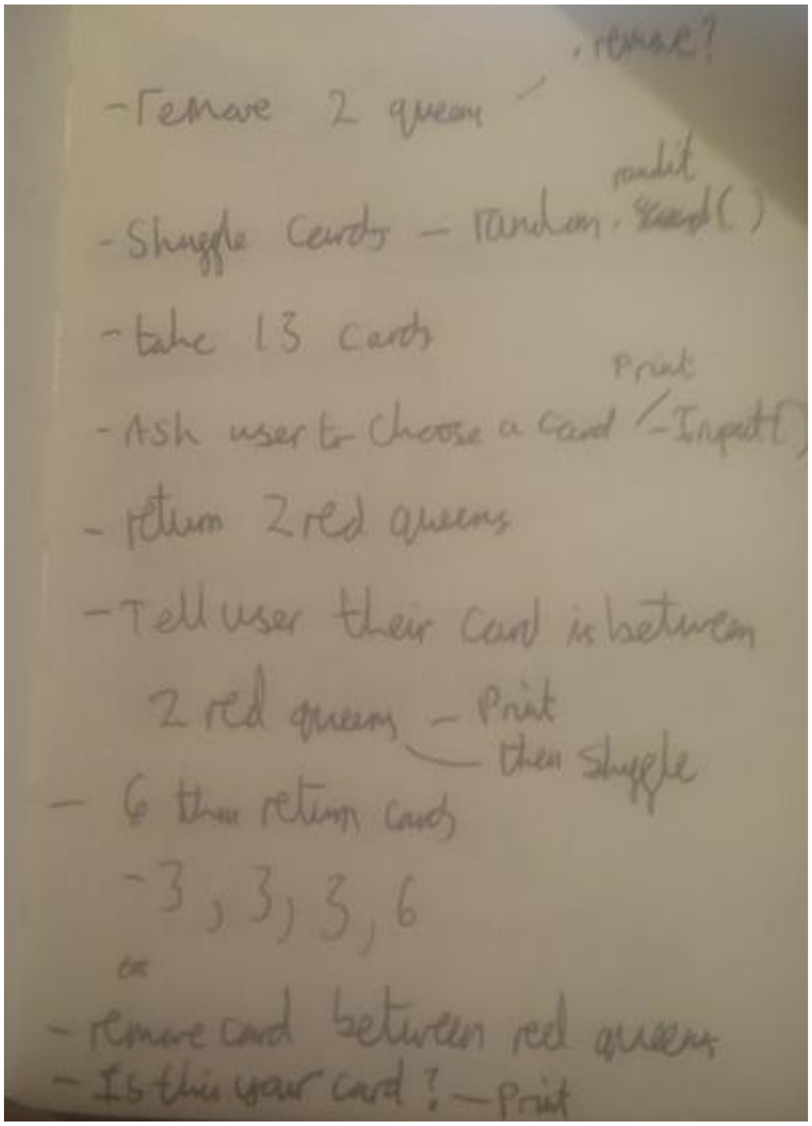
I imported random and shuffle as they are important in allowing me to shuffle the cards in the deck and randomize selection features. These are key parts of the trick that keep the user constantly guessing along with a random deck being quite necessary in any card trick.



```python
my_class = ["A", "K", "Q", "J"]
#List of card in a deck, suits and classes


def magic_trick():
    deck.remove('QH')
    deck.remove('QD')
    new_deck = [[i] for i in range(53)]
    shuffle(deck)
    return(deck)

magic_trick()
print(deck)
#shuffling the card deck


print('Choose a card from here!')
card_set = deck[:13]
def magic_trick_two():
    shuffle(card_set)
    return(card_set)

magic_trick_two()
print(card_set)
#taking 13 random cards from the deck

```

Here you can see an example of the comments that I put in my code. I tried to comment as if I was returning to the code in 6 months from leaving it now so after each function I stated what it did just so it would be easier for me or a colleague to understand. I didn't put comments for the print statements as they are pretty self-explanatory.

Just as an extra here's my planning before I started coding:



.remove?

- remove 2 queens ✓

randint

- Shuffle cards – random.seed( )

- take 13 cards

Print

- Ask user to choose a card ✓ – Input( )

- return 2 red queens

- Tell user their card is between 2 red queens – Print – then shuffle

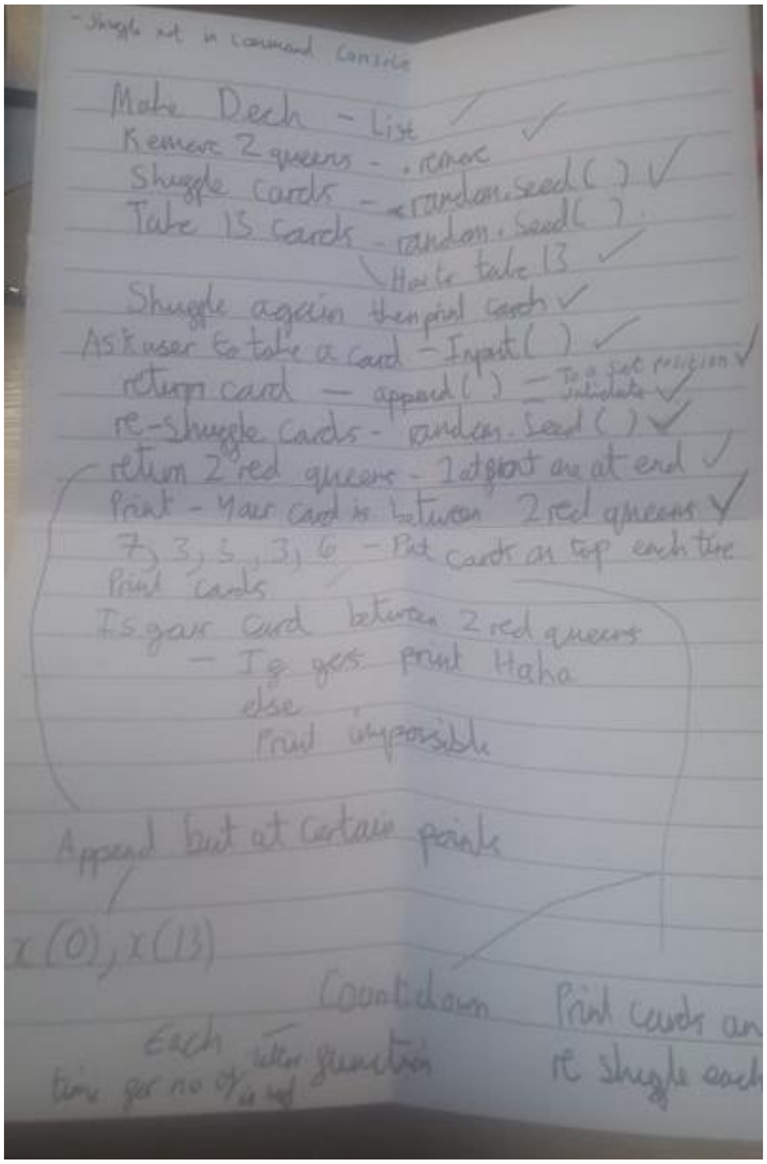
- 6 then return cards

- 3, 3, 3, 6

or

- remove card between red queens

- Is this your card? – Print



- Shuffle out in command console

Make Deck – List ✓ ✓

Remove 2 queens – .remove

Shuffle cards – random.seed( ) ✓

Take 13 cards – random.seed( )

How to take 13 ✓

Shuffle again then print cards ✓

Ask user to take a card – Input( ) ✓

return card – append( ) – To a set position / Initiate ✓ ✓

re-shuffle cards – random.seed( ) ✓

return 2 red queens – 1st front one at end ✓

Print – Your card is between 2 red queens ✓

7, 3, 3, 3, 6 – Put cards on top each time

Print cards

Is your card between 2 red queens

– If yes print Haha

else

Print impossible

Append but at certain points

x(0), x(13)

Countdown

Each time for no of iter in set / iter function

Print cards an

re shuffle each