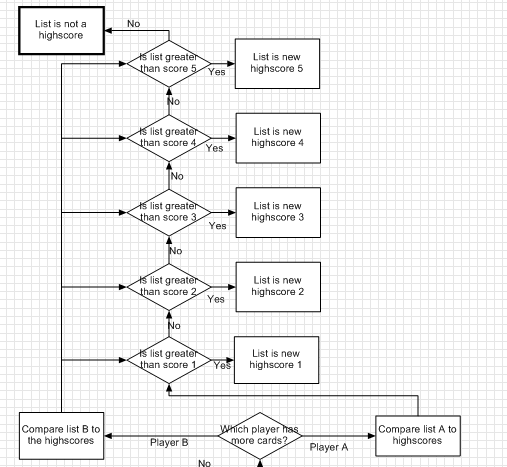
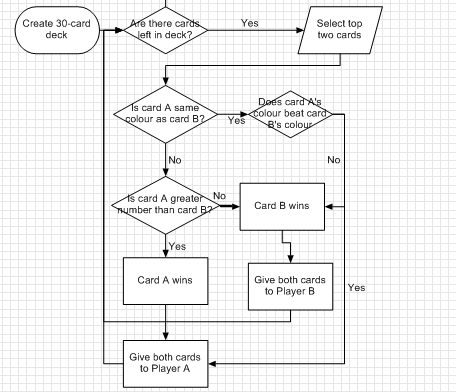
GCSE NEA Computer Science Exam

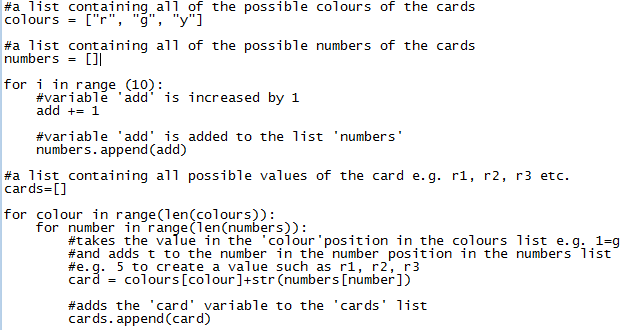
Algorithm



Requirements Breakdown

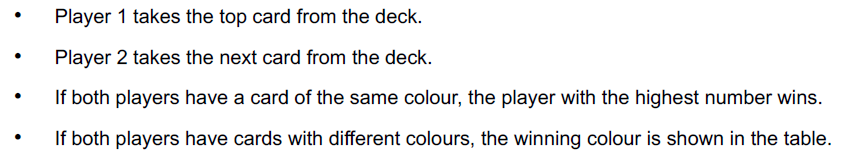


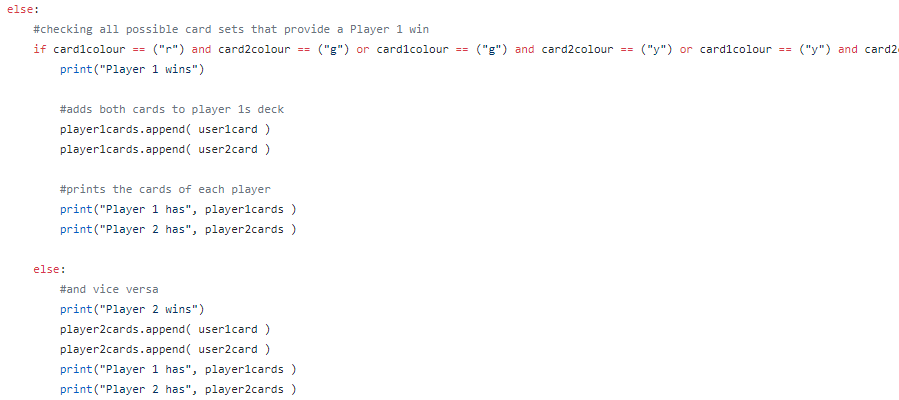
The code needs to have 30 cards with 10 numbers for each colour. This is best achieved with two lists and a FOR loop.

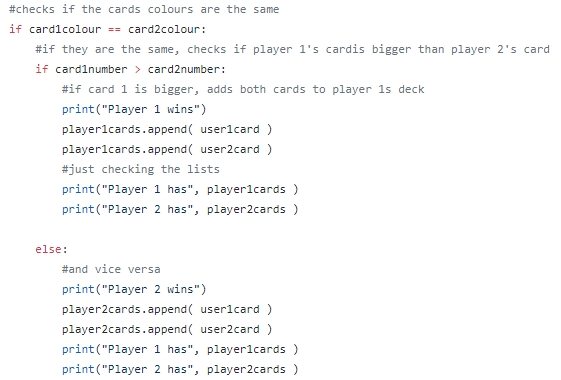




The shuffle can be done using the random module two ways, the most efficient is random.shuffle(list). This will code will simply shuffle the values in the list into random positions.

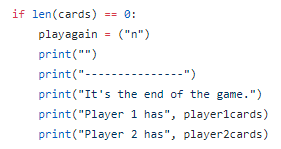


After shuffling, all I have to do is take the two top cards and then remove them from the list. I compare the colour first and then the number.

First of all I check if the card colours are the same, if they are then I compare the card numbers. The card with the biggest number wins and the winner gets both cards.

If the cards have different colours then I first check to see if card 1 wins, if card 1 doesn’t win then card 2 does. The winner gets both cards.

Red > green > yellow.



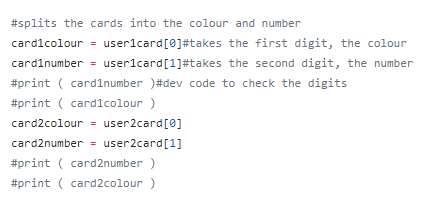
If there are no more cards in the deck, I set playagain to n so that the code no longer runs and give a display of all of the cards that each player has.



After the game is over, I check which player has more cards and then add the winner’s name to the list so that it can be added to the scoreboard later. I also display the difference between the two player’s decks.



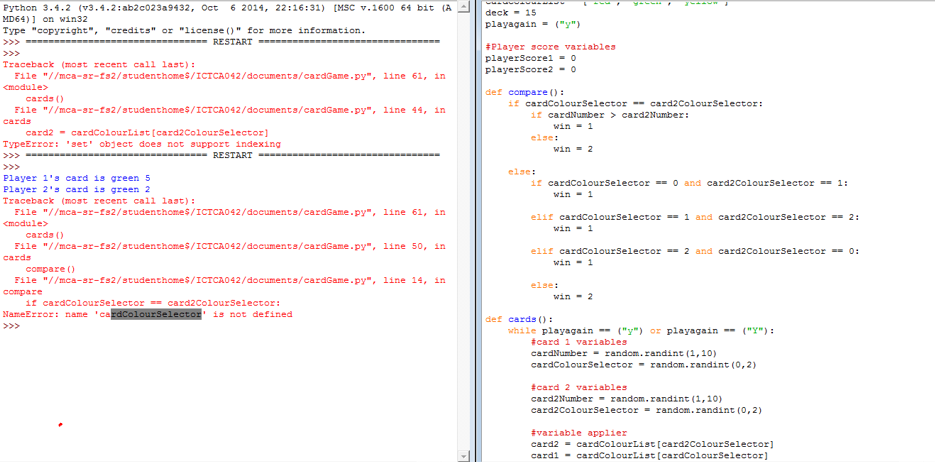
If there are still cards in the deck then it asks if the players want to go again.



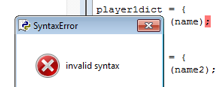
When I was comparing the cards I had to split them into the colour and number, I separated the number by creating a variable that is equal to the second digits of each players’ cards .I then did the same for the first digit to get the colour.

I have not managed to get a script working that allows a scoreboard to be implemented.

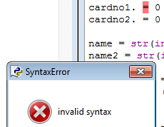
Errors



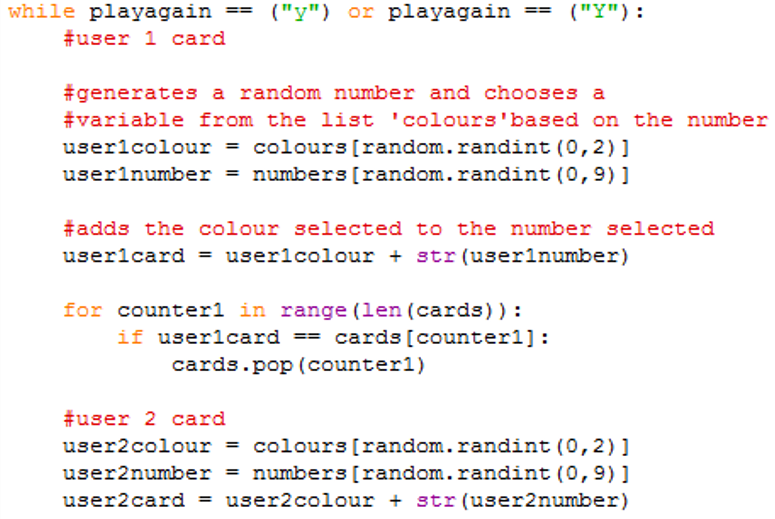
My first error occurred in the first version of my game where I tried to use defined scripts to perform all of the tasks. This didn’t work because the variable was called before it was defined. This error is why I switched to a non-definition version.



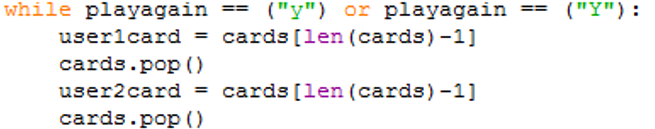
The second error was a syntax error, I accidentally used semi-colons instead of colons.



Python doesn’t like full stops.

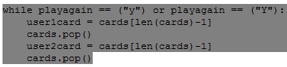


I shuffle the cards and then I select a random card, but I need to pick the top two cards and the top 2 are already random because of the shuffle anyway.

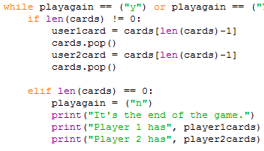


This is a far, far simpler piece of code that does the same thing.

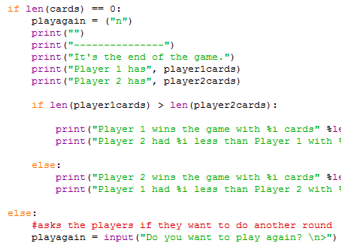




An index out of range occurred when the cards are all used up because there was no deck check, so the code tries to keep selecting and deleting cards when the list is empty.



I added code that checked if the deck was empty or not. However, because it occurs after the player tries to play another round, it was quite misleading.

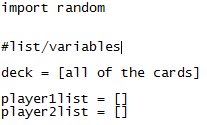


I moved the check to the end of the code, instead of checking right after trying to start a new round, it checks right before the game asks if the player wants to play a new round.

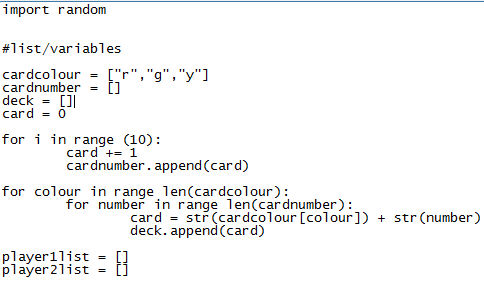
Development

The first thing I did was figure out which modules I’d need and how many variables I needed. I luckily only needed the random module for the code, I knew I would need lists to hold the player’s cards, and store the deck.

I then gave these lists appropriate names. The first handful of lines I wrote were:



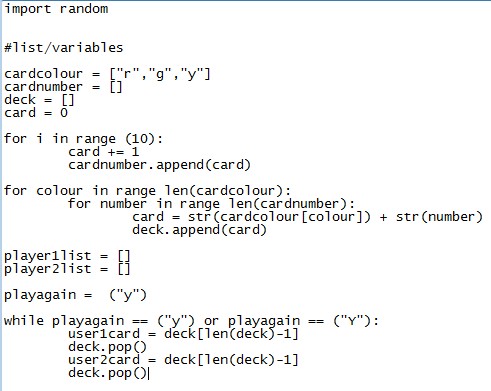
I then decided that I didn’t want to write out every single card combination so I created another two lists.



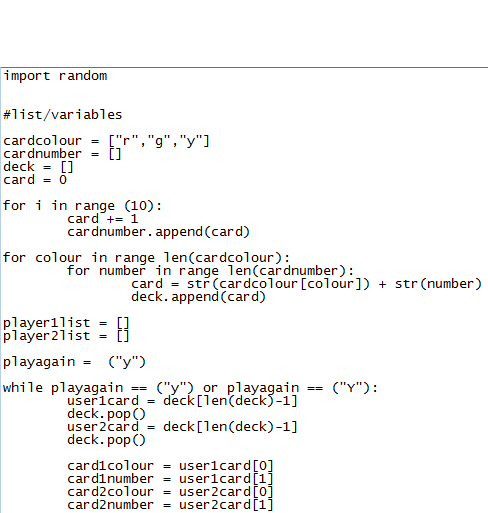
The first list contained all three colours and the second list contained all 10 numbers, I used a list to fill in all the numbers because I’d prefer to do that than enter 1-10 into a list. Then, I’d loop through the colours and add a number to them in increasing order E.g. r1, r2, r3, r4, r5 etc., I then added these to a list called ‘deck’. It fills the list with all colour-number combinations possible.

After I did this, I added random.shuffle() so that the deck was in a random order.

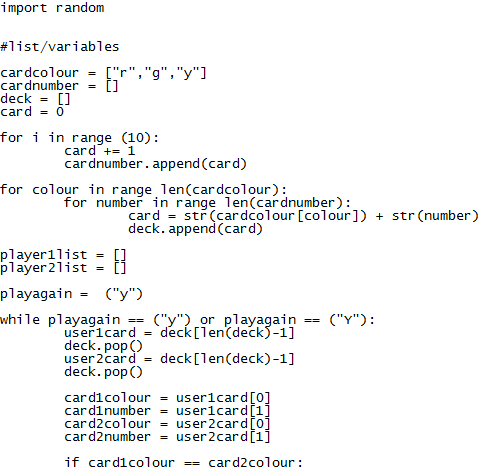
I then moved on to the main block of my code. I began by running a check on a variable called playagain, I used this variable to allow the user to choose whether to keep playing or not. After this, I added the code to select each player’s card and then remove it from the deck



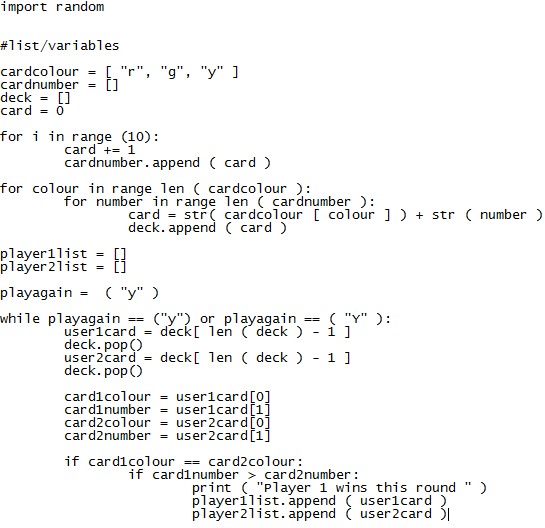
This code works by taking the length of the list, say 30, and taking one off because Python counts lists from 0. This will then become player 1’s card, to remove this card from the deck I use deck.pop() which pops the uppermost card when a value is not specified, thereby removing it from the deck.



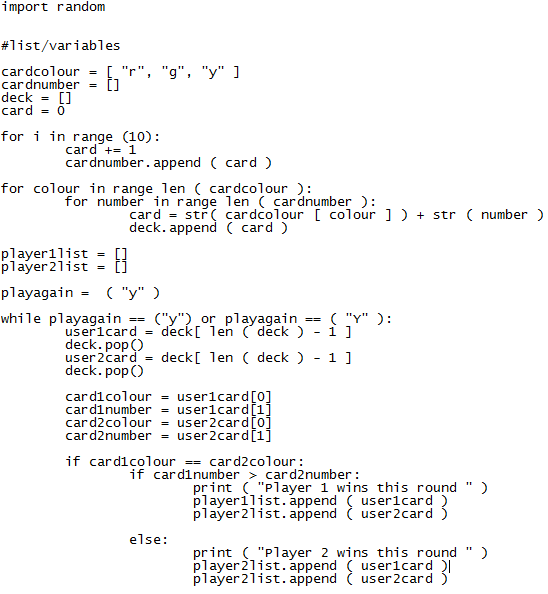
The next step is to split the cards into two separate variables that can be more easily compared. I do this by creating a variable that will store a single digit of the card, this is easy to do with my code because the colours are represented by a single letter rather than a full word. The first digit is the colour and the number is the second digit, however, Python counts from 0 so the first digit is variable[0]. In this examples the variables I want to split are user 1’s card list and user 2’s card list.



Next I need to compare the cards. I begin by checking if the card’s colours are the same.

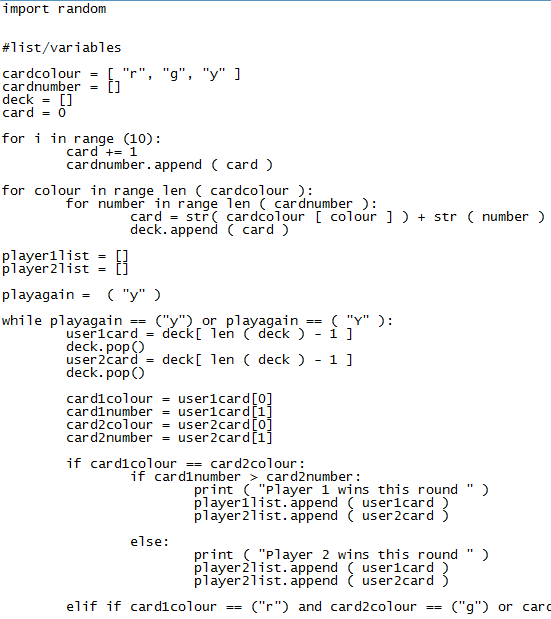


If the card’s colours are equal then I check if card 1 has a greater number than card 2, if it does then the code tells the player who won the round and gives both cards to player 1.

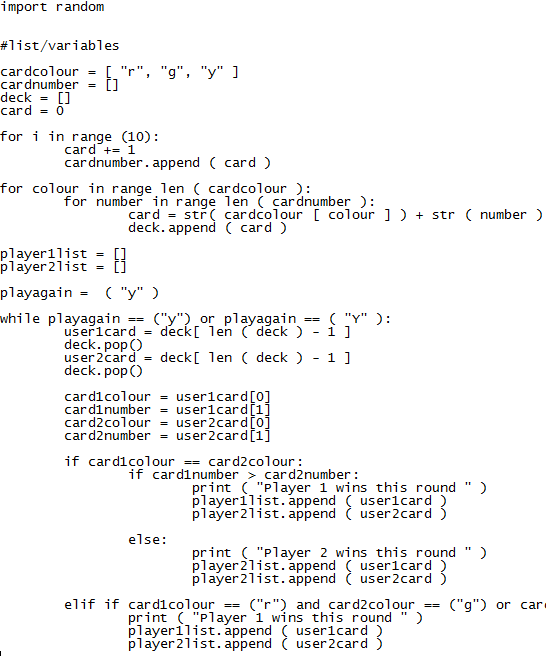


If player 1’s card hasn’t got a larger number then there can only be one other option, player 2 has a larger card number. Because of this I don’t need to compare the numbers a second time and only need to use an else command. This time around, the system tells the players that player 2 won and gives both of the cards to player 2.





Next I have to compare the colours. At the time I was writing this code, this horrible line is the only method I could think of how to do this (I now realise that there is probably a far more efficient method). What it does is it individually goes through each colour combination where player 1 wins, so red and green, green and yellow, or yellow and red.



If the players’ cards fit into one of the pairs where player 1 wins then the system tells the players that player 1 wins and gives player 1 both of the cards.

