Task 3 - Card Game

Analysis

Success Criteria

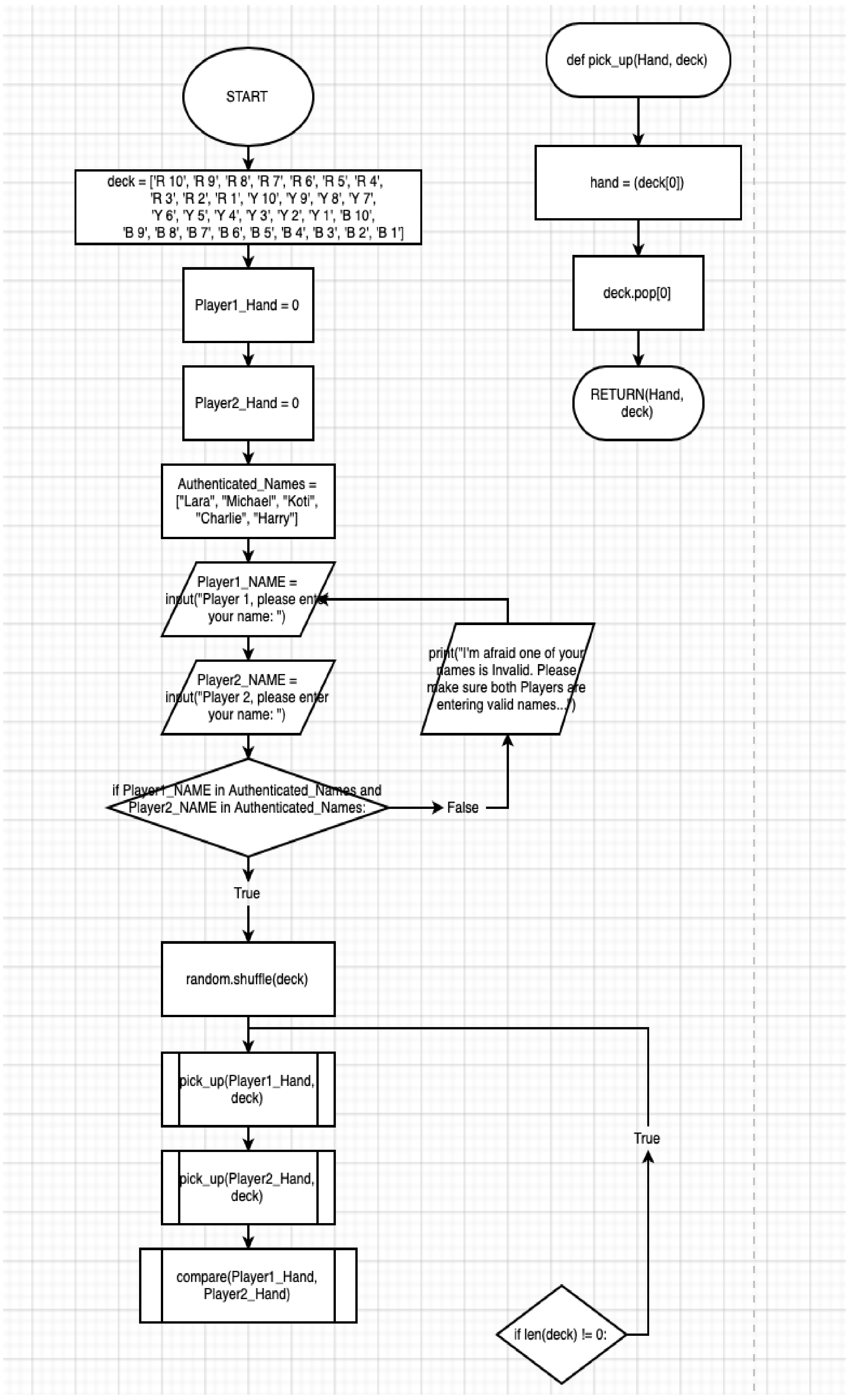
1. There must be 30 cards each card must have:
    - A colour assigned to it, this colour must be RED, BLACK or YELLOW
    - For each colour there must be ten numbers, one for each card.
    - There should be 10 RED cards, 10 BLACK cards and 10 YELLOW cards.
2. These cards must be shuffled and stored in the deck (a list called “deck”)
3. Every turn each player takes a card from the top of the deck. Player 1 starts by taking a card and then Player 2 follows suit.
4. The two cards picked up by the 2 players are compared by:
    - Comparing the colours of the two cards, if they are the same then:
        - Compare the size of the number, the player with the higher number wins both cards
    - If they are different colours then they are compared with the table shown below:

Card	Card	Winner
Red	Black	Red
Yellow	Red	Yellow
Black	Yellow	Black

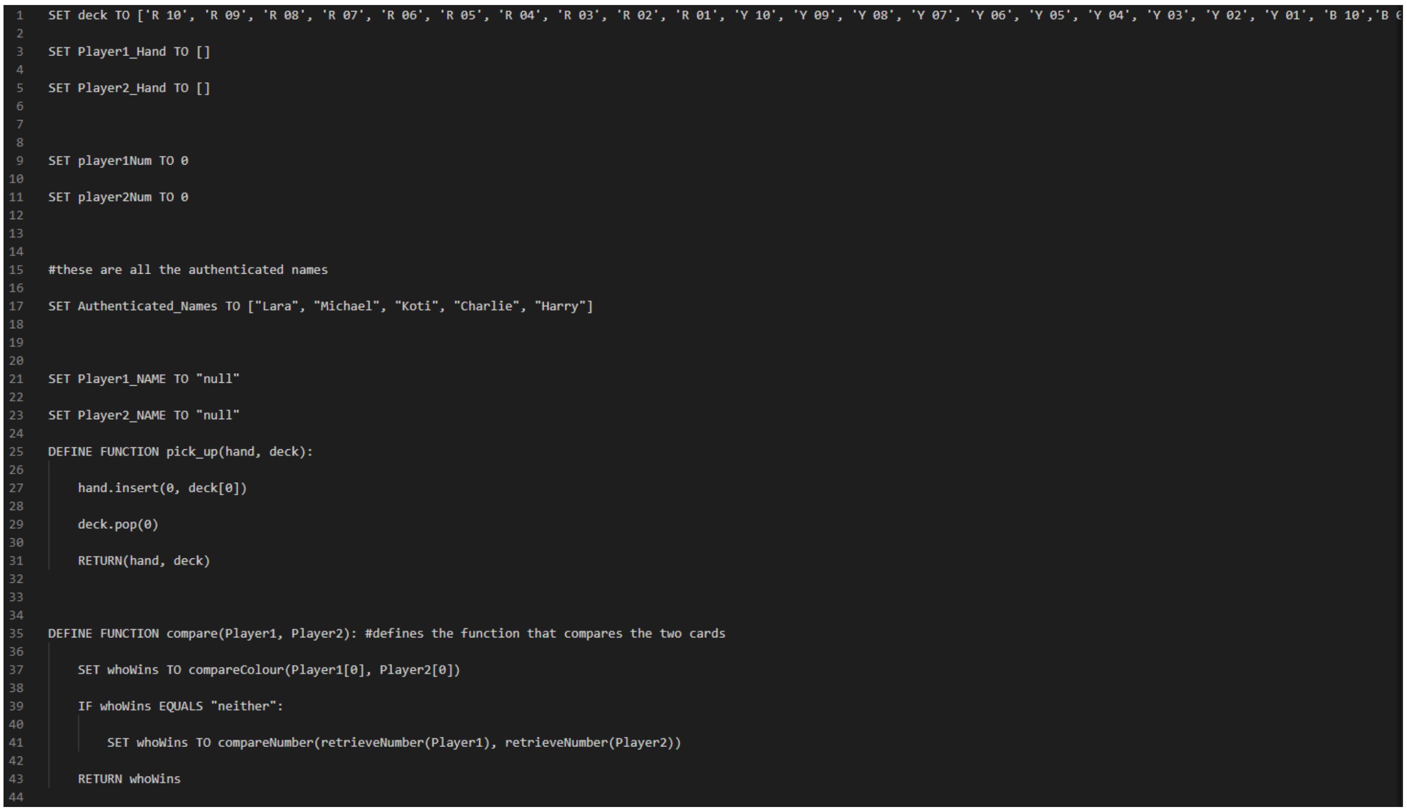
5. The winner of each round keeps both cards played
6. At the start, let the two players enter their details, which are then checked by the program to ensure both players are authorised players.
7. Once the game is finished the amount of cards the winner has is outputted and saved to an external save file with the authenticated players name.
8. The program should have the ability to display the name and quantity of cards of the 5 players with the highest quantity of cards, from the external file.

Justification of Design and initial Code Ideas:

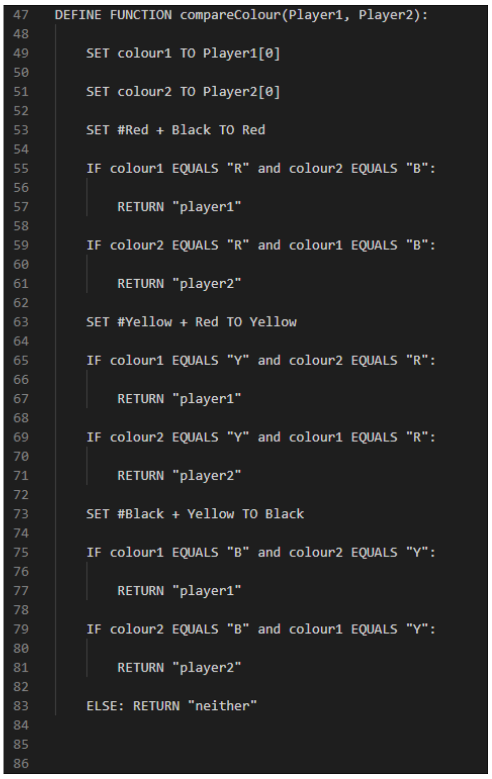
1. I will create a list called ‘deck’ which will contain 30 ‘Cards’ (strings) each assigned with a colour. 10 Items (1-10) will be assigned the Red colour, 10 Items (1-10) will be assigned the Yellow colour and 10 Items (1-10) will be assigned the Black colour.
2. Player 1 and Player 2 will be asked for their information, this information will be compared to a variable which will contain the needed information to access the game. It will also be accessed at the end of the game to be saved into the external file if either player wins.
3. I will use the shuffle function from the Random Library to randomize the items in the list ‘deck’, I will then use the indexs of each item to retrieve the ‘card’ from the top of the deck.
4. Using the index of each item, the program will retrieve a card from the top of the ‘deck’, this card will then be subtracted from the list ‘deck’ and added to ‘Player#_Picked’ (hash being the number of the Player) Then this will happen again except the Card will be added to the other players ‘Picked’ variable.
5. Then the two cards in the players 'Picked’ variable will be compared.
6. This will then repeat for the other player until the entire ‘deck’ has been depleted.


START
deck = ['R 10', 'R 9', 'R 8', 'R 7', 'R 6', 'R 5', 'R 4', 'R 3', 'R 2', 'R 1', 'Y 10', 'Y 9', 'Y 8', 'Y 7', 'Y 6', 'Y 5', 'Y 4', 'Y 3', 'Y 2', 'Y 1', 'B 10', 'B 9', 'B 8', 'B 7', 'B 6', 'B 5', 'B 4', 'B 3', 'B 2', 'B 1']
Player1_Hand = 0
Player2_Hand = 0
Authenticated_Names = ["Lara", "Michael", "Koti", "Charlie", "Harry"]
Player1_NAME = input("Player 1, please enter your name: ")
Player2_NAME = input("Player 2, please enter your name: ")
if Player1_NAME in Authenticated_Names and Player2_NAME in Authenticated_Names:
False
print("I'm afraid one of your names is Invalid. Please make sure both Players are entering valid names...")
True
random.shuffle(deck)
pick_up(Player1_Hand, deck)
pick_up(Player2_Hand, deck)
compare(Player1_Hand, Player2_Hand)
if len(deck) != 0:
True
def pick_up(Hand, deck)
hand = (deck[0])
deck.pop[0]
RETURN(Hand, deck)

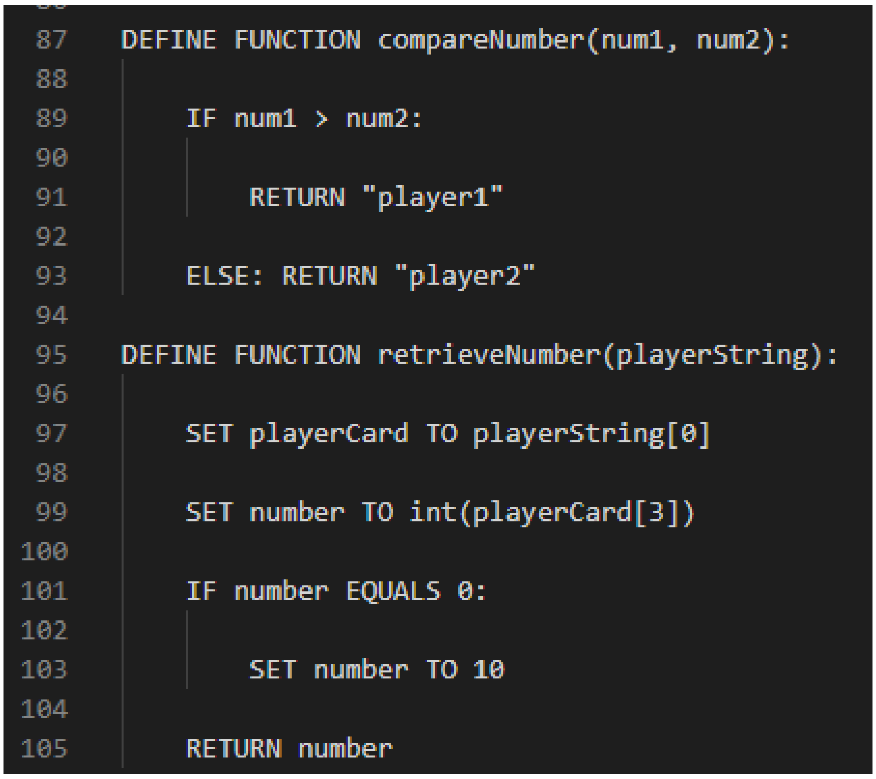
PSEUDO Code



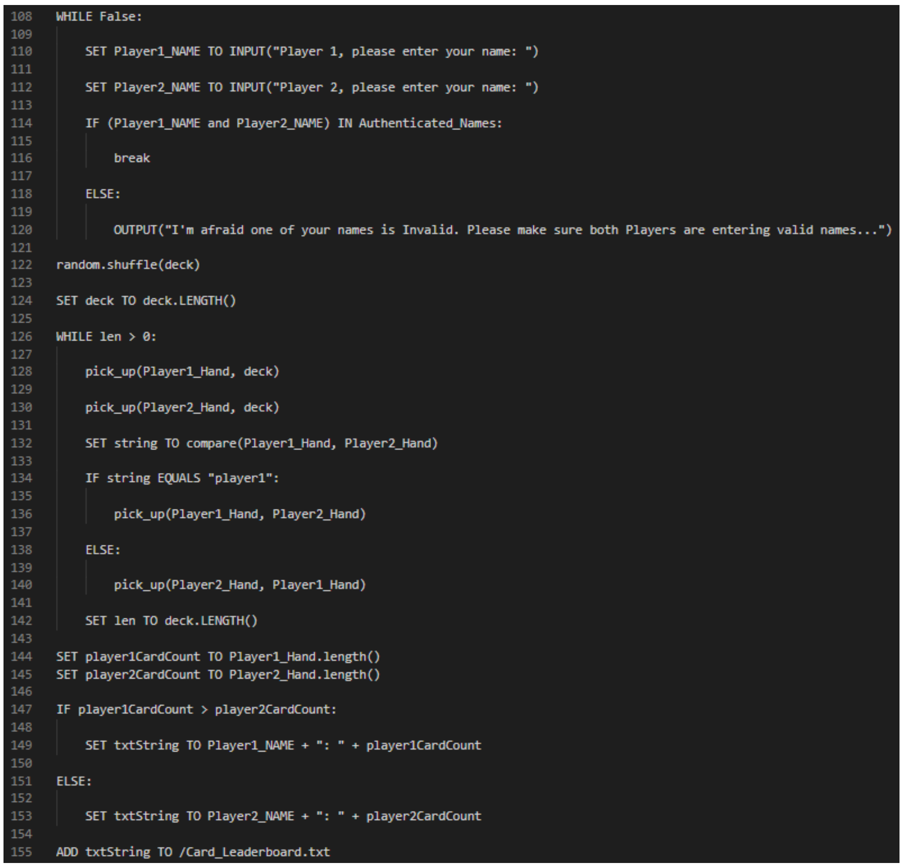
```
SET deck TO ['R 10', 'R 09', 'R 08', 'R 07', 'R 06', 'R 05', 'R 04', 'R 03', 'R 02', 'R 01', 'Y 10', 'Y 09', 'Y 08', 'Y 07', 'Y 06', 'Y 05', 'Y 04', 'Y 03', 'Y 02', 'Y 01', 'B 10','B

SET Player1_Hand TO []

SET Player2_Hand TO []



SET player1Num TO 0

SET player2Num TO 0



#these are all the authenticated names

SET Authenticated_Names TO ["Lara", "Michael", "Koti", "Charlie", "Harry"]



SET Player1_NAME TO "null"

SET Player2_NAME TO "null"

DEFINE FUNCTION pick_up(hand, deck):

    hand.insert(0, deck[0])

    deck.pop(0)

    RETURN(hand, deck)



DEFINE FUNCTION compare(Player1, Player2): #defines the function that compares the two cards

    SET whoWins TO compareColour(Player1[0], Player2[0])

    IF whoWins EQUALS "neither":

        SET whoWins TO compareNumber(retrieveNumber(Player1), retrieveNumber(Player2))

    RETURN whoWins

```



```
DEFINE FUNCTION compareColour(Player1, Player2):

    SET colour1 TO Player1[0]

    SET colour2 TO Player2[0]

    SET #Red + Black TO Red

    IF colour1 EQUALS "R" and colour2 EQUALS "B":

        RETURN "player1"

    IF colour2 EQUALS "R" and colour1 EQUALS "B":

        RETURN "player2"

    SET #Yellow + Red TO Yellow

    IF colour1 EQUALS "Y" and colour2 EQUALS "R":

        RETURN "player1"

    IF colour2 EQUALS "Y" and colour1 EQUALS "R":

        RETURN "player2"

    SET #Black + Yellow TO Black

    IF colour1 EQUALS "B" and colour2 EQUALS "Y":

        RETURN "player1"

    IF colour2 EQUALS "B" and colour1 EQUALS "Y":

        RETURN "player2"

    ELSE: RETURN "neither"

```



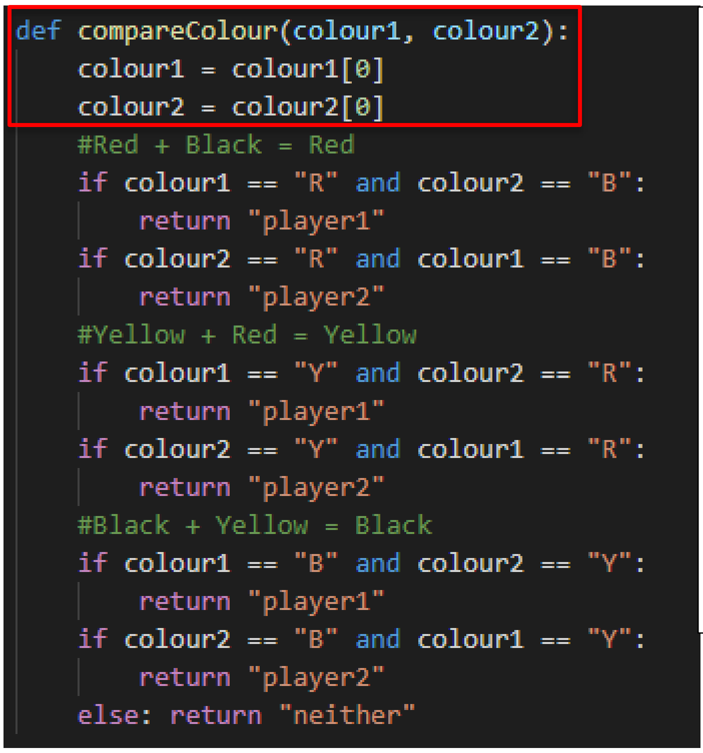
```
DEFINE FUNCTION compareNumber(num1, num2):

    IF num1 > num2:

        RETURN "player1"

    ELSE: RETURN "player2"

DEFINE FUNCTION retrieveNumber(playerString):

    SET playerCard TO playerString[0]

    SET number TO int(playerCard[3])

    IF number EQUALS 0:

        SET number TO 10

    RETURN number
```



```
WHILE False:

    SET Player1_NAME TO INPUT("Player 1, please enter your name: ")

    SET Player2_NAME TO INPUT("Player 2, please enter your name: ")

    IF (Player1_NAME and Player2_NAME) IN Authenticated_Names:

        break

    ELSE:

        OUTPUT("I'm afraid one of your names is Invalid. Please make sure both Players are entering valid names...")

random.shuffle(deck)

SET deck TO deck.LENGTH()

WHILE len > 0:

    pick_up(Player1_Hand, deck)

    pick_up(Player2_Hand, deck)

    SET string TO compare(Player1_Hand, Player2_Hand)

    IF string EQUALS "player1":

        pick_up(Player1_Hand, Player2_Hand)

    ELSE:

        pick_up(Player2_Hand, Player1_Hand)

    SET len TO deck.LENGTH()

SET player1CardCount TO Player1_Hand.length()
SET player2CardCount TO Player2_Hand.length()

IF player1CardCount > player2CardCount:

    SET txtString TO Player1_NAME + ": " + player1CardCount

ELSE:

    SET txtString TO Player2_NAME + ": " + player2CardCount

ADD txtString TO /Card_Leaderboard.txt
```

Python Code

Day 2

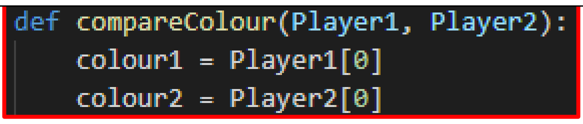


```python
def compareColour(colour1, colour2):
    colour1 = colour1[0]
    colour2 = colour2[0]
    #Red + Black = Red
    if colour1 == "R" and colour2 == "B":
        return "player1"
    if colour2 == "R" and colour1 == "B":
        return "player2"
    #Yellow + Red = Yellow
    if colour1 == "Y" and colour2 == "R":
        return "player1"
    if colour2 == "Y" and colour1 == "R":
        return "player2"
    #Black + Yellow = Black
    if colour1 == "B" and colour2 == "Y":
        return "player1"
    if colour2 == "B" and colour1 == "Y":
        return "player2"
    else: return "neither"
```

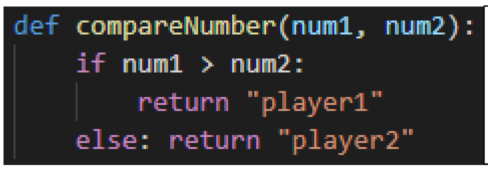
This function compares two colours by taking two strings (`'R 10'` and `'Y 09'`) reading the first value from each string (`R` and `Y`) and comparing them to the table.

Card	Card	Winner
Red	Black	Red
Yellow	Red	Yellow
Black	Yellow	Black

Then, the program returns a string containing the player that won (this function assumes that colour1 comes from player1). Something that I changed later to make more obvious:

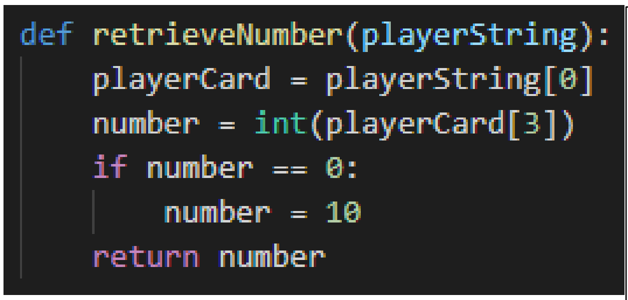


```python
def compareColour(Player1, Player2):
    colour1 = Player1[0]
    colour2 = Player2[0]
```



```python
def compareNumber(num1, num2):
    if num1 > num2:
        return "player1"
    else: return "player2"
```

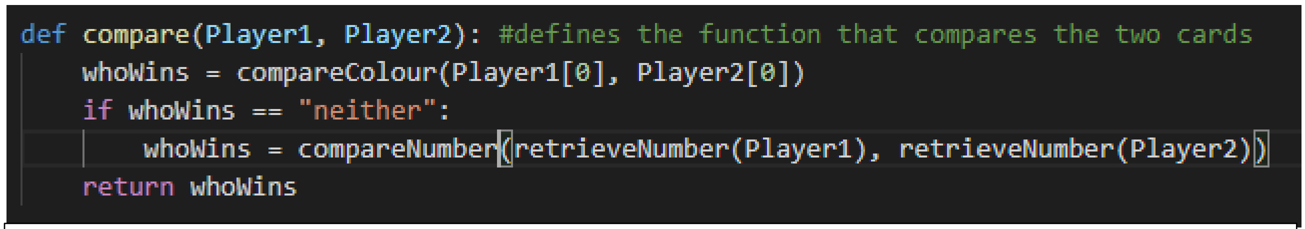
This takes two numbers, (num1 being player1 and num2 being player2) and returns a string containing the player with the highest number.



```python
def retrieveNumber(playerString):
    playerCard = playerString[0]
    number = int(playerCard[3])
    if number == 0:
        number = 10
    return number
```

This takes the players string (so the card) and retrieves the number contained in it.

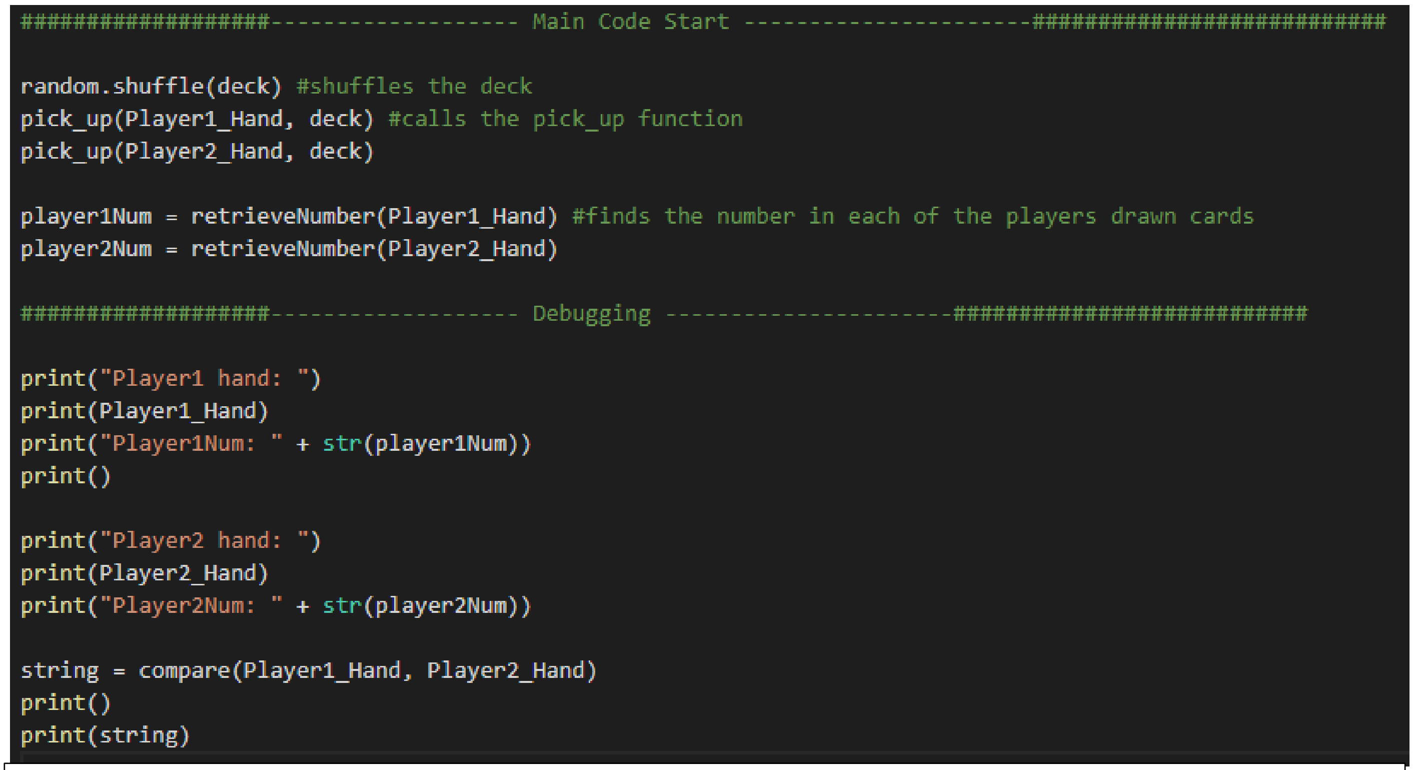
It also converts the character at: playerCard, index 3 to an integer so it can check if the number == 0. This means the program can account for the number 10 without using a complicated algorithm. In hindsight, I could've made the range of number 0-9, which could also make the program more efficient. (May add this later)



```python
def compare(Player1, Player2): #defines the function that compares the two cards
    whoWins = compareColour(Player1[0], Player2[0])
    if whoWins == "neither":
        whoWins = compareNumber(retrieveNumber(Player1), retrieveNumber(Player2))
    return whoWins
```

This is where all of the comparing functions are called from, this allows for me to easily determine where problems in the code are and allows me to repeatedly call the function with little effort.

The function takes the 1st item in each of the players 'deck's and compares the colour (explained above) then if neither player wins from their colour alone, then each players number is compared. The variable whoWins is temporary (within the function) and allowed for debugging when determining problems with the function later.



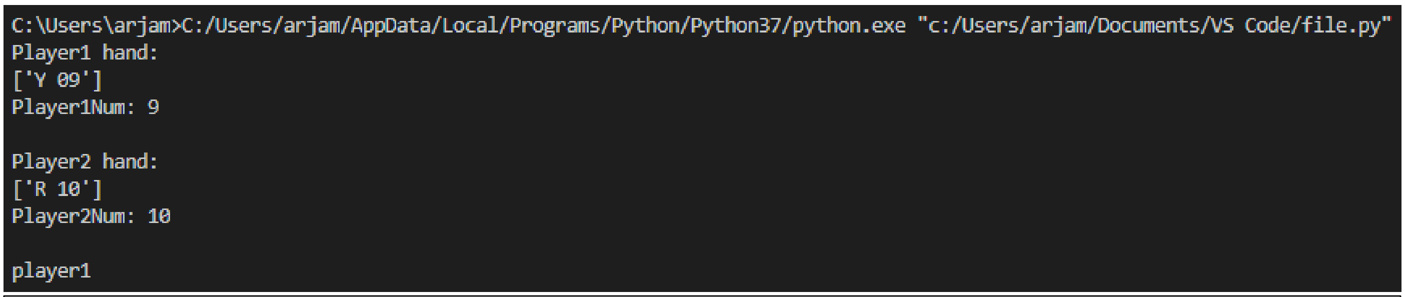
```python
###################------------------- Main Code Start ---------------------############################

random.shuffle(deck) #shuffles the deck
pick_up(Player1_Hand, deck) #calls the pick_up function
pick_up(Player2_Hand, deck)

player1Num = retrieveNumber(Player1_Hand) #finds the number in each of the players drawn cards
player2Num = retrieveNumber(Player2_Hand)

###################------------------- Debugging ---------------------###########################

print("Player1 hand: ")
print(Player1_Hand)
print("Player1Num: " + str(player1Num))
print()

print("Player2 hand: ")
print(Player2_Hand)
print("Player2Num: " + str(player2Num))

string = compare(Player1_Hand, Player2_Hand)
print()
print(string)
```

This is the main area of the program, everything above is where I declare functions and variables.

The code starts by shuffling the array 'deck' and calling the 'pick_up' function (explained below). Then the number on each of the players cards is retrieved (using the retrieveNumber function explained above).

Then some debugging takes place. This allowed me to check for any logic/syntax errors, of which one was found (below).

Finally, a variable 'string' (temporary name) is declared as the result of the function compare() (also highlighted above)

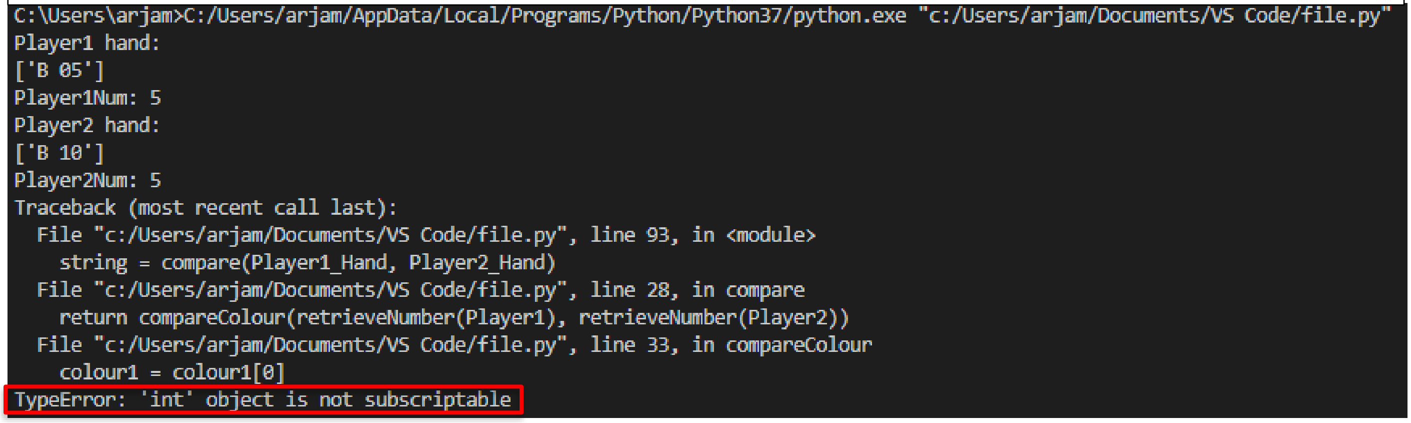


```
C:\Users\arjam>C:/Users/arjam/AppData/Local/Programs/Python/Python37/python.exe "c:/Users/arjam/Documents/VS Code/file.py"
Player1 hand:
['Y 09']
Player1Num: 9

Player2 hand:
['R 10']
Player2Num: 10

player1
```

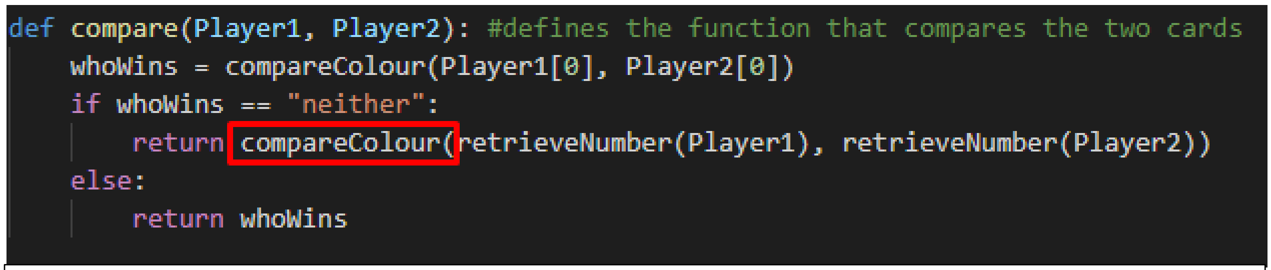
This is the output I got from a 'previous' version of the code (before I fixed the problem). As you can see, the program runs fine when comparing the colours of the two players' cards.

However when the two colours are the same and the numbers should be compared, an error occurs.

The error reads: 'int' object is not subscriptable as a TypeError.



```
C:\Users\arjam>C:/Users/arjam/AppData/Local/Programs/Python/Python37/python.exe "c:/Users/arjam/Documents/VS Code/file.py"
Player1 hand:
['B 05']
Player1Num: 5
Player2 hand:
['B 10']
Player2Num: 5
Traceback (most recent call last):
  File "c:/Users/arjam/Documents/VS Code/file.py", line 93, in <module>
    string = compare(Player1_Hand, Player2_Hand)
  File "c:/Users/arjam/Documents/VS Code/file.py", line 28, in compare
    return compareColour(retrieveNumber(Player1), retrieveNumber(Player2))
  File "c:/Users/arjam/Documents/VS Code/file.py", line 33, in compareColour
    colour1 = colour1[0]
TypeError: 'int' object is not subscriptable
```



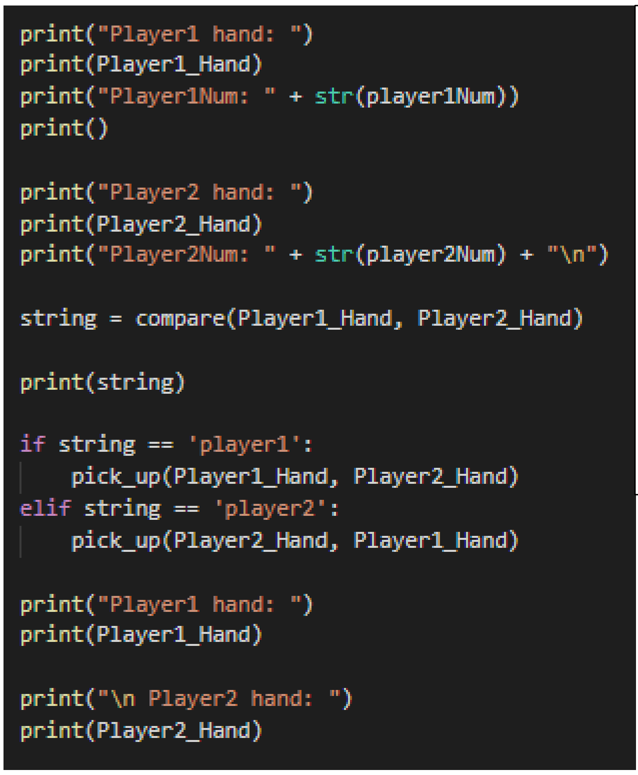
```python
def compare(Player1, Player2): #defines the function that compares the two cards
    whoWins = compareColour(Player1[0], Player2[0])
    if whoWins == "neither":
        return compareColour(retrieveNumber(Player1), retrieveNumber(Player2))
    else:
        return whoWins
```

Turns out, I had accidentally called the wrong function when meaning to compare the numbers. I proceeded to fix this and no more errors occurred (a good sign)

Day 2 of Python Coding Summary

The code now runs absolutely fine and it's clear that the program will complete the task with relative ease however I still need to make the 'main' part of the program repeats itself until all the cards in the deck have been depleted and improve the efficiency by changing the range of numbers from 1-10 to 0-9.

Day 3



```
print("Player1 hand: ")
print(Player1_Hand)
print("Player1Num: " + str(player1Num))
print()

print("Player2 hand: ")
print(Player2_Hand)
print("Player2Num: " + str(player2Num) + "\n")

string = compare(Player1_Hand, Player2_Hand)

print(string)

if string == 'player1':
    pick_up(Player1_Hand, Player2_Hand)
elif string == 'player2':
    pick_up(Player2_Hand, Player1_Hand)

print("Player1 hand: ")
print(Player1_Hand)

print("\n Player2 hand: ")
print(Player2_Hand)
```

The current code prints player 1's hand, then player1's number on the top card in their hand and does the same for player 2. This was only added for debugging

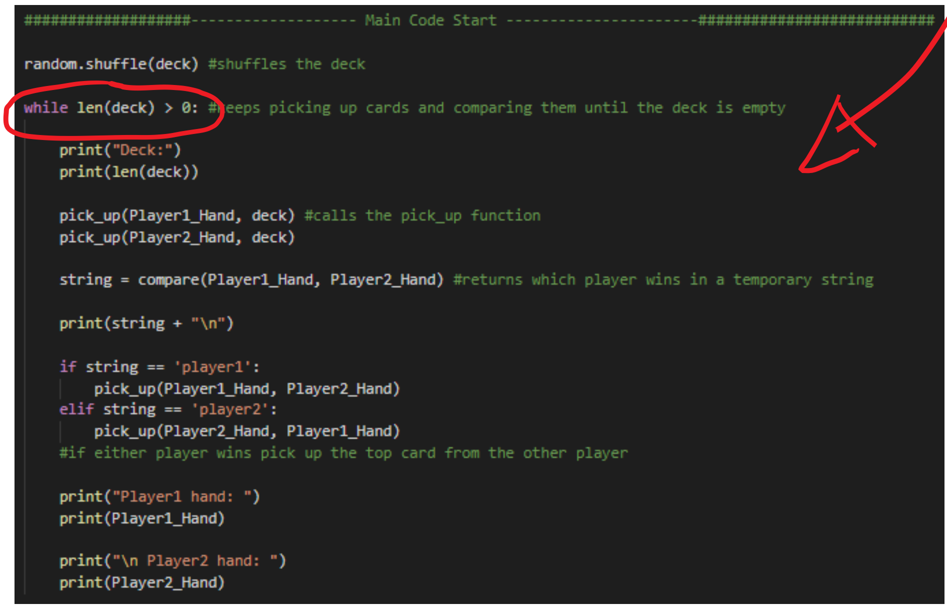
Then the two hands are compared ONCE and the outcome is outputted. I then used the 'pick_up' function I coded earlier to allow the winner to take the other players top card.

Then, for debugging, I outputted each player's hand.

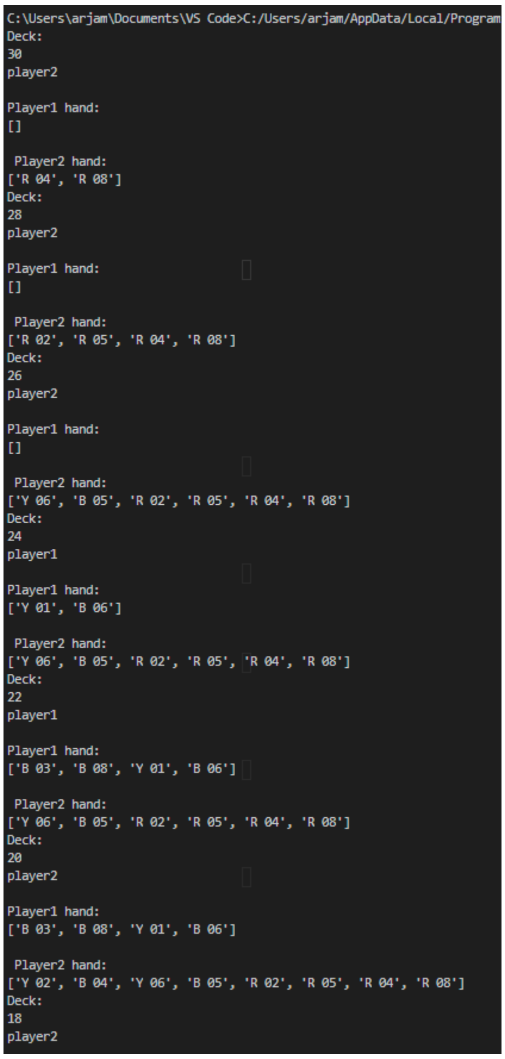
I ran this code and ran into no errors, this means the main function of the program was complete, now all I needed to do was repeat the process until the 'deck' was exhausted.

The repeating part I added was a while loop that looped continuously until the length of the 'deck' was 0. Then it would stop.

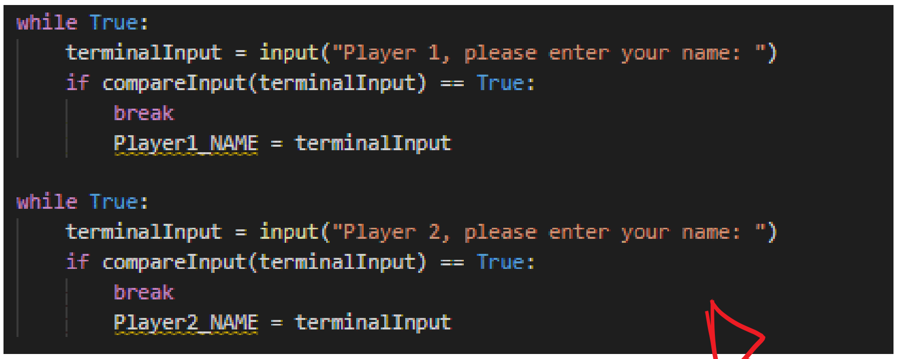
This was the output once the entire program had run through the while loop until the 'deck' was exhausted.



```
##################------------------ Main Code Start --------------------###########################

random.shuffle(deck) #shuffles the deck

while len(deck) > 0: #keeps picking up cards and comparing them until the deck is empty

    print("Deck:")
    print(len(deck))

    pick_up(Player1_Hand, deck) #calls the pick_up function
    pick_up(Player2_Hand, deck)

    string = compare(Player1_Hand, Player2_Hand) #returns which player wins in a temporary string

    print(string + "\n")

    if string == 'player1':
        pick_up(Player1_Hand, Player2_Hand)
    elif string == 'player2':
        pick_up(Player2_Hand, Player1_Hand)
    #if either player wins pick up the top card from the other player

    print("Player1 hand: ")
    print(Player1_Hand)

    print("\n Player2 hand: ")
    print(Player2_Hand)
```

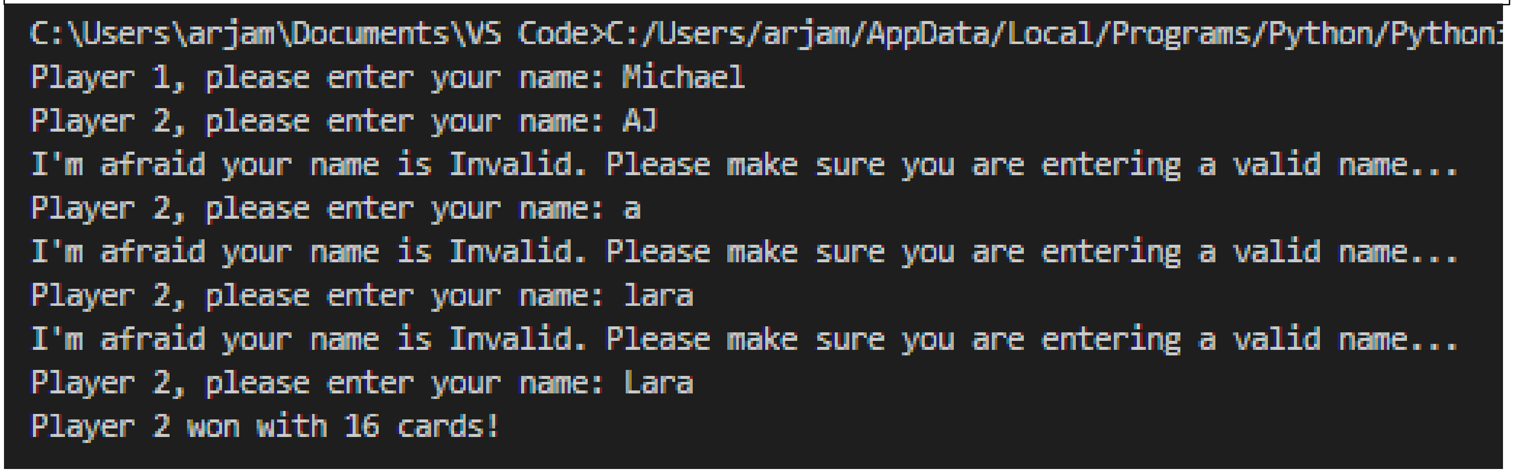


```
C:\Users\arjam\Documents\VS Code>C:/Users/arjam/AppData/Local/Program
Deck:
30
player2

Player1 hand:
[]

 Player2 hand:
['R 04', 'R 08']
Deck:
28
player2

Player1 hand:
[]

 Player2 hand:
['R 02', 'R 05', 'R 04', 'R 08']
Deck:
26
player2

Player1 hand:
[]

 Player2 hand:
['Y 06', 'B 05', 'R 02', 'R 05', 'R 04', 'R 08']
Deck:
24
player1

Player1 hand:
['Y 01', 'B 06']

 Player2 hand:
['Y 06', 'B 05', 'R 02', 'R 05', 'R 04', 'R 08']
Deck:
22
player1

Player1 hand:
['B 03', 'B 08', 'Y 01', 'B 06']

 Player2 hand:
['Y 06', 'B 05', 'R 02', 'R 05', 'R 04', 'R 08']
Deck:
20
player2

Player1 hand:
['B 03', 'B 08', 'Y 01', 'B 06']

 Player2 hand:
['Y 02', 'B 04', 'Y 06', 'B 05', 'R 02', 'R 05', 'R 04', 'R 08']
Deck:
18
player2
```



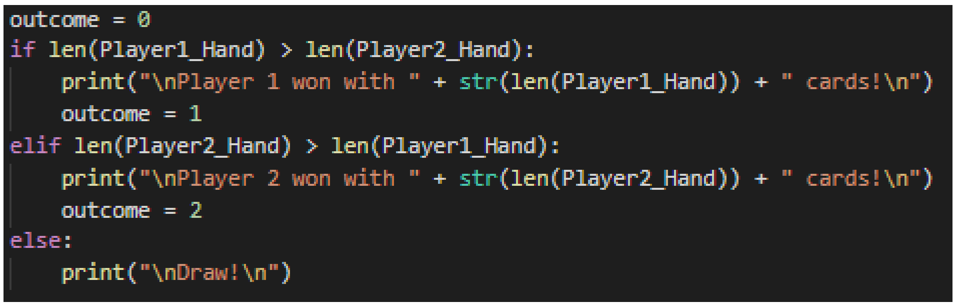
```
while True:
    terminalInput = input("Player 1, please enter your name: ")
    if compareInput(terminalInput) == True:
        break
        Player1_NAME = terminalInput

while True:
    terminalInput = input("Player 2, please enter your name: ")
    if compareInput(terminalInput) == True:
        break
        Player2_NAME = terminalInput
```

I then proceeded to change my initial player name input by using a repeated function. This meant that the user would have a better idea about which player had the invalid name. Since before, an error would only be returned if either name was invalid.



```
C:\Users\arjam\Documents\VS Code>C:/Users/arjam/AppData/Local/Programs/Python/Python
Player 1, please enter your name: Michael
Player 2, please enter your name: AJ
I'm afraid your name is Invalid. Please make sure you are entering a valid name...
Player 2, please enter your name: a
I'm afraid your name is Invalid. Please make sure you are entering a valid name...
Player 2, please enter your name: lara
I'm afraid your name is Invalid. Please make sure you are entering a valid name...
Player 2, please enter your name: Lara
Player 2 won with 16 cards!
```

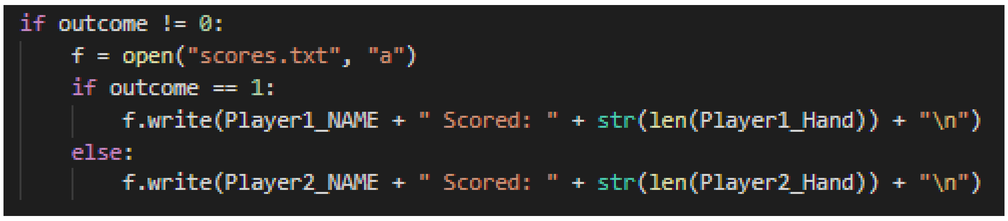
This code:



```python
outcome = 0
if len(Player1_Hand) > len(Player2_Hand):
    print("\nPlayer 1 won with " + str(len(Player1_Hand)) + " cards!\n")
    outcome = 1
elif len(Player2_Hand) > len(Player1_Hand):
    print("\nPlayer 2 won with " + str(len(Player2_Hand)) + " cards!\n")
    outcome = 2
else:
    print("\nDraw!\n")
```

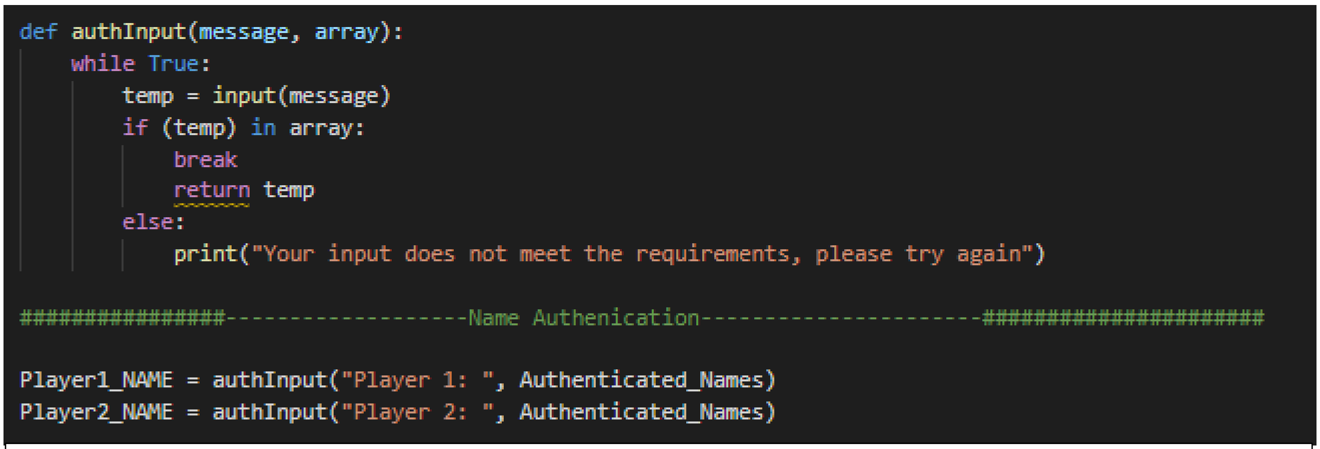
Outputs the winner and their card count.

Then using the variable ‘outcome’, I can store the winner’s player number in memory.



```python
if outcome != 0:
    f = open("scores.txt", "a")
    if outcome == 1:
        f.write(Player1_NAME + " Scored: " + str(len(Player1_Hand)) + "\n")
    else:
        f.write(Player2_NAME + " Scored: " + str(len(Player2_Hand)) + "\n")
```

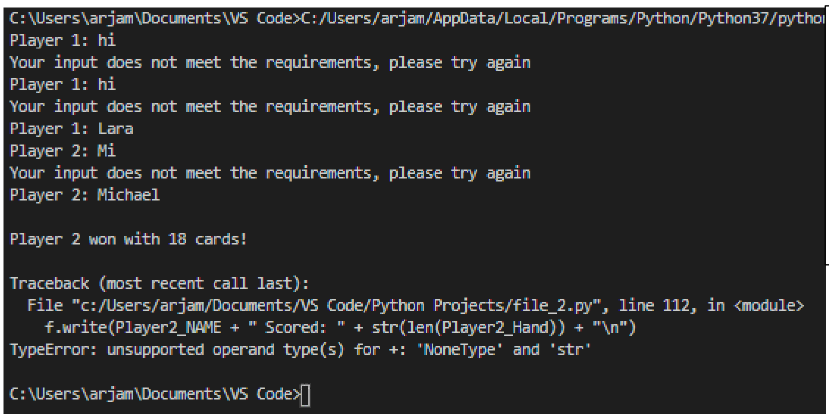
Then, if the outcome isn’t 0 (which is the draw condition) then the players name and score is written to a separate text document called ‘scores’



```python
def authInput(message, array):
    while True:
        temp = input(message)
        if (temp) in array:
            break
            return temp
        else:
            print("Your input does not meet the requirements, please try again")

################-------------------Name Authenication---------------------######################

Player1_NAME = authInput("Player 1: ", Authenticated_Names)
Player2_NAME = authInput("Player 2: ", Authenticated_Names)
```

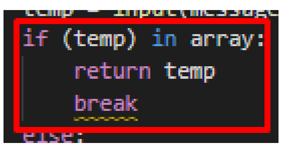
I decided to turn this into a re-useable function so I could use it later in the program, this function displays a message, takes an input and checks if the inputted string is in the array given when the function is called.

In the player name input, I used the previous array of ‘Authenticated_Names’.



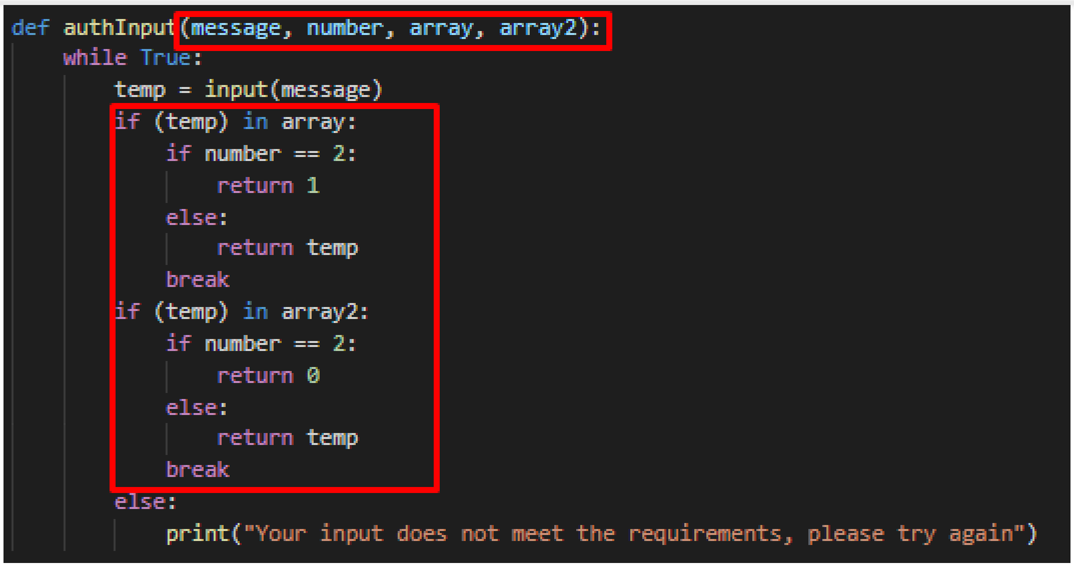
```
C:\Users\arjam\Documents\VS Code>C:/Users/arjam/AppData/Local/Programs/Python/Python37/pytho
Player 1: hi
Your input does not meet the requirements, please try again
Player 1: hi
Your input does not meet the requirements, please try again
Player 1: Lara
Player 2: Mi
Your input does not meet the requirements, please try again
Player 2: Michael

Player 2 won with 18 cards!

Traceback (most recent call last):
  File "c:/Users/arjam/Documents/VS Code/Python Projects/file_2.py", line 112, in <module>
    f.write(Player2_NAME + " Scored: " + str(len(Player2_Hand)) + "\n")
TypeError: unsupported operand type(s) for +: 'NoneType' and 'str'

C:\Users\arjam\Documents\VS Code>
```

However, when I ran this new code, I got a type error. I checked my code and realized that I hadn't returned the input until after I broke the while loop. This meant that nothing was being returned by the function and the Players Name was just 'null' without a type.



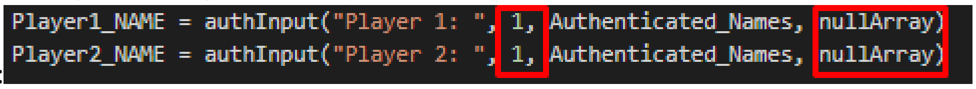
```
if (temp) in array:
    return temp
    break
```

I then needed to ask the user if they wanted to see the top 5 scorers. I wanted to use this authInput function to allow for multiple versions of yes and no, to allow for user error and preference. However, the function could only check if the input was in one array. So I decided to allow the function to take at least 2 possible arrays to be compared:



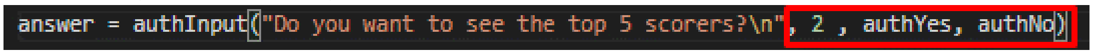
```
def authInput(message, number, array, array2):
    while True:
        temp = input(message)
        if (temp) in array:
            if number == 2:
                return 1
            else:
                return temp
            break
        if (temp) in array2:
            if number == 2:
                return 0
            else:
                return temp
            break
        else:
            print("Your input does not meet the requirements, please try again")
```

Then I changed the original auth input to use a nullArray and specify that they only wanted to compare with one array:



```
Player1_NAME = authInput("Player 1: ", 1, Authenticated_Names, nullArray)
Player2_NAME = authInput("Player 2: ", 1, Authenticated_Names, nullArray)
```

This was my 'Yes' or 'No' input:



```
answer = authInput("Do you want to see the top 5 scorers?\n", 2 , authYes, authNo)
```

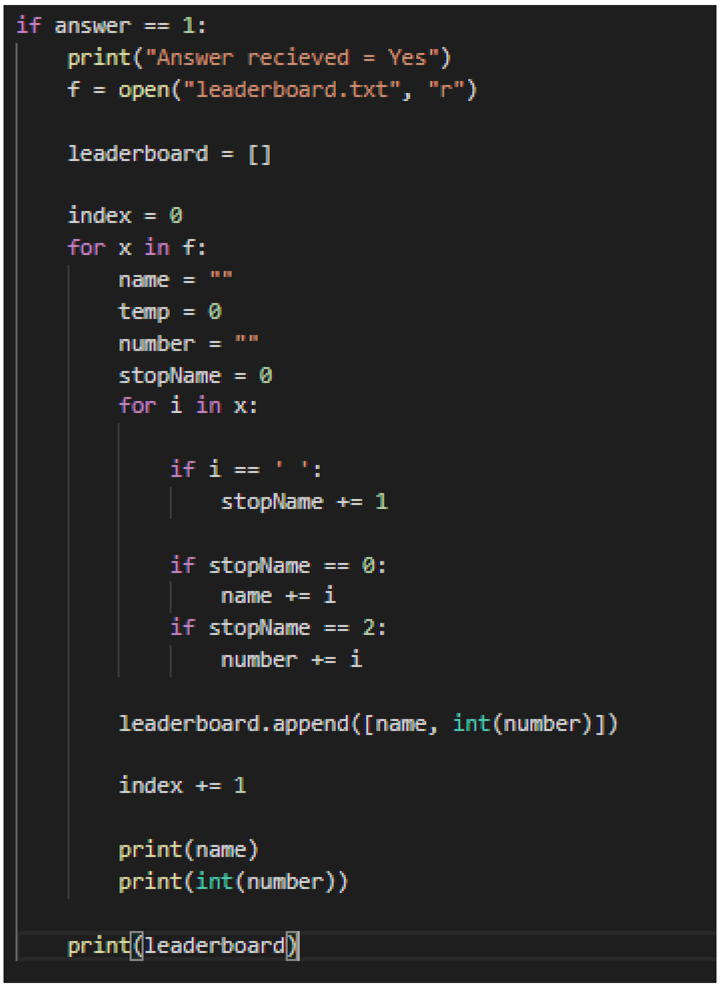
<u>Day 3 of Python Coding Summary</u>

I realise now that I should probably have separated the processes that:

- checks if the users input was correct
- asks if the input was yes or no

This part of the program is too long and complicated for what it's worth, and could be slightly easier to read and debug if I just created another function. I will probably fix this later on. However, the other areas work brilliantly and can be super-efficient!

Day 4



```python
if answer == 1:
    print("Answer recieved = Yes")
    f = open("leaderboard.txt", "r")

    leaderboard = []

    index = 0
    for x in f:
        name = ""
        temp = 0
        number = ""
        stopName = 0
        for i in x:

            if i == ' ':
                stopName += 1

            if stopName == 0:
                name += i
            if stopName == 2:
                number += i

        leaderboard.append([name, int(number)])

        index += 1

        print(name)
        print(int(number))

    print(leaderboard)
```

Next, I took action on the result of the 'Yes or No' input from earlier.

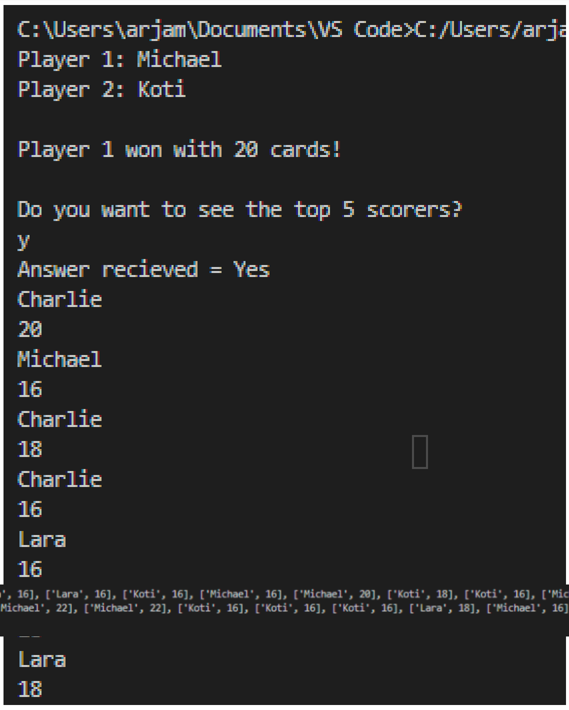
First I outputted that the computer had received the answer (mainly for debugging my previous messy functions and to accommodate for user error)

Then I opened the text document 'leaderboard' (same as scores.txt, just renamed) as read only.

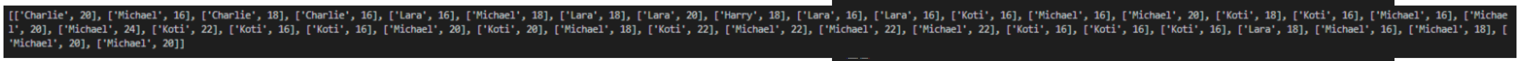
Then, I iterated through each line retrieving the name of the player on that line and their score. These values were then stored in an array called leaderboard.

For debugging, I outputted each name and number on the lines in the .txt document and finally outputted the dedicated array.

This was the output when run:

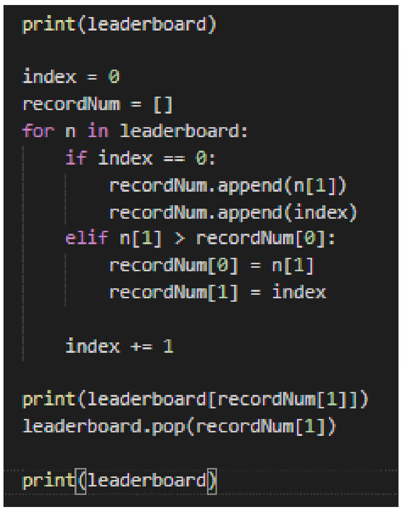


```
C:\Users\arjam\Documents\VS Code>C:/Users/arja
Player 1: Michael
Player 2: Koti

Player 1 won with 20 cards!

Do you want to see the top 5 scorers?
y
Answer recieved = Yes
Charlie
20
Michael
16
Charlie
18
Charlie
16
Lara
16

Lara
18
```



```
[['Charlie', 20], ['Michael', 16], ['Charlie', 18], ['Charlie', 16], ['Lara', 16], ['Michael', 18], ['Lara', 18], ['Lara', 20], ['Harry', 18], ['Lara', 16], ['Lara', 16], ['Koti', 16], ['Michael', 16], ['Michael', 20], ['Koti', 18], ['Koti', 16], ['Michael', 16], ['Michae
l', 20], ['Michael', 24], ['Koti', 22], ['Koti', 16], ['Koti', 16], ['Michael', 20], ['Koti', 20], ['Michael', 18], ['Koti', 22], ['Michael', 22], ['Michael', 22], ['Michael', 22], ['Koti', 16], ['Koti', 16], ['Koti', 16], ['Lara', 18], ['Michael', 16], ['Michael', 18], [
'Michael', 20], ['Michael', 20]]
```

Leaderboard Array:

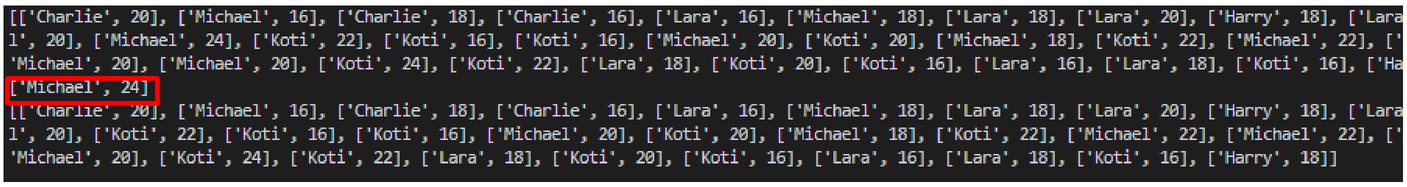


```python
print(leaderboard)

index = 0
recordNum = []
for n in leaderboard:
    if index == 0:
        recordNum.append(n[1])
        recordNum.append(index)
    elif n[1] > recordNum[0]:
        recordNum[0] = n[1]
        recordNum[1] = index

    index += 1

print(leaderboard[recordNum[1]])
leaderboard.pop(recordNum[1])

print(leaderboard)
```

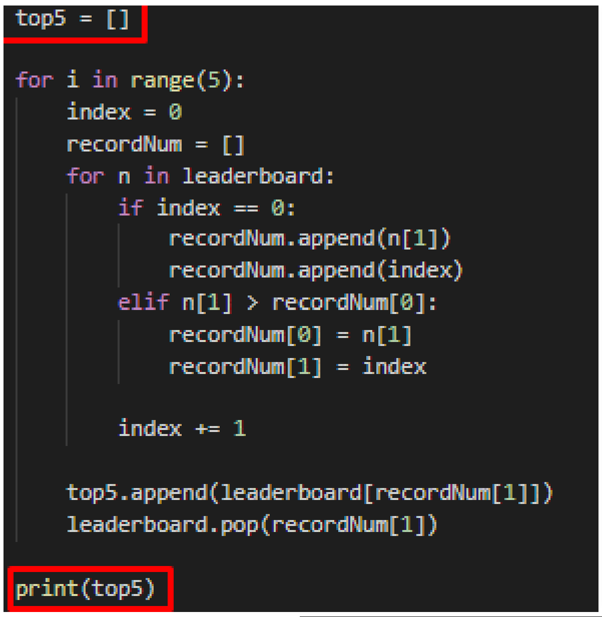
Everything checked out so I moved onwards.

The program then looks at the first item in the leaderboard array and adds it to the array 'recordNum' the number contained in this array is then compared with every other number in the leaderboard array. If a larger number in the leaderboard array is found, then the name and number in recordNum is replaced. Once this process has finished, the highest score in the leaderboard array is found, stored in recordNum and removed from the leaderboard array.

This is the output of this code:



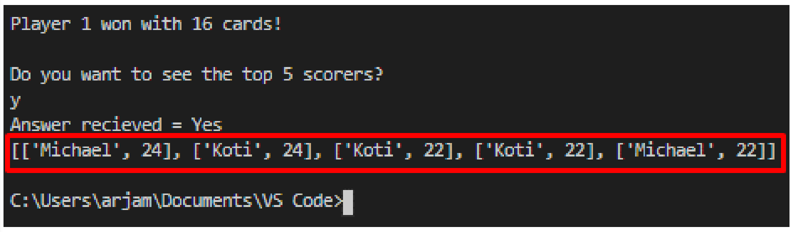
```
[['Charlie', 20], ['Michael', 16], ['Charlie', 18], ['Charlie', 16], ['Lara', 16], ['Michael', 18], ['Lara', 18], ['Lara', 20], ['Harry', 18], ['Lara
l', 20], ['Michael', 24], ['Koti', 22], ['Koti', 16], ['Koti', 16], ['Michael', 20], ['Koti', 20], ['Michael', 18], ['Koti', 22], ['Michael', 22], ['
'Michael', 20], ['Michael', 20], ['Koti', 24], ['Koti', 22], ['Lara', 18], ['Koti', 20], ['Koti', 16], ['Lara', 16], ['Lara', 18], ['Koti', 16], ['Ha
['Michael', 24]
[['Charlie', 20], ['Michael', 16], ['Charlie', 18], ['Charlie', 16], ['Lara', 16], ['Michael', 18], ['Lara', 18], ['Lara', 20], ['Harry', 18], ['Lara
l', 20], ['Koti', 22], ['Koti', 16], ['Koti', 16], ['Michael', 20], ['Koti', 20], ['Michael', 18], ['Koti', 22], ['Michael', 22], ['Michael', 22], ['
'Michael', 20], ['Koti', 24], ['Koti', 22], ['Lara', 18], ['Koti', 20], ['Koti', 16], ['Lara', 16], ['Lara', 18], ['Koti', 16], ['Harry', 18]]
```



```python
top5 = []

for i in range(5):
    index = 0
    recordNum = []
    for n in leaderboard:
        if index == 0:
            recordNum.append(n[1])
            recordNum.append(index)
        elif n[1] > recordNum[0]:
            recordNum[0] = n[1]
            recordNum[1] = index

        index += 1

    top5.append(leaderboard[recordNum[1]])
    leaderboard.pop(recordNum[1])

print(top5)
```

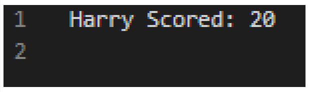
This can then be repeated 5 times so that the array 'top5' contains the top 5 scorers in the leaderboard array

This is the output:



```
Player 1 won with 16 cards!

Do you want to see the top 5 scorers?
y
Answer recieved = Yes
[['Michael', 24], ['Koti', 24], ['Koti', 22], ['Koti', 22], ['Michael', 22]]

C:\Users\arjam\Documents\VS Code>
```

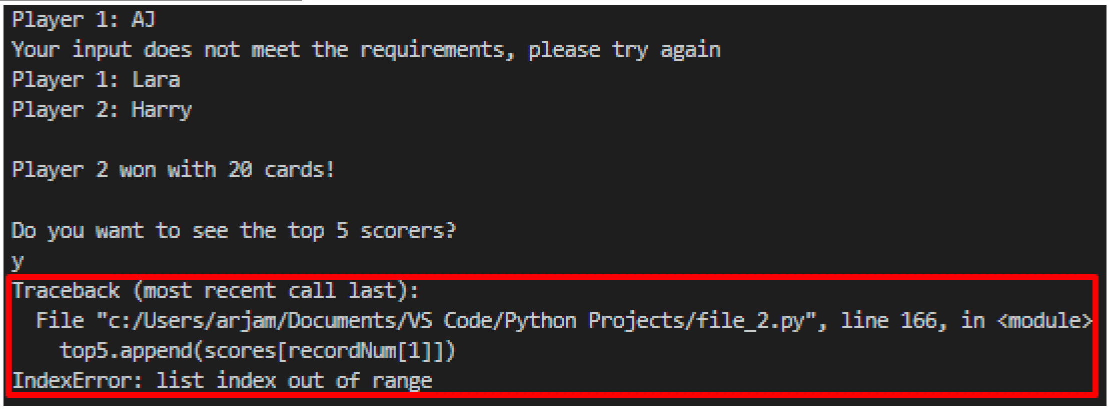
This code had a fatal flaw however. What happens when the leaderboard has less than 5 scores in total? Like this:



```
Harry Scored: 20
```

I decided to see what would happen when I tried this.

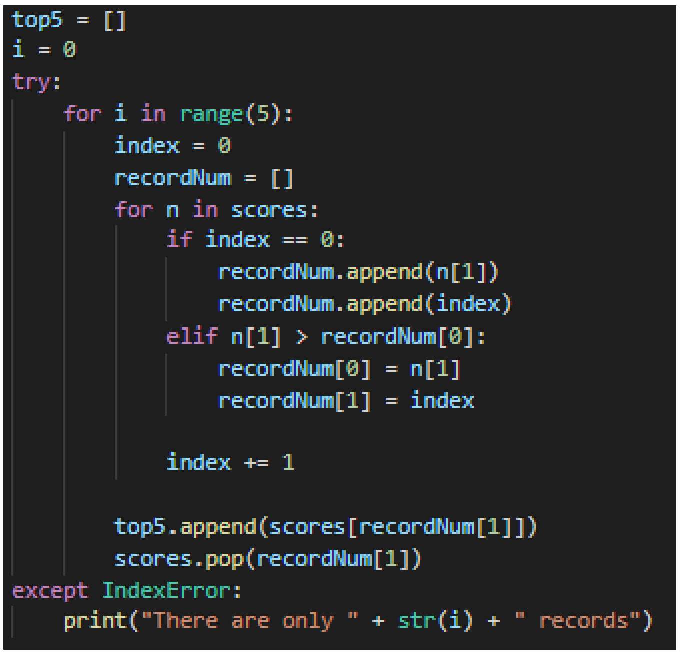
This is what the program returned:



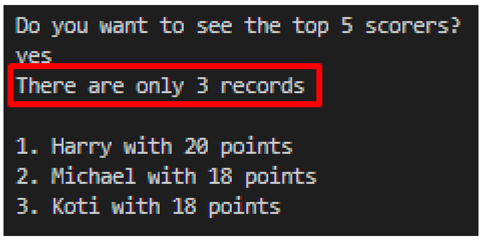
```
Player 1: AJ
Your input does not meet the requirements, please try again
Player 1: Lara
Player 2: Harry

Player 2 won with 20 cards!

Do you want to see the top 5 scorers?
y
Traceback (most recent call last):
  File "c:/Users/arjam/Documents/VS Code/Python Projects/file_2.py", line 166, in <module>
    top5.append(scores[recordNum[1]])
IndexError: list index out of range
```

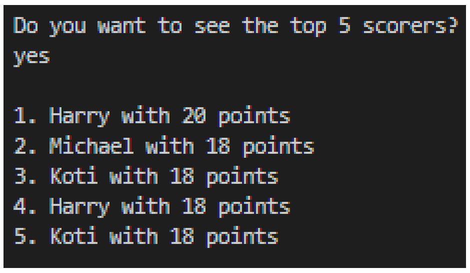
Day 5

In order to combat this, I made some changes to the code:


```
top5 = []
i = 0
try:
    for i in range(5):
        index = 0
        recordNum = []
        for n in scores:
            if index == 0:
                recordNum.append(n[1])
                recordNum.append(index)
            elif n[1] > recordNum[0]:
                recordNum[0] = n[1]
                recordNum[1] = index

            index += 1

        top5.append(scores[recordNum[1]])
        scores.pop(recordNum[1])
except IndexError:
    print("There are only " + str(i) + " records")
```

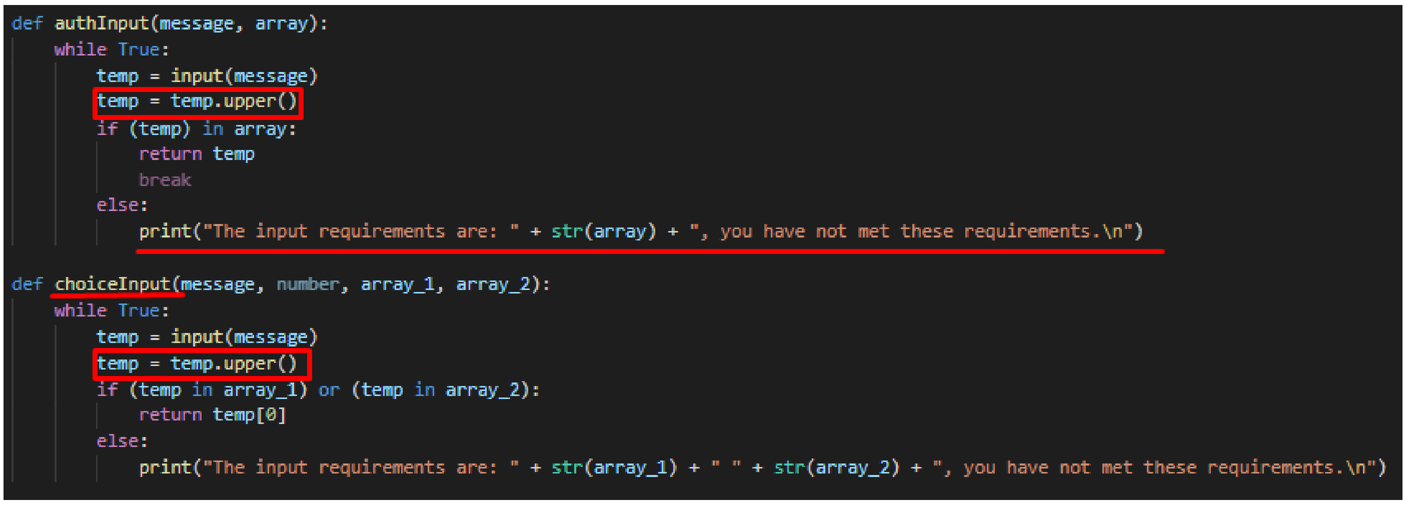
I added an error exception, this meant that the program would attempt to retrieve 5 of the top numbers in the score.txt file. But if at any point an IndexError occurred (when the file read didn't contain enough data to output the top 5 scorers) the program would stop, output to the user the number of records it could find and subsequently output those records.


```
Do you want to see the top 5 scorers?
yes
There are only 3 records

1. Harry with 20 points
2. Michael with 18 points
3. Koti with 18 points
```


```
Do you want to see the top 5 scorers?
yes

1. Harry with 20 points
2. Michael with 18 points
3. Koti with 18 points
4. Harry with 18 points
5. Koti with 18 points
```

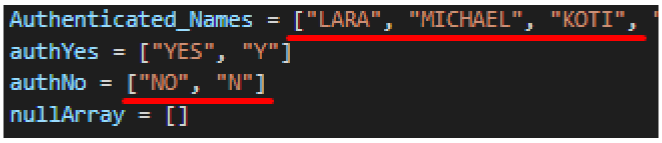
Just to test, I also made sure that this 'error' message didn't show when there was enough data. Sure enough, this was no issue.

IMPROVED INPUT FUNCTIONS


```
def authInput(message, array):
    while True:
        temp = input(message)
        temp = temp.upper()
        if (temp) in array:
            return temp
            break
        else:
            print("The input requirements are: " + str(array) + ", you have not met these requirements.\n")

def choiceInput(message, number, array_1, array_2):
    while True:
        temp = input(message)
        temp = temp.upper()
        if (temp in array_1) or (temp in array_2):
            return temp[0]
        else:
            print("The input requirements are: " + str(array_1) + " " + str(array_2) + ", you have not met these requirements.\n")
```

I changed my authInput function. I realised that the input was case-sensitive and that users might find this difficult to work around in order to use my program. I also didn't like how I merged two very different functions into one.

I split the function in two, and in both I decided to convert any user input into uppercase (temp.upper() does this) then they both do roughly the same thing (they check if the user input is in the given array/s and returns a value based off that. If this is not the case then print an error message and repeat)

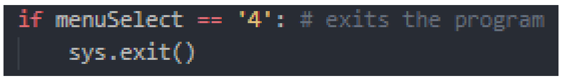

```
Authenticated_Names = ["LARA", "MICHAEL", "KOTI", '
authYes = ["YES", "Y"]
authNo = ["NO", "N"]
nullArray = []
```

I do slightly tweak this later to add some security to the authInput functions and I realized that I could probably have taken advantage of Pythons support of function overloading and used the same function name twice.

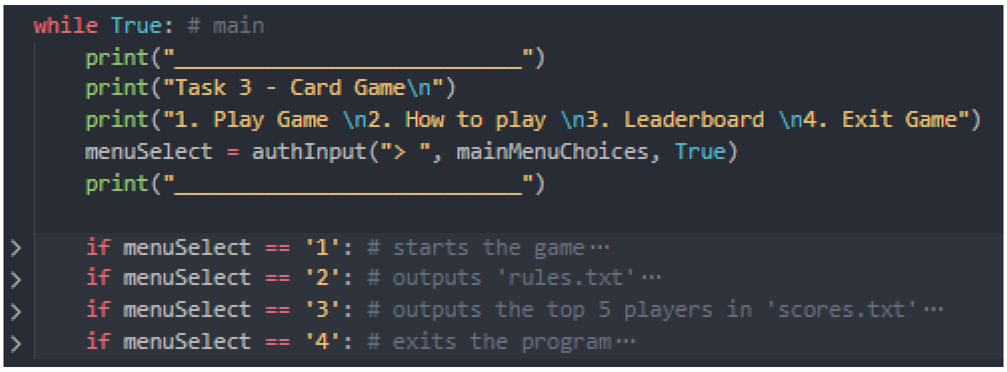
Day 6

MENU SYSTEM

I wanted to implement a menu system to the program to improve the user interface and add some level of polish to the program. The program uses a while loop so the only way to exit the entire program is to select "Exit Game" now.



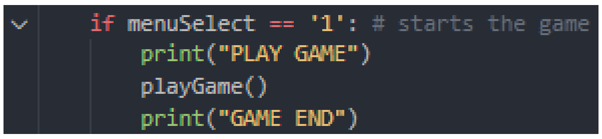
```
if menuSelect == '4': # exits the program
    sys.exit()
```



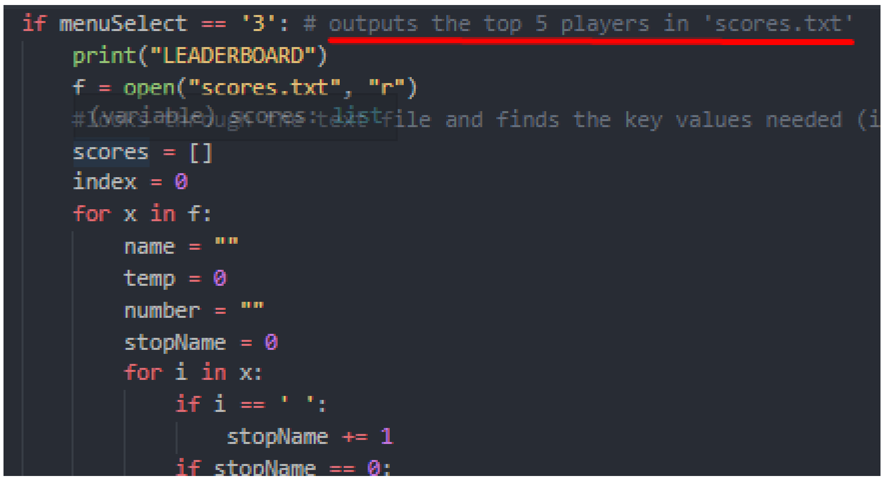
```
while True: # main
    print("______________________________")
    print("Task 3 - Card Game\n")
    print("1. Play Game \n2. How to play \n3. Leaderboard \n4. Exit Game")
    menuSelect = authInput("> ", mainMenuChoices, True)
    print("______________________________")

>   if menuSelect == '1': # starts the game···
>   if menuSelect == '2': # outputs 'rules.txt'···
>   if menuSelect == '3': # outputs the top 5 players in 'scores.txt'···
>   if menuSelect == '4': # exits the program···
```

Visual Studio Code has this useful feature which allows for the collapsing of some regions of code, so if the screenshot looks weird, it's because the code is collapsed under these if statements. The comments explain the code contained as well.

I moved the game into the function playGame() for convenience and added markers to show the user when the game had started and ended. This menu system also adds re-playability.



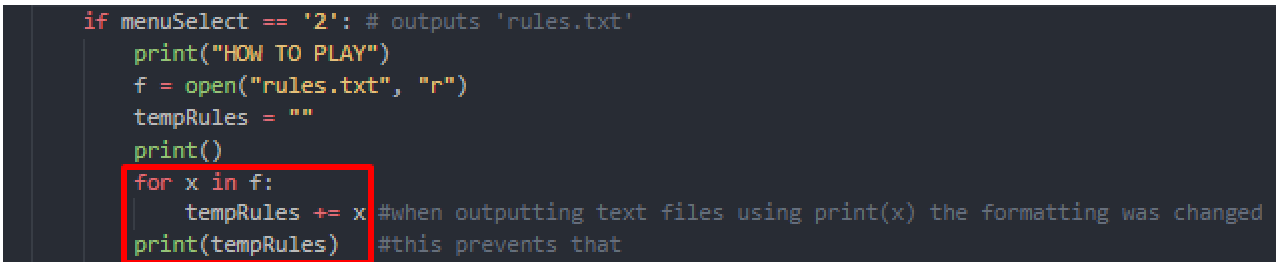
```
    if menuSelect == '1': # starts the game
        print("PLAY GAME")
        playGame()
        print("GAME END")
```



```
if menuSelect == '3': # outputs the top 5 players in 'scores.txt'
    print("LEADERBOARD")
    f = open("scores.txt", "r")
    #looks through the text file and finds the key values needed (i
    scores = []
    index = 0
    for x in f:
        name = ""
        temp = 0
        number = ""
        stopName = 0
        for i in x:
            if i == ' ':
                stopName += 1
            if stopName == 0:
```

I also moved the score output function into the menu under "leaderboard". Now, instead of only outputting the top 5 players at the end of the program, the user can ask for the top 5 scorers whenever they want!

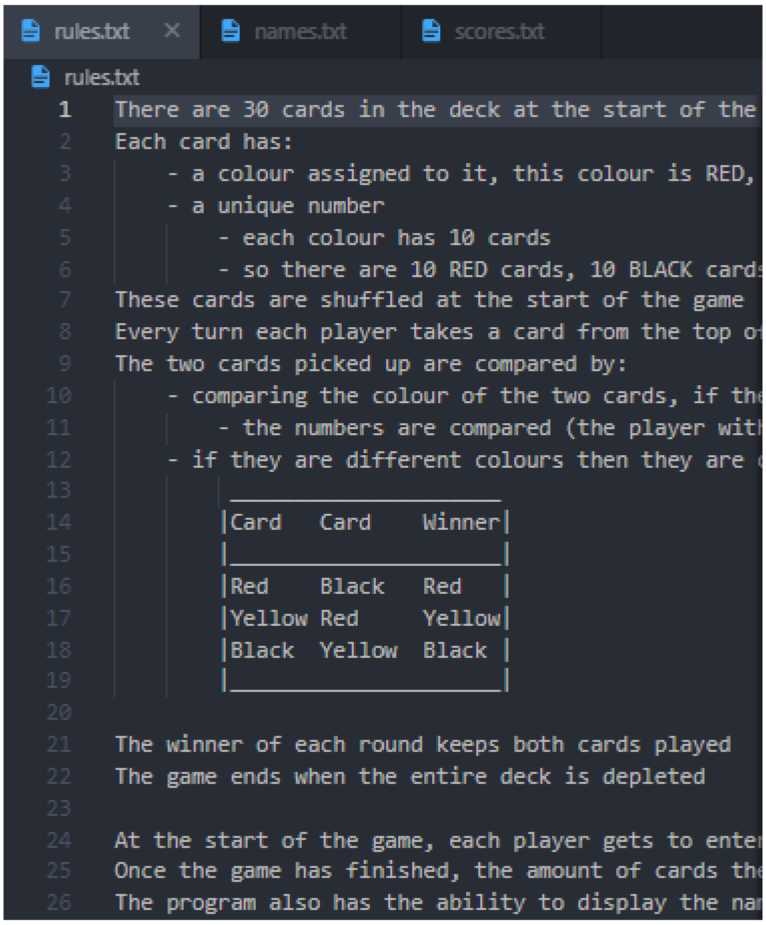
I also added a 'rule book' which outputs the rules of the game when selected in the menu.



```
    if menuSelect == '2': # outputs 'rules.txt'
        print("HOW TO PLAY")
        f = open("rules.txt", "r")
        tempRules = ""
        print()
        for x in f:
            tempRules += x #when outputting text files using print(x) the formatting was changed
        print(tempRules)   #this prevents that
```

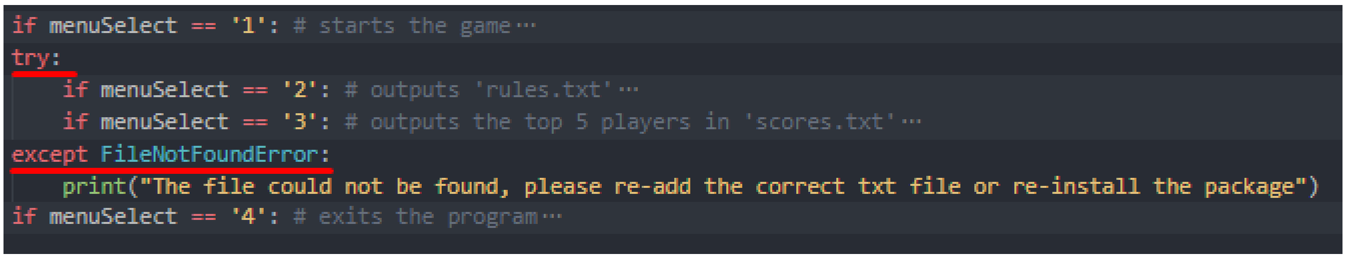
The code outlined was my way of ensuring python would output the text file exactly how it is written. The normal way python outputted it added weird indentation in some cases.

This is part of the rule book:



```
rules.txt    names.txt    scores.txt
rules.txt
There are 30 cards in the deck at the start of the
Each card has:
    - a colour assigned to it, this colour is RED,
    - a unique number
        - each colour has 10 cards
        - so there are 10 RED cards, 10 BLACK cards
These cards are shuffled at the start of the game
Every turn each player takes a card from the top of
The two cards picked up are compared by:
    - comparing the colour of the two cards, if the
        - the numbers are compared (the player with
    - if they are different colours then they are c

        ______________________
        |Card    Card     Winner|
        |______________________|
        |Red     Black    Red   |
        |Yellow Red      Yellow|
        |Black  Yellow   Black |
        |______________________|

The winner of each round keeps both cards played
The game ends when the entire deck is depleted

At the start of the game, each player gets to enter
Once the game has finished, the amount of cards the
The program also has the ability to display the nam
```

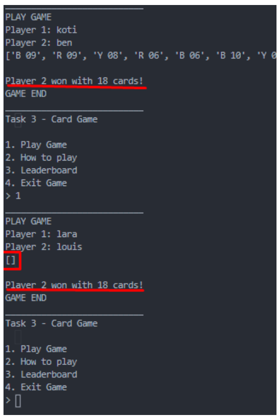
The program also spits out some errors when it cannot find the files needed, so I added some error detection



```python
if menuSelect == '1': # starts the game···
try:
    if menuSelect == '2': # outputs 'rules.txt'···
    if menuSelect == '3': # outputs the top 5 players in 'scores.txt'···
except FileNotFoundError:
    print("The file could not be found, please re-add the correct txt file or re-install the package")
if menuSelect == '4': # exits the program···
```

Which outputs a more civilised message and doesn't end the program.

MENU SYSTEM PROBLEMS



```
PLAY GAME
Player 1: koti
Player 2: ben
['B 09', 'R 09', 'Y 08', 'R 06', 'B 06', 'B 10', 'Y 0

Player 2 won with 18 cards!
GAME END

Task 3 - Card Game

1. Play Game
2. How to play
3. Leaderboard
4. Exit Game
> 1

PLAY GAME
Player 1: lara
Player 2: louis
[]

Player 2 won with 18 cards!
GAME END

Task 3 - Card Game

1. Play Game
2. How to play
3. Leaderboard
4. Exit Game
>
```

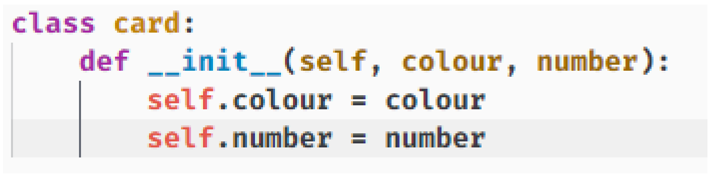
The re-playability of the main menu introduced a Logic Error which meant that the final winner was repeated over and over again

I realized this was because I forgot to reset the deck every time you start a game.

OBJECT ORIENTED PROGRAMMING

This next section outlines how I went about converting some elements of the project into an Object Oriented Program (a “technique” which I learnt which will make the whole project run more effectively)

I started off by creating a simple class in a separate python file here:

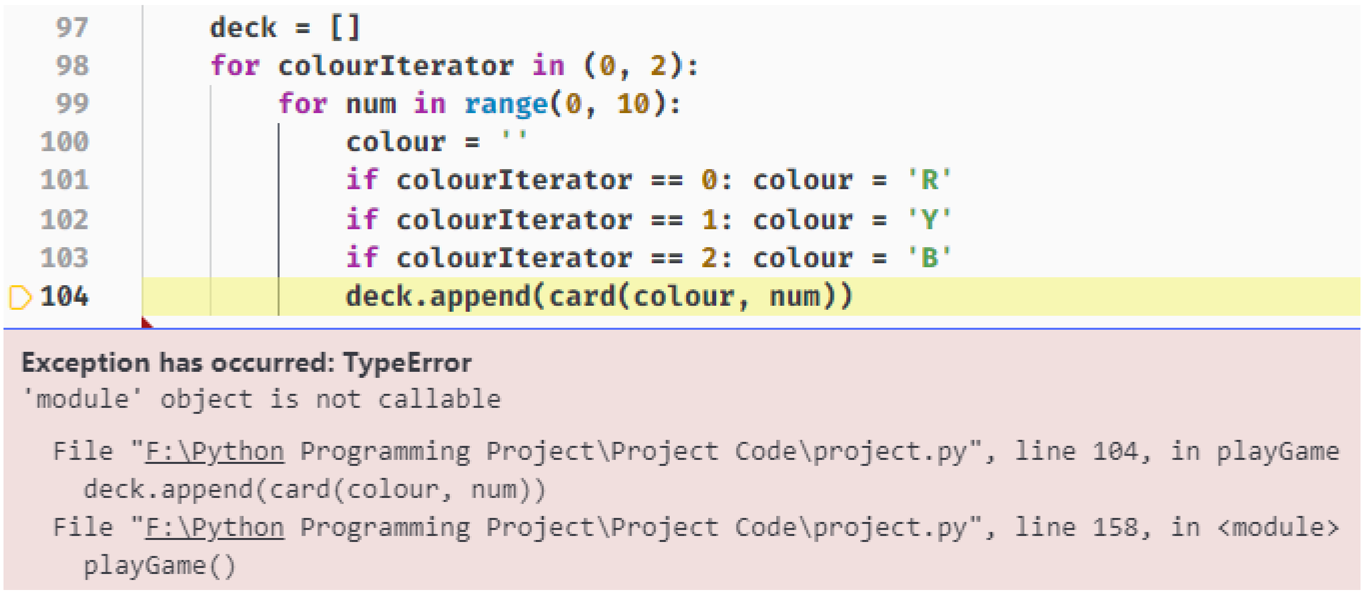


```
class card:
    def __init__(self, colour, number):
        self.colour = colour
        self.number = number
```

This is a simple class at the moment and will allow me to convert the deck from an array of strings to an array of card objects.

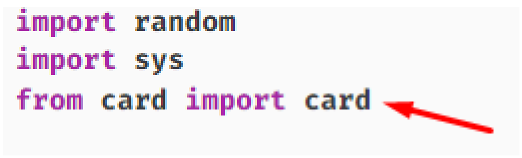
I plan to convert all the current arrays and variables into classes to make managing code easier.

I toyed with the idea of turning the entire deck into a class, but I thought that keeping the current python array would work just fine.



```
 97     deck = []
 98     for colourIterator in (0, 2):
 99         for num in range(0, 10):
100             colour = ''
101             if colourIterator == 0: colour = 'R'
102             if colourIterator == 1: colour = 'Y'
103             if colourIterator == 2: colour = 'B'
104             deck.append(card(colour, num))
```

```
Exception has occurred: TypeError
'module' object is not callable
  File "F:\Python Programming Project\Project Code\project.py", line 104, in playGame
    deck.append(card(colour, num))
  File "F:\Python Programming Project\Project Code\project.py", line 158, in <module>
    playGame()
```

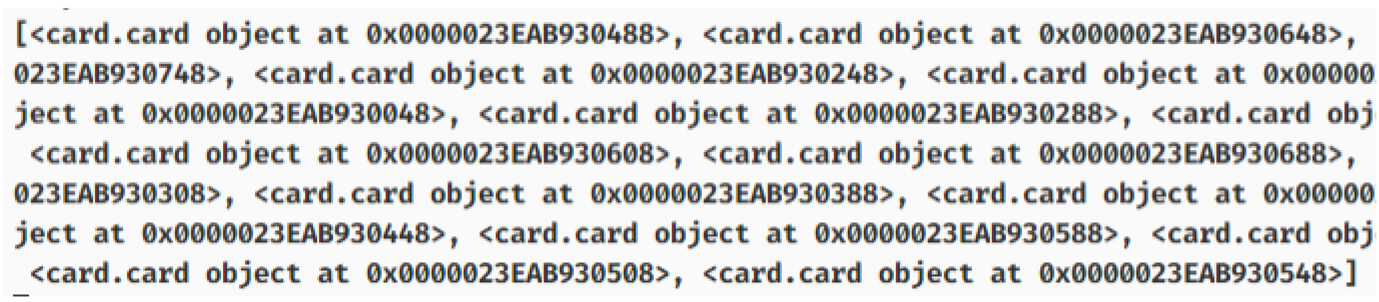


```
import random
import sys
from card import card
```

This error occurred when I tried initializing the array of card objects.

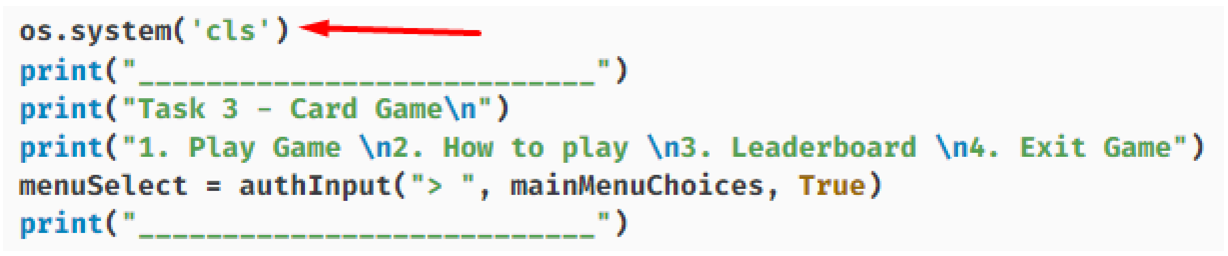
My problem was that I created my class in a different python file and forgot to import the class into the main python file.

Now when I print the deck, it outputs this:



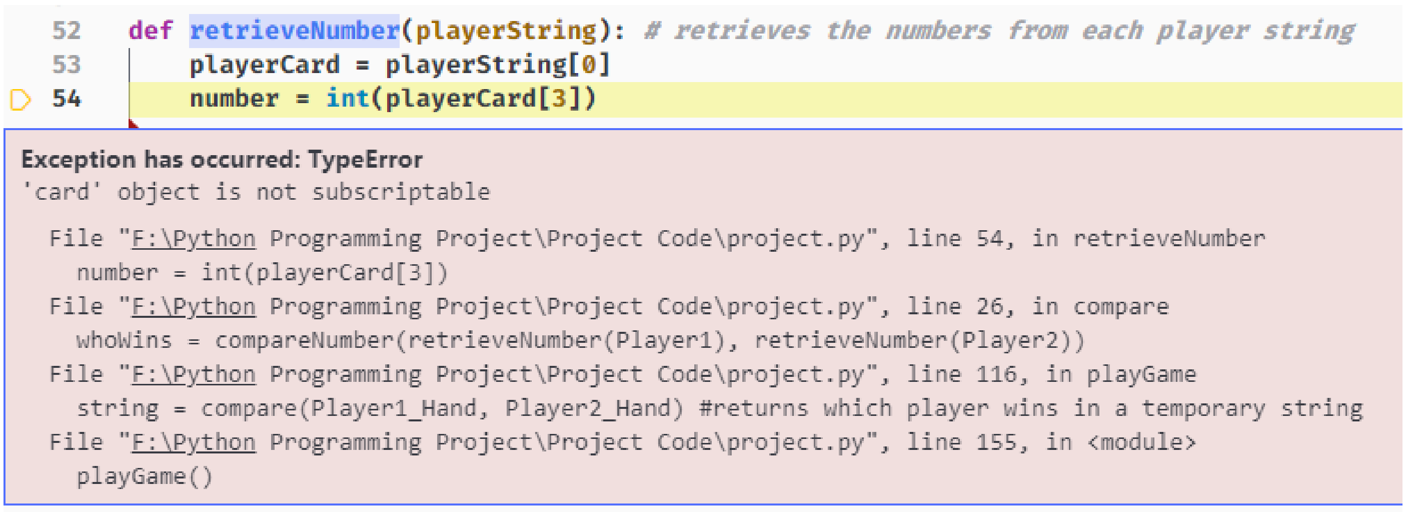
```
[<card.card object at 0x0000023EAB930488>, <card.card object at 0x0000023EAB930648>,
023EAB930748>, <card.card object at 0x0000023EAB930248>, <card.card object at 0x00000
ject at 0x0000023EAB930048>, <card.card object at 0x0000023EAB930288>, <card.card obj
 <card.card object at 0x0000023EAB930608>, <card.card object at 0x0000023EAB930688>,
023EAB930308>, <card.card object at 0x0000023EAB930388>, <card.card object at 0x00000
ject at 0x0000023EAB930448>, <card.card object at 0x0000023EAB930588>, <card.card obj
 <card.card object at 0x0000023EAB930508>, <card.card object at 0x0000023EAB930548>]
```

Which is what I expect! It shows that the deck is filled with card objects at different points in memory, which shows that the class system works. Now I need to make sure the rest of the program handles this change.



```
os.system('cls')
print("____________________________")
print("Task 3 - Card Game\n")
print("1. Play Game \n2. How to play \n3. Leaderboard \n4. Exit Game")
menuSelect = authInput("> ", mainMenuChoices, True)
print("____________________________")
```

I've used os.system("cls") to clear the python console at certain points in my menu system to improve overall clarity



```
def retrieveNumber(playerString): # retrieves the numbers from each player string
    playerCard = playerString[0]
    number = int(playerCard[3])
```

```
Exception has occurred: TypeError
'card' object is not subscriptable
  File "F:\Python Programming Project\Project Code\project.py", line 54, in retrieveNumber
    number = int(playerCard[3])
  File "F:\Python Programming Project\Project Code\project.py", line 26, in compare
    whoWins = compareNumber(retrieveNumber(Player1), retrieveNumber(Player2))
  File "F:\Python Programming Project\Project Code\project.py", line 116, in playGame
    string = compare(Player1_Hand, Player2_Hand) #returns which player wins in a temporary string
  File "F:\Python Programming Project\Project Code\project.py", line 155, in <module>
    playGame()
```

Another error, this was because I hadn't changed my functions to work with this new format and the retrieveNumber function was getting confused

All I need to do is fix this:

```
whoWins = compareNumber(retrieveNumber(Player1), retrieveNumber(Player2))
```

By turning it into this:

```
whoWins = compareNumber(Player1[0].number, Player1[0].number)
```

Which means I can also get rid of the retrieveNumber function and replace the compareNumber function with python's ternary operator

```
whoWins = "player1" if (Player1[0].number > Player2[0].number) else "player2"
```

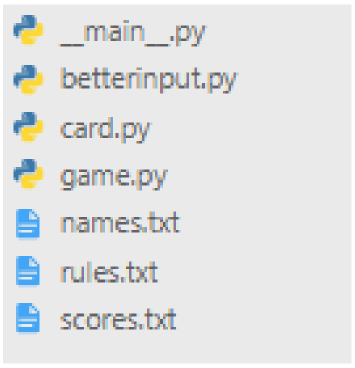
I also moved the authenticated names to a text file and added error handling to ensure the game wouldn't run without the text file.

```
Authenticated_Names = []
try:
    n = open("names.txt", "r")
    for l in n:
        Authenticated_Names.append(str.rstrip(l)) # str.rstrip() removes the trailing newline
except FileNotFoundError:
    print("File 'names.txt' was not found, the game cannot run. Please re-add the correct txt file or re-install the package")
    return 0
```

Now this all works perfectly and I have improved my code readability and efficiency!

File Organisation

Next I decided to learn how to separate the program into separate python files for better organization:



```
__main__.py
betterinput.py
card.py
game.py
names.txt
rules.txt
scores.txt
```

card.py – contains the card class

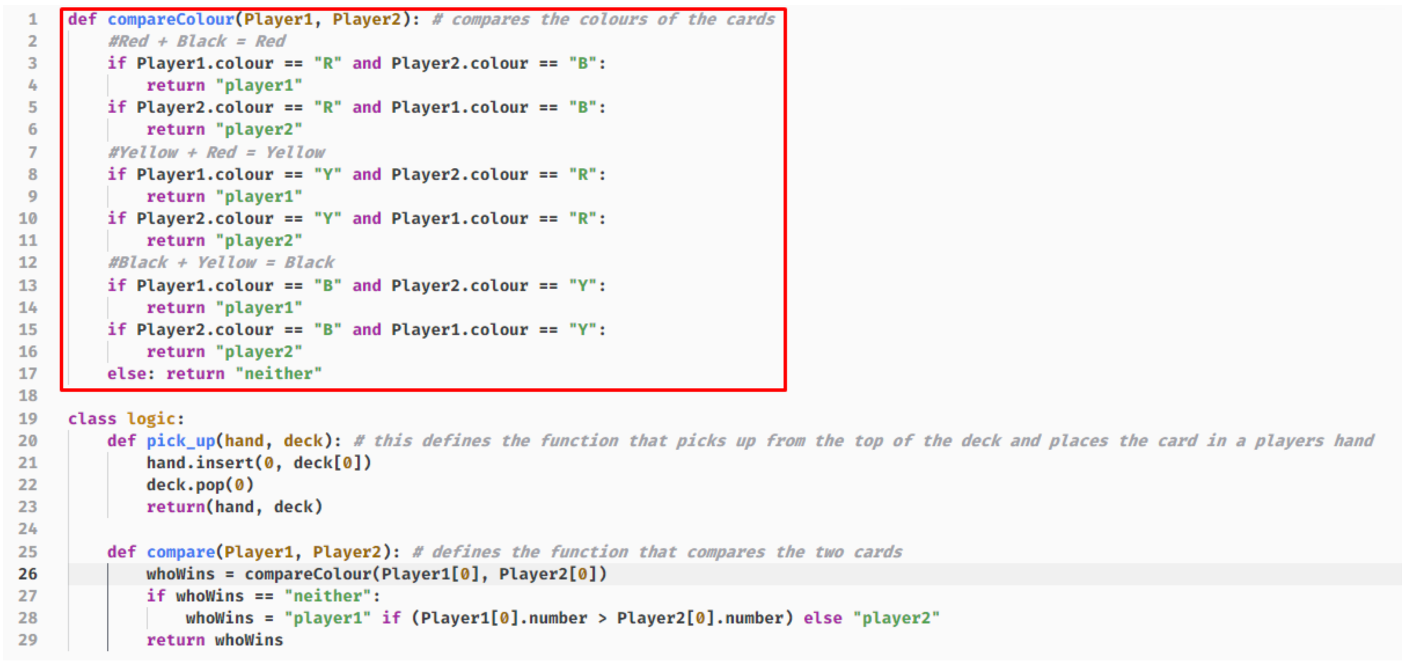
game.py – is where the code for the game resides

betterinput.py – is where the input functions I have created go

main.py – imports game.py and input capture and at the moment includes the main menu system

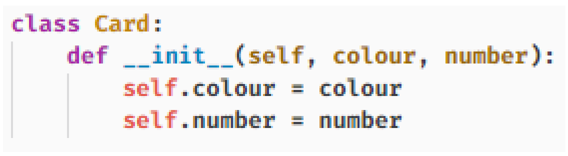
card.py

Contains the logic for comparing and managing the cards, I layed this out by creating a class called logic which contains the functions used by other python files (is imported in 'game.py') and then defining the function compareColour separately. This means that it isn't imported unnecessarily into other python files.



```python
def compareColour(Player1, Player2): # compares the colours of the cards
    #Red + Black = Red
    if Player1.colour == "R" and Player2.colour == "B":
        return "player1"
    if Player2.colour == "R" and Player1.colour == "B":
        return "player2"
    #Yellow + Red = Yellow
    if Player1.colour == "Y" and Player2.colour == "R":
        return "player1"
    if Player2.colour == "Y" and Player1.colour == "R":
        return "player2"
    #Black + Yellow = Black
    if Player1.colour == "B" and Player2.colour == "Y":
        return "player1"
    if Player2.colour == "B" and Player1.colour == "Y":
        return "player2"
    else: return "neither"

class logic:
    def pick_up(hand, deck): # this defines the function that picks up from the top of the deck and places the card in a players hand
        hand.insert(0, deck[0])
        deck.pop(0)
        return(hand, deck)

    def compare(Player1, Player2): # defines the function that compares the two cards
        whoWins = compareColour(Player1[0], Player2[0])
        if whoWins == "neither":
            whoWins = "player1" if (Player1[0].number > Player2[0].number) else "player2"
        return whoWins
```

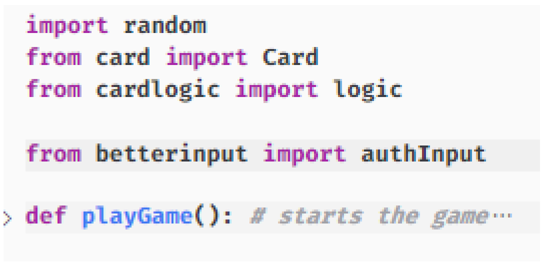
This also contains the card class



```python
class Card:
    def __init__(self, colour, number):
        self.colour = colour
        self.number = number
```

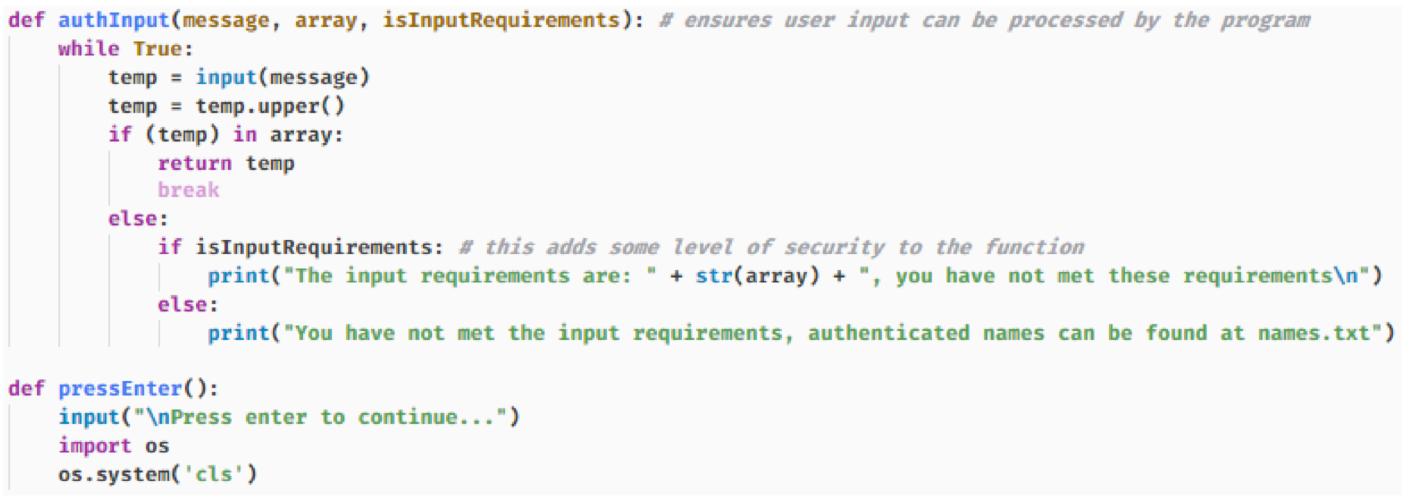
game.py

Contains a function which starts and runs the game. It has random imported and has the Card and logic classes imported.


```
import random
from card import Card
from cardlogic import logic

from betterinput import authInput

def playGame(): # starts the game···
```

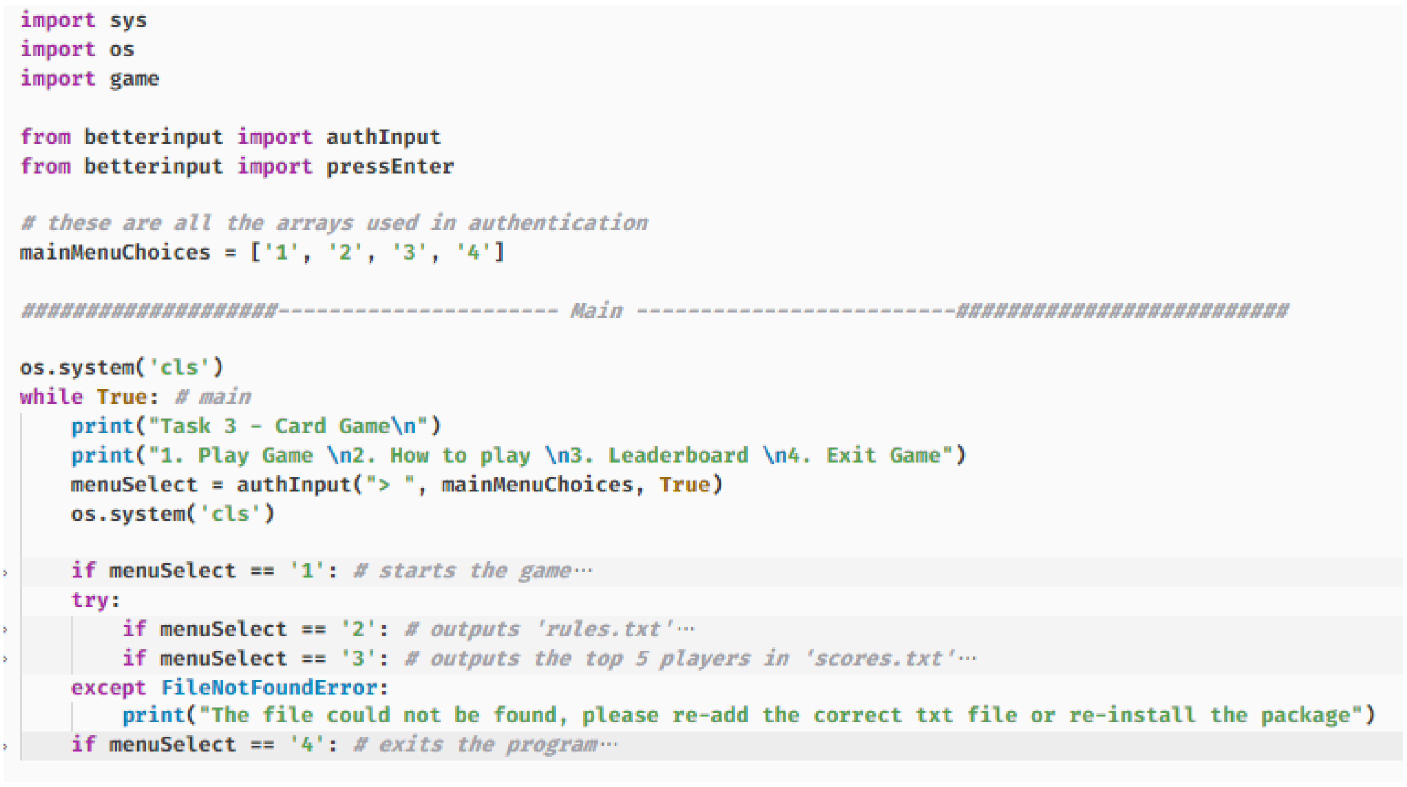
betterinput.py

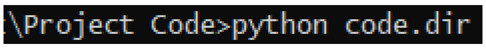
This only contains two functions but both are used in multiple places in the project. authInput has already been described and pressEnter just allows for a better user experience by clearing the screen when the user presses enter.


```
def authInput(message, array, isInputRequirements): # ensures user input can be processed by the program
    while True:
        temp = input(message)
        temp = temp.upper()
        if (temp) in array:
            return temp
            break
        else:
            if isInputRequirements: # this adds some level of security to the function
                print("The input requirements are: " + str(array) + ", you have not met these requirements\n")
            else:
                print("You have not met the input requirements, authenticated names can be found at names.txt")

def pressEnter():
    input("\nPress enter to continue...")
    import os
    os.system('cls')
```

__main__.py

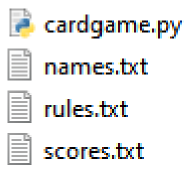
This contains the main menu and is where the program is ran from!


```
import sys
import os
import game

from betterinput import authInput
from betterinput import pressEnter

# these are all the arrays used in authentication
mainMenuChoices = ['1', '2', '3', '4']

####################---------------------- Main -------------------------##########################

os.system('cls')
while True: # main
    print("Task 3 - Card Game\n")
    print("1. Play Game \n2. How to play \n3. Leaderboard \n4. Exit Game")
    menuSelect = authInput("> ", mainMenuChoices, True)
    os.system('cls')

    if menuSelect == '1': # starts the game···
    try:
        if menuSelect == '2': # outputs 'rules.txt'···
        if menuSelect == '3': # outputs the top 5 players in 'scores.txt'···
    except FileNotFoundError:
        print("The file could not be found, please re-add the correct txt file or re-install the package")
    if menuSelect == '4': # exits the program···
```

I also found a really cool feature which makes distributing all these files a bit easier! When you rename your main python file to __main__.py and place all of the needed python files into a folder, you can run the 'directory' from the command line 

```
\Project Code>python code.dir
```

like this and python will run the __main__.py file instead of spitting up a bunch of errors.



```
cardgame.py
names.txt
rules.txt
scores.txt
```

This also works if you zip the file. So knowing this, I tried zipping the python files into a .zip file then renaming the resulting zip file to 'cardgame.py'.

This means that windows would assume that the .zip file was a python file and would run it as if it was a python file. It would also appear to most users to be just an ordinary python file (which isn't editable by IDLE or any python IDE) instead of a zip file meaning the source code is hidden!

Obviously the user could just unzip the python file and it isn't anywhere near as secure (or portable) as it would be if I'd used something like pyInstaller or cython, which don't need the python interpreter installed on the system to run the outputted files. But I like the idea of all the files being kept in one 'python file'

EDIT: Turns out, this works, but the file handling won't work. Since windows uses the python.exe when you run python, the current working directory will be in the wrong place entirely so it cannot access any of the necessary files! This functionality is still possible, but the only way to run it is if you run it straight from command line or use a batch file.

Alright, this is the end of my programming project! The project runs according to the specification and I have added tons of functionality to the project as well, I have learnt a lot about how to structure my programs in the future and have built a program that I am proud of!