

NEA sample solution

Task 2 – Cows and bulls (solution 1)

GCSE Computer Science 8520

NEA (8520/CA/CB/CC/CD/CE)

Introduction

The attached NEA sample scenario solution is provided to give teachers an indication of the type of solution that students could complete in response to the specimen material NEA scenario: Task 2 – Cows and Bulls for the new GCSE Computer Science (8520) specification. This specification is for first teaching from September 2016.

This sample solution should only be used to enable teachers to commence planning work for the NEA, (the live NEA scenario will be available for the first time from September 2017). As a result teachers should use this only as a guide to the forthcoming live scenarios. This solution is not a 'real' solution and is provided as an example only. It cannot be used to accurately determine standards in the future.

At our preparing to teach events, spring 2016, the sample scenarios and solutions will be considered and discussed. After these events, appropriate commentaries will be provided, by the senior examining team, to enable teachers to understand the appropriate level achieved by this solution.

Teacher Standardisation events will be used to prepare teachers for the first NEA assessment, which will be available in centres from September 2017. At these meetings teachers will be made aware of the standard.

.

Contents

Designing the solution	3
Creating the solution	9
Testing the solution	13
Test planning	13
Testing evidence	15
Potential enhancements and refinements	24
Appendix – Complete code listing.	25

Designing the solution

The program is a game of Cows and Bulls. The aim of the game is to guess a 4 digit number that has been randomly generated. To help guessing the player will get hints to say what numbers are correct and in the right place (bulls) or correct but in the wrong place (cows).

The program will be created using python programming language and using the IDLE IDE.

The first part of the program needs to generate a number and make sure the 4 numbers are not repeated. The python random module should be imported into the program, in order to generate a random number. The generated number will be the format of a python list of 4 digits, so that the values can be looped through.

The python function randrange() can be used to generate a single number between 0 and 9, as required in the scenario. A loop is needed to do this 4 times and check that the number is not duplicated, by comparing each new number to those generated in previous iterations of the loop. See **Flow Chart 2**.

The program needs to allow the player to enter a guess and continue doing this until the generated number is guessed or they type exit. See **Flow Chart 1**.

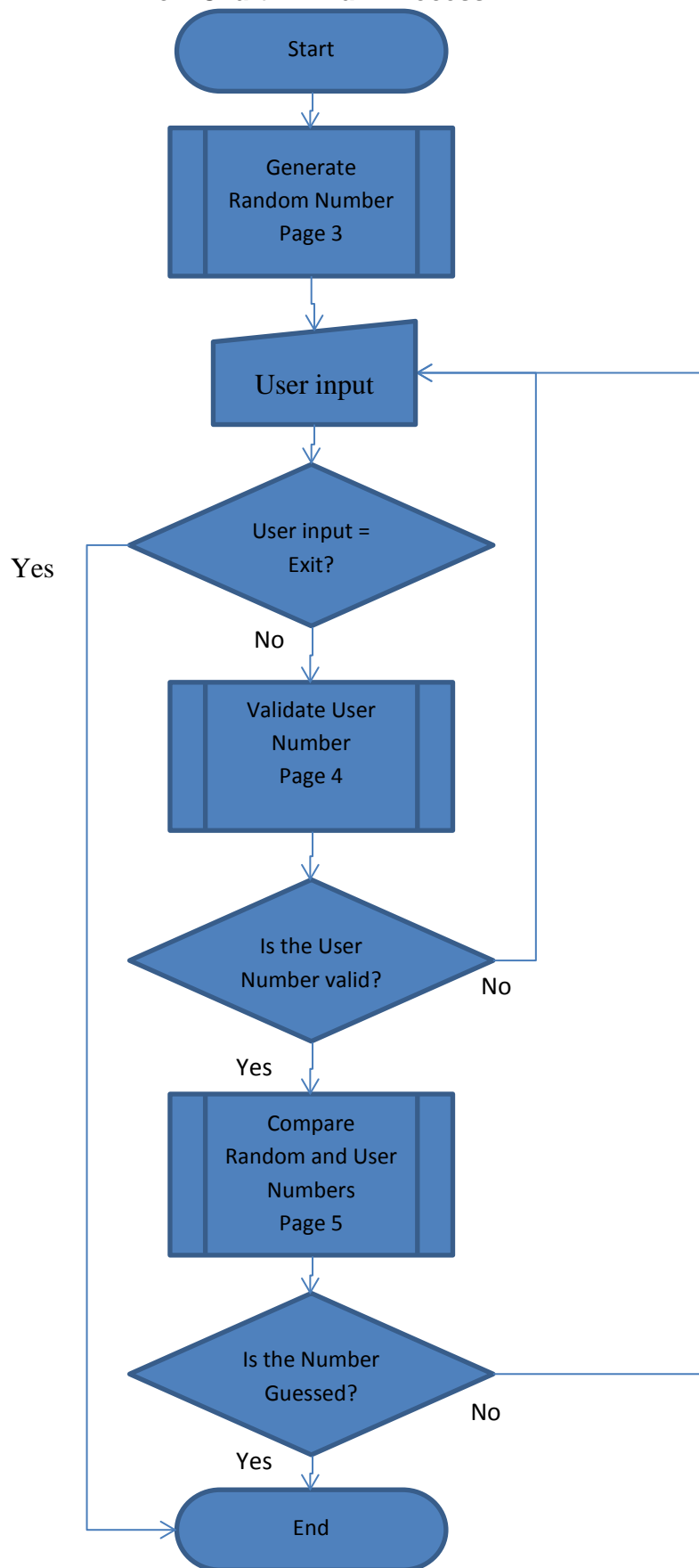
The user entered number needs to be validated to ensure it is a four digit number without duplicates. A loop is needed to check each digit in the number and ensure it is not used twice or more. See **Flow Chart 3**.

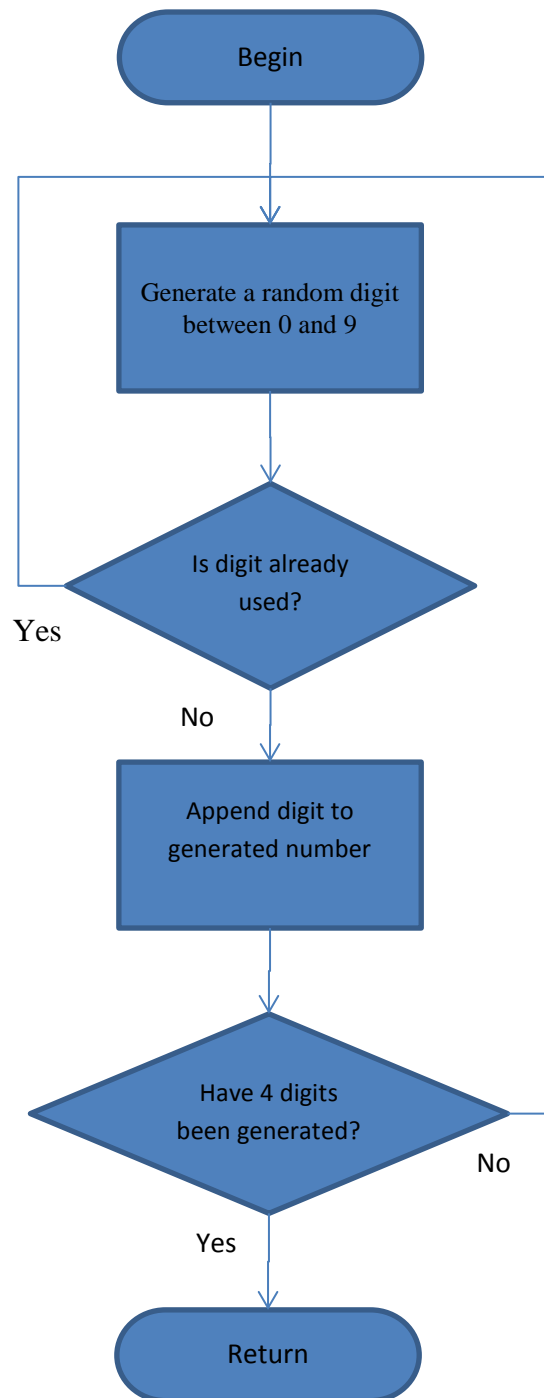
The user entered number needs to be compared to the random generated number. To do this each digit in the user number is compared to the digit in the same position in the random number. If they are equal then 1 will be added to the bulls counter. If the user digit is not in the same position but is in the random number anyway then 1 is added to the cows counter. See **Flow Chart 4**.

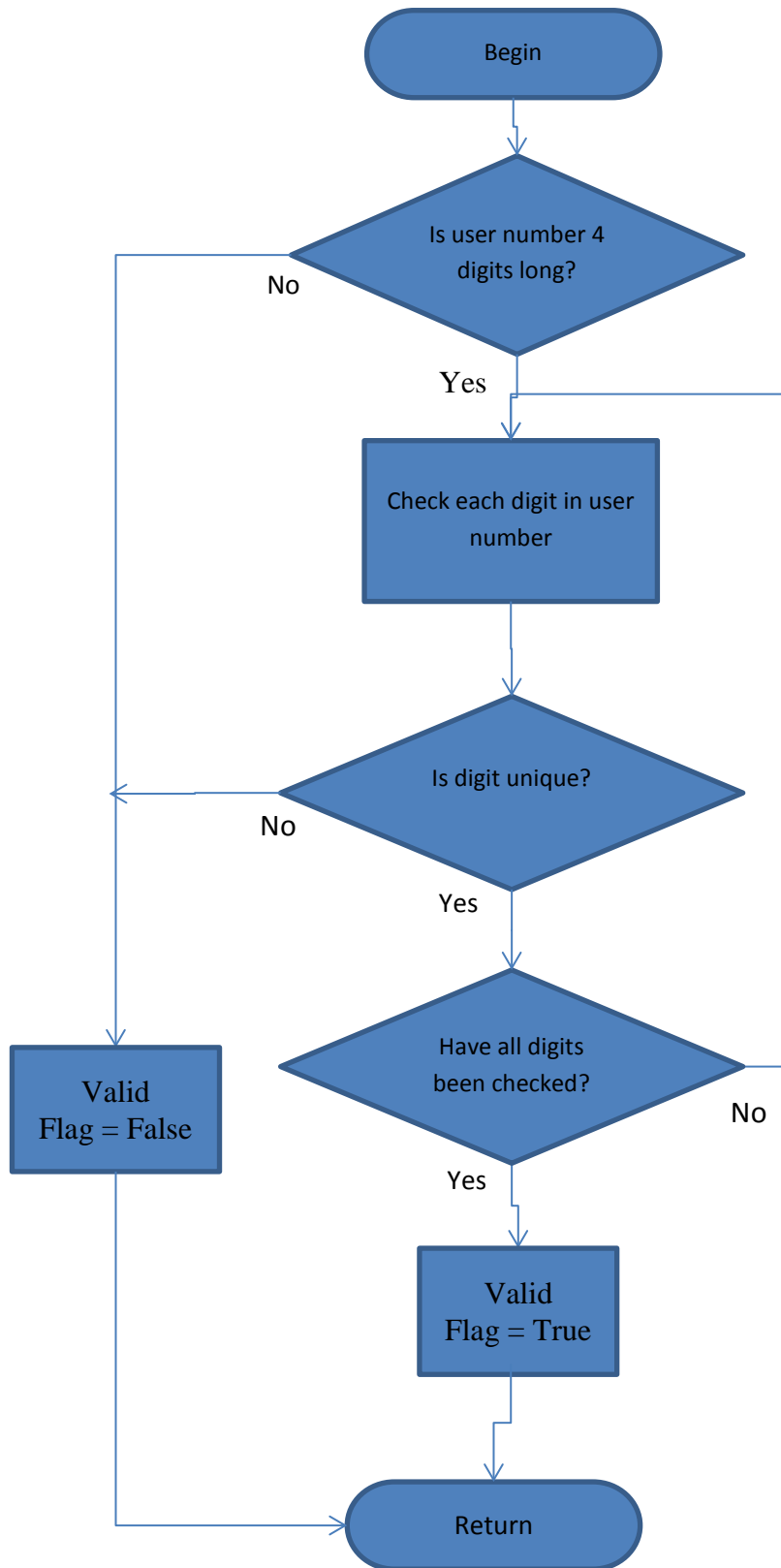
The data variables needed in the program are defined below:

Data Variable Name	Data Type	Purpose
GeneratedNumber[]	List	Used to hold the 4 digit generated number that is to be guessed by the player.
UserNumber[]	List	Used to hold the number input by the player. This will be compared digit by digit to the generated number.
Bulls	Integer	Holds a count of the number of bulls (a correct digit in the correct position).
Cows	Integer	Holds a count of the number of cows (a correct number in the wrong position).
NoOfGuesses	Integer	Holds the count of the number of guesses that the player has made.
Valid	Boolean	A flag that is set to true if the UserNumber is valid, or false if it is invalid.

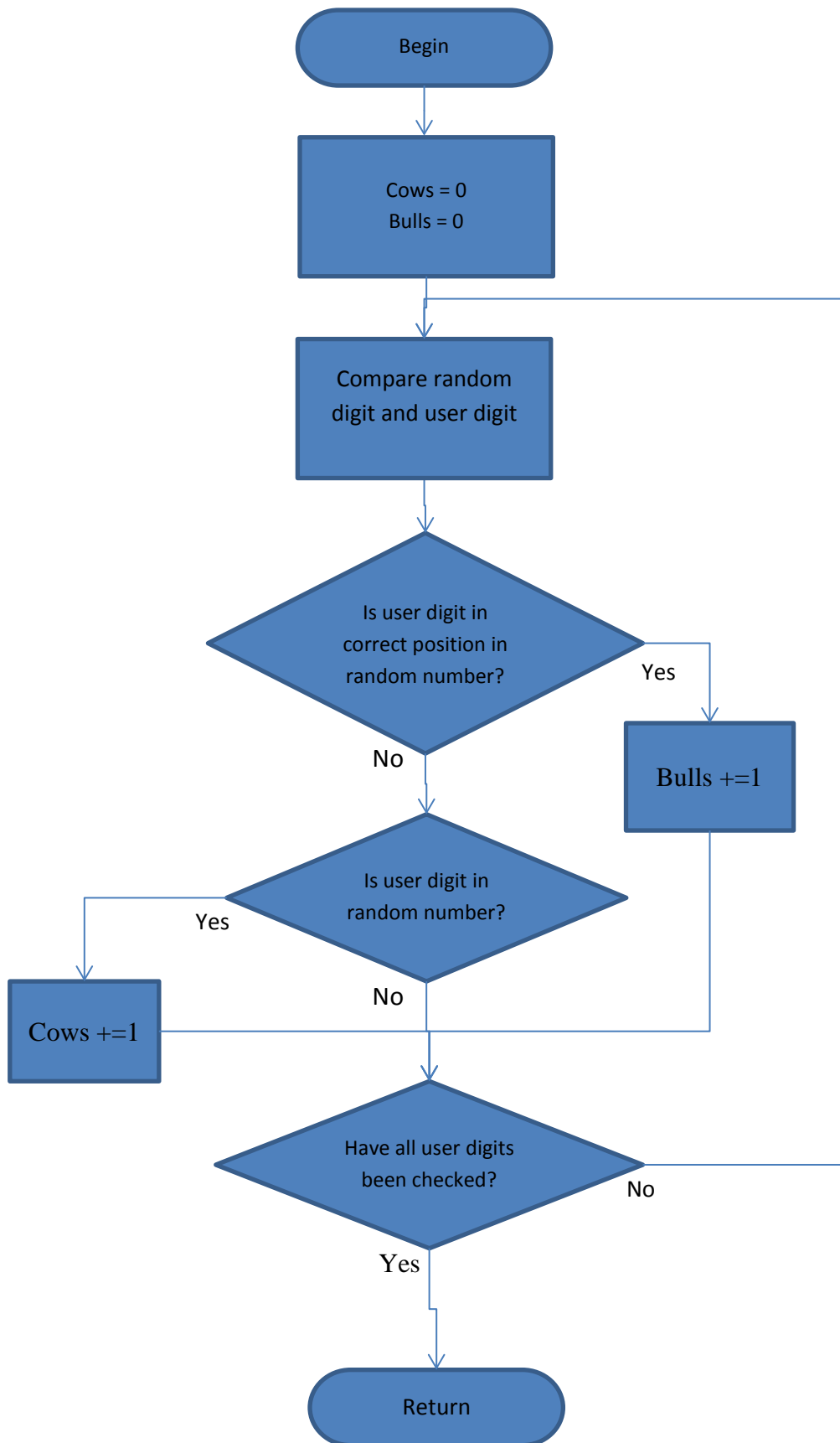
Flow Chart 1 – Main Process



Flow Chart 2 – Generate Random Number

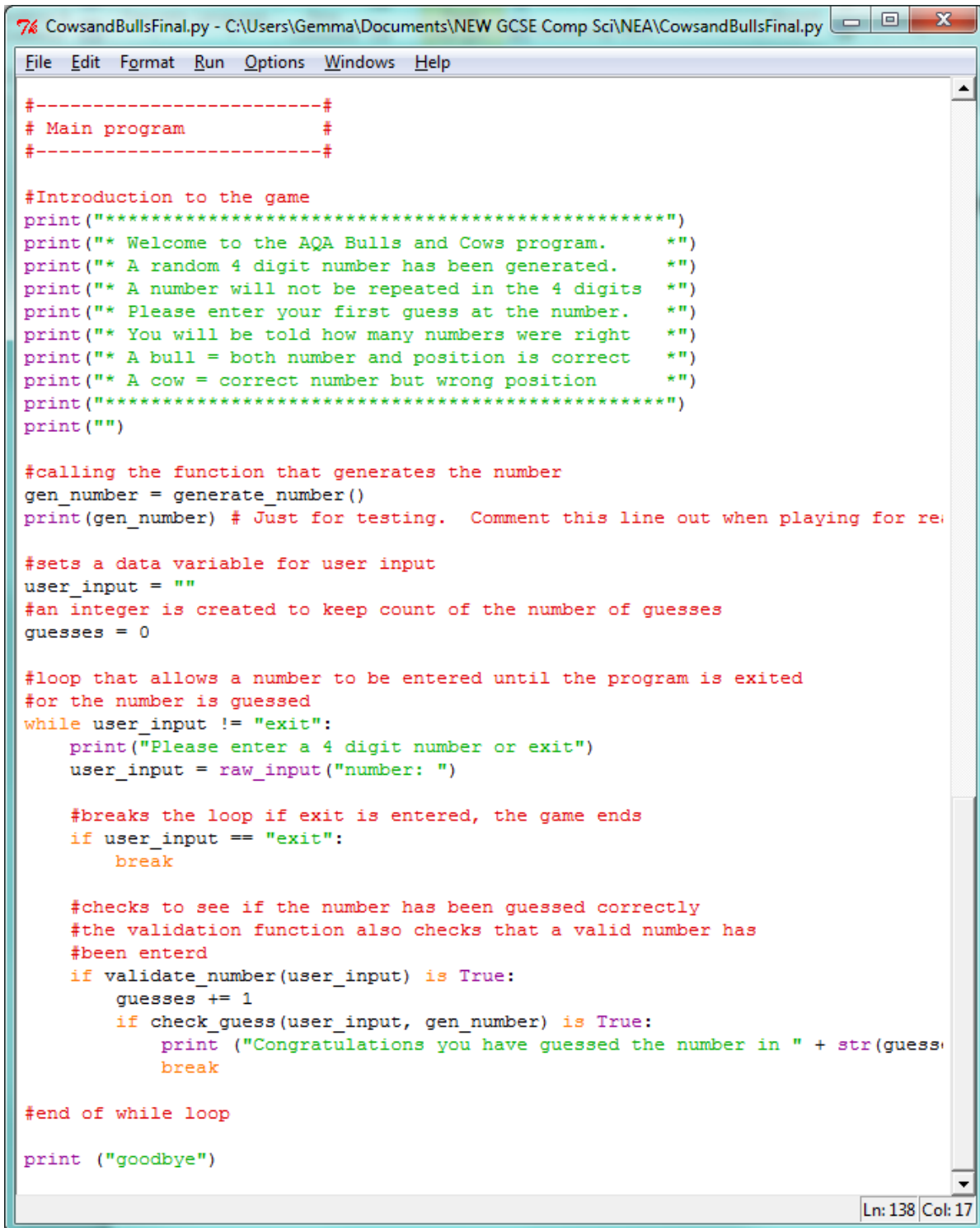
Flow Chart 3 – Validate User Number

Flow Chart 4 – Compare Random and User Numbers



Creating the solution

The code starts with the main program. I have added comments in the program to explain the processing. I have a main loop that continues until the user wants to exit by typing Exit. Inside that loop a user input is validated and checked by calling other functions.



```

76 CowsandBullsFinal.py - C:\Users\Gemma\Documents\NEW GCSE Comp Sci\NEA\CowsandBullsFinal.py
File Edit Format Run Options Windows Help

#-----#
# Main program      #
#-----#

#Introduction to the game
print("*****")
print("** Welcome to the AQA Bulls and Cows program.      **")
print("** A random 4 digit number has been generated.     **")
print("** A number will not be repeated in the 4 digits  **")
print("** Please enter your first guess at the number.    **")
print("** You will be told how many numbers were right   **")
print("** A bull = both number and position is correct    **")
print("** A cow = correct number but wrong position      **")
print("*****")
print("")

#calling the function that generates the number
gen_number = generate_number()
print(gen_number) # Just for testing. Comment this line out when playing for real

#sets a data variable for user input
user_input = ""
#an integer is created to keep count of the number of guesses
guesses = 0

#loop that allows a number to be entered until the program is exited
#or the number is guessed
while user_input != "exit":
    print("Please enter a 4 digit number or exit")
    user_input = raw_input("number: ")

    #breaks the loop if exit is entered, the game ends
    if user_input == "exit":
        break

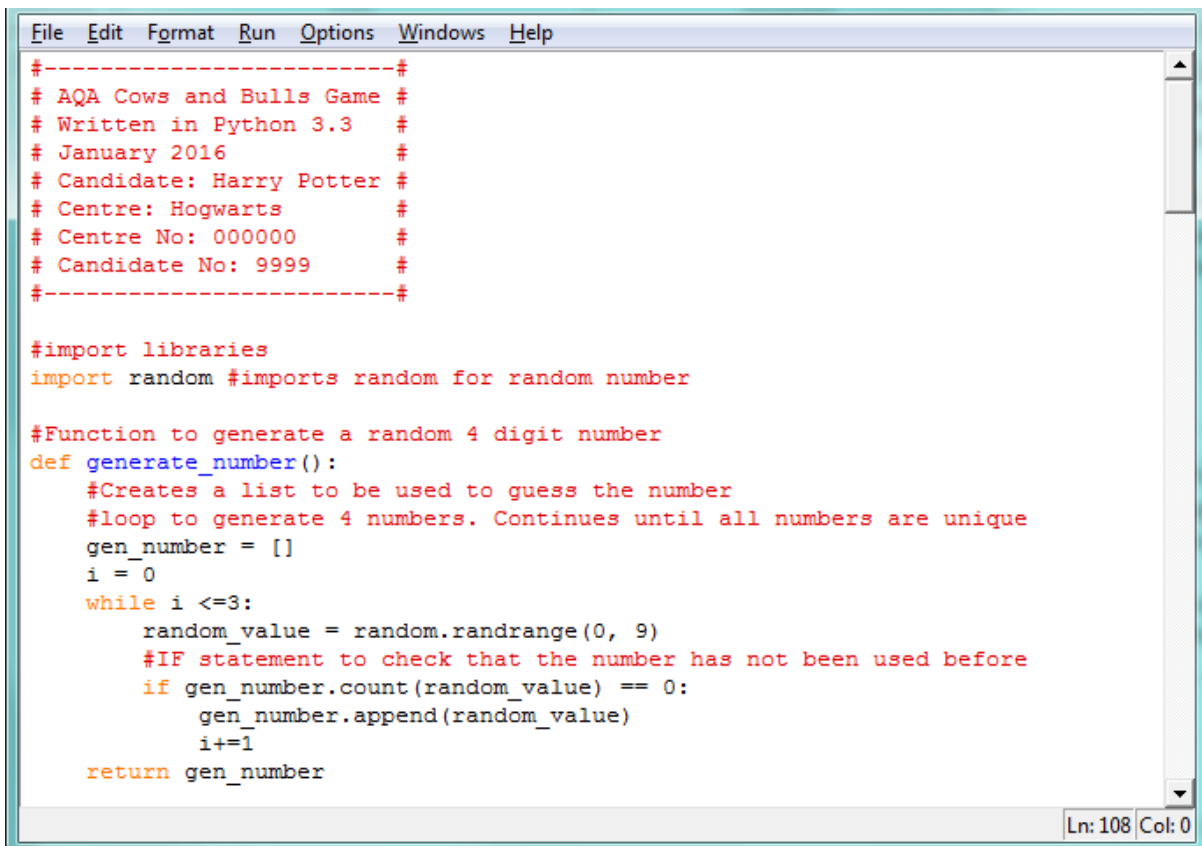
    #checks to see if the number has been guessed correctly
    #the validation function also checks that a valid number has
    #been entered
    if validate_number(user_input) is True:
        guesses += 1
        if check_guess(user_input, gen_number) is True:
            print ("Congratulations you have guessed the number in " + str(guesses))
            break

#end of while loop

print ("goodbye")
Ln: 138 Col: 17

```

Before the main loop there is a call to a function called **generate_number()**. This function returns a list data structure to the main program and is saved in the variable **gen_number**.



```
File Edit Format Run Options Windows Help
#-----#
# AQA Cows and Bulls Game #
# Written in Python 3.3 #
# January 2016 #
# Candidate: Harry Potter #
# Centre: Hogwarts #
# Centre No: 000000 #
# Candidate No: 9999 #
#-----#

#import libraries
import random #imports random for random number

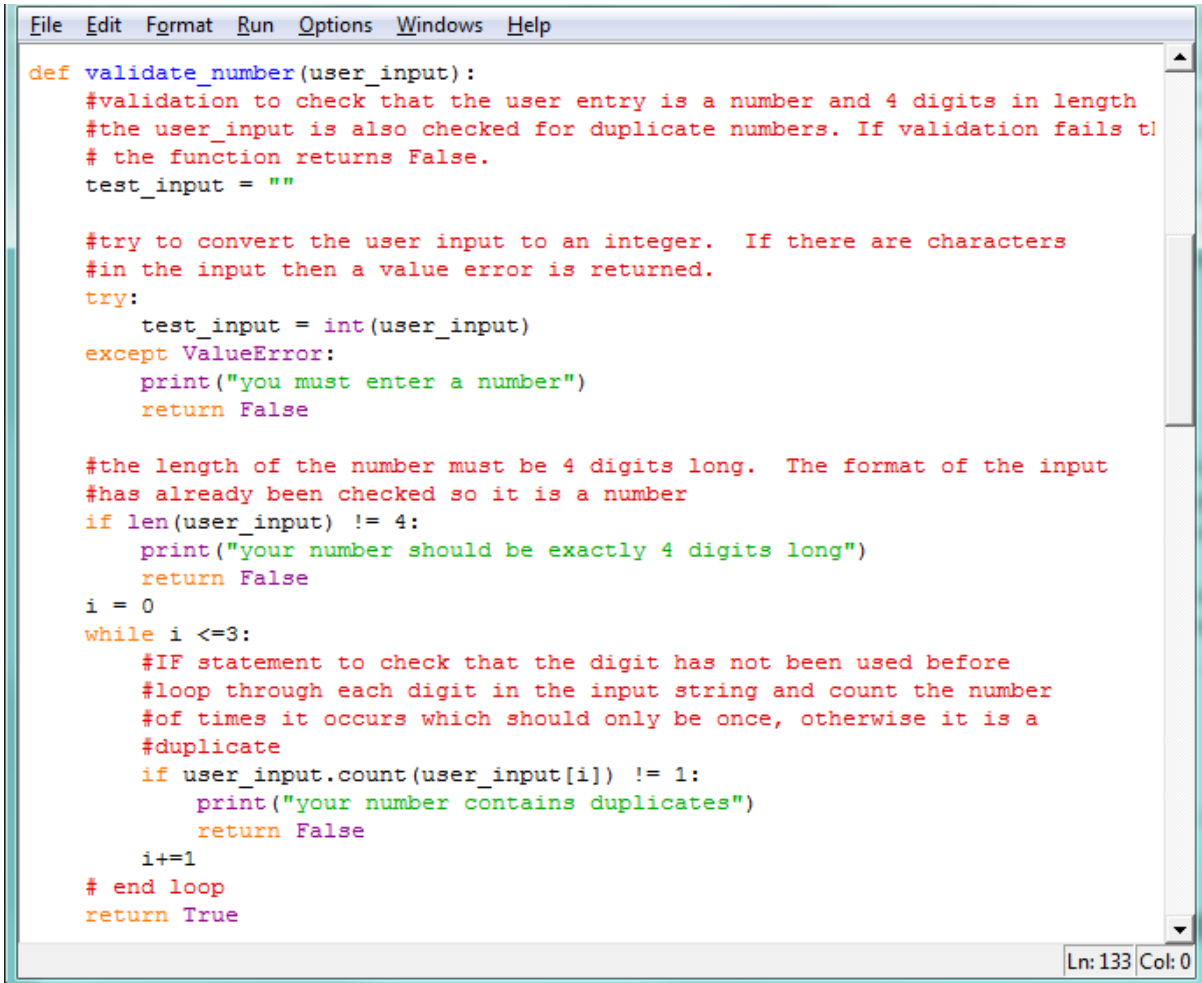
#Function to generate a random 4 digit number
def generate_number():
    #Creates a list to be used to guess the number
    #loop to generate 4 numbers. Continues until all numbers are unique
    gen_number = []
    i = 0
    while i <=3:
        random_value = random.randrange(0, 9)
        #IF statement to check that the number has not been used before
        if gen_number.count(random_value) == 0:
            gen_number.append(random_value)
            i+=1
    return gen_number

Ln: 108 Col: 0
```

This function uses the python count method to count the occurrence of a value in a list and the append method to add a value to a list. It also used the **random.randrange** function, which required a python library to be imported.

The main program then checks that a user entered number is valid by calling a function called **validate_number**. This function is passed the **user_input** and returns a Boolean value indicating if the user input has passed the validation and so can be checked against the randomly generated number **gen_number**. A loop is used to look at each digit in the user input.

The **user_input** variable has been passed by the main program. It was manually entered using the **raw_input()** function. Originally I used the `input()` function but python creates a variable that takes the form of the data entered when this function is used. Therefore the `user_input` would be created as an integer if a 4 digit number is entered, but this is no use for this program as a string is better because it can be used as a list and looped through. I originally used this and got an error, so changed to use `raw_input()` function that always returns a string.

A screenshot of a Python IDE window. The window title bar shows 'File Edit Format Run Options Windows Help'. The code is as follows:

```
def validate_number(user_input):
    #validation to check that the user entry is a number and 4 digits in length
    #the user_input is also checked for duplicate numbers. If validation fails t!
    # the function returns False.
    test_input = ""

    #try to convert the user input to an integer. If there are characters
    #in the input then a value error is returned.
    try:
        test_input = int(user_input)
    except ValueError:
        print("you must enter a number")
        return False

    #the length of the number must be 4 digits long. The format of the input
    #has already been checked so it is a number
    if len(user_input) != 4:
        print("your number should be exactly 4 digits long")
        return False

    i = 0
    while i <=3:
        #IF statement to check that the digit has not been used before
        #loop through each digit in the input string and count the number
        #of times it occurs which should only be once, otherwise it is a
        #duplicate
        if user_input.count(user_input[i]) != 1:
            print("your number contains duplicates")
            return False
        i+=1
    # end loop
    return True
```

The status bar at the bottom right shows 'Ln: 133 Col: 0'.

The final part of the main program calls the **function check_guess**. This function is passed the **user_input** and **gen_number** variables. Because these two parameters can be interpreted as lists they will be compared item by item in order to determine if the match makes a bull or a cow.

The **check_guess** procedure returns True if there are 4 bulls, meaning the number has been guessed correctly, or False if it hasn't. A count of the number of guesses is kept in the main program.

The main program finally ends when exit has been entered by the user OR the number has been guessed correctly.

There is a full code listing in the appendix.

```
File Edit Format Run Options Windows Help
def check_guess(user_input, gen_number):
    # initialise all the variables to count bulls and cows
    i = 0
    bulls = 0
    cows = 0

    #a loop that looks at each digit in the user input and compares it to the gen
    #number.

    while i <=3:
        #if the input digit is the same as the generated digit in the same posit:
        #then it is a bull

        if gen_number[i] == int(user_input[i]):
            bulls = bulls + 1

        #if the digit is in the generated number but in a different position
        #then it is a cow
        elif int(user_input[i]) in gen_number:
            cows = cows + 1
        i += 1
    # end of loop

    print ("You have " + str(bulls) + " bulls and " + str(cows) + " cows")

    #If there are 4 bulls then True is returned to the main program
    #so that a congratulations message can be printed
    if bulls == 4:
        return True
    else:
        return False
Ln: 87 Col: 18
```

Testing the solution

Test planning

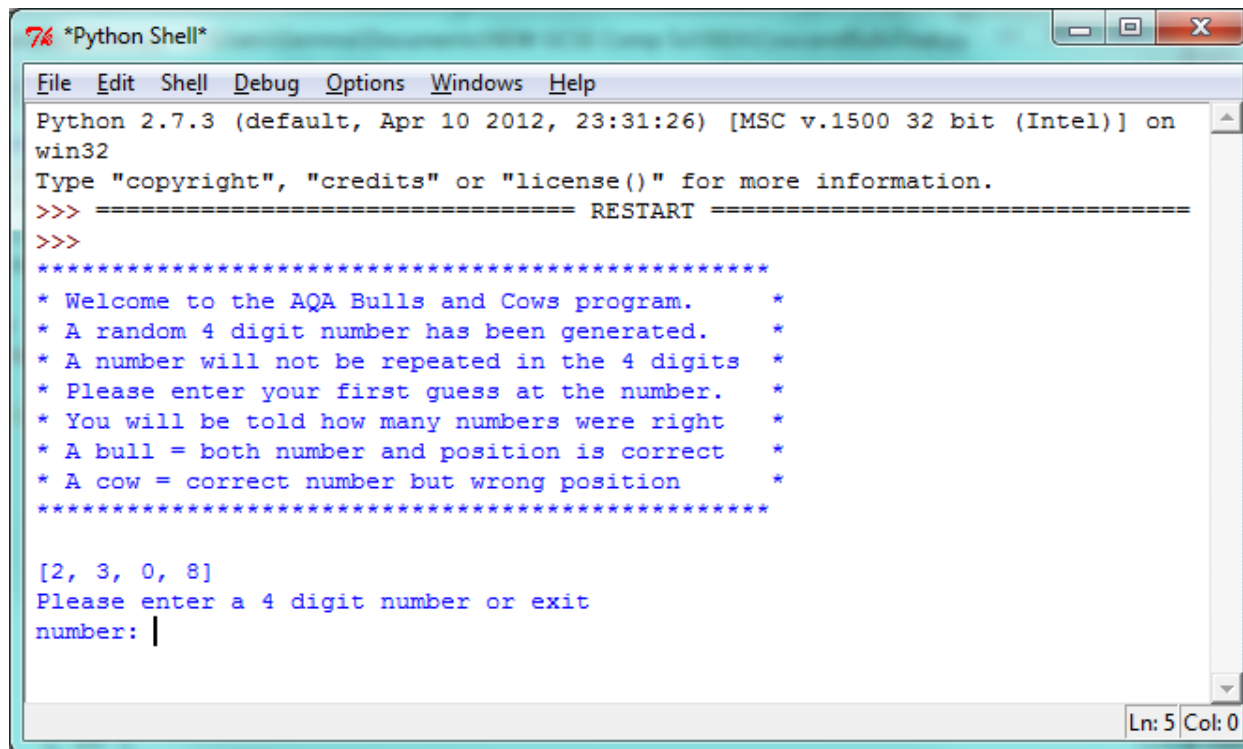
Description of Test	Test Data	Expected Results	Actual Results
The instructions are displayed and a prompt is shown for the user.	None	Instructions displayed and prompt allows keyboard entry.	As expected - Figure 1
A random 4 digit number is generated without duplicates	None	A test print output shows that a valid random number is generated.	Number [2, 3, 0, 8] has been generated and will be used for further tests. See Figure 1
Test that valid data is entered in the user prompt	1234	The number is accepted and checked against the random number. A count of cows and bulls is shown.	Error found, Figure 2a. The user entered number gave an error that the user_entry was an integer and had no length value. Retested, As expected – Figure 2b. Using the results from test 2 a count of 2 cows and 0 bulls is shown.
Test that erroneous data is entered, 4 non-digits	abcd	An error message is output stating that invalid data has been entered.	As expected – Figure 3.
Test that boundary data is entered, more than 4 digits	12345	An error message is output stating that the number is the wrong length	As expected – Figure 3.
Test that boundary data is entered, less than 4 digits	123	An error message is output stating that the number is the wrong length	As expected – Figure 3.
Test that erroneous data is entered, duplicate digits	1123	An error message is output stating that the number has duplicates.	As Expected – Figure 3.

Description of Test	Test Data	Expected Results	Actual Results
Test that the number entered has correct number of bulls and cows.	Based on random generated number	A message is output stating correct number of cows and bulls.	As expected – Figure 4.
Test that the number entered matches the random number and number of guesses is correct.	Based on random generated number	A message is output stating 4 bulls and congratulations.	As expected – Figure 5.
Exit typed before any guesses made.	Exit	Program exits with message output.	Program did not exit but gave an invalid format error – Figure 6a. Input case sensitive, so code corrected to use lower() python format function. Retested As Expected Figure 6b
Additional test to ensure program working with errors fixed.	None	Program accepted incorrect input with error messages, and correct guesses.	Figure 7.

Testing evidence

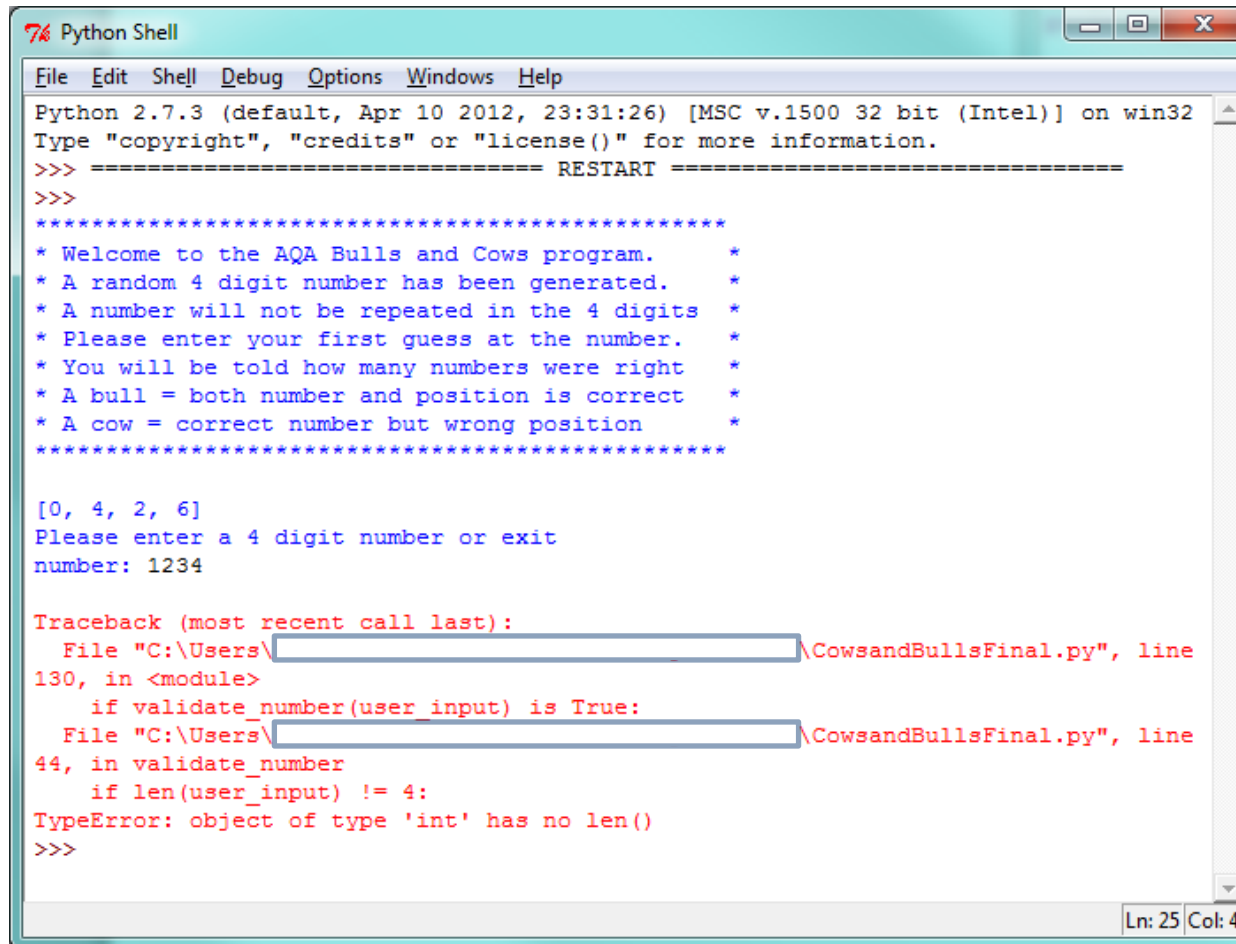
Figure 1.

The line [2, 3, 0, 8] shows the randomly generated number.



```
*Python Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
*****
* Welcome to the AQA Bulls and Cows program.      *
* A random 4 digit number has been generated.      *
* A number will not be repeated in the 4 digits    *
* Please enter your first guess at the number.     *
* You will be told how many numbers were right    *
* A bull = both number and position is correct     *
* A cow = correct number but wrong position        *
*****
[2, 3, 0, 8]
Please enter a 4 digit number or exit
number: |
Ln: 5 Col: 0
```

Figure 2a.



The screenshot shows a Python Shell window titled "Python Shell" with a menu bar (File, Edit, Shell, Debug, Options, Windows, Help). The shell displays the following text:

```
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
*****
* Welcome to the AQA Bulls and Cows program.      *
* A random 4 digit number has been generated.    *
* A number will not be repeated in the 4 digits  *
* Please enter your first guess at the number.   *
* You will be told how many numbers were right  *
* A bull = both number and position is correct  *
* A cow = correct number but wrong position     *
*****

[0, 4, 2, 6]
Please enter a 4 digit number or exit
number: 1234

Traceback (most recent call last):
  File "C:\Users\...\CowsandBullsFinal.py", line
130, in <module>
    if validate_number(user_input) is True:
  File "C:\Users\...\CowsandBullsFinal.py", line
44, in validate_number
    if len(user_input) != 4:
TypeError: object of type 'int' has no len()
>>>
```

The status bar at the bottom right of the shell window shows "Ln: 25 Col: 4".

```
while user_input != "exit":
    print("Please enter a 4 digit number or exit")
    user_input = input("number: ")
```


This was a problem because I had used the function `input()` for the user input. I changed the code to use `raw_input()` and it worked.

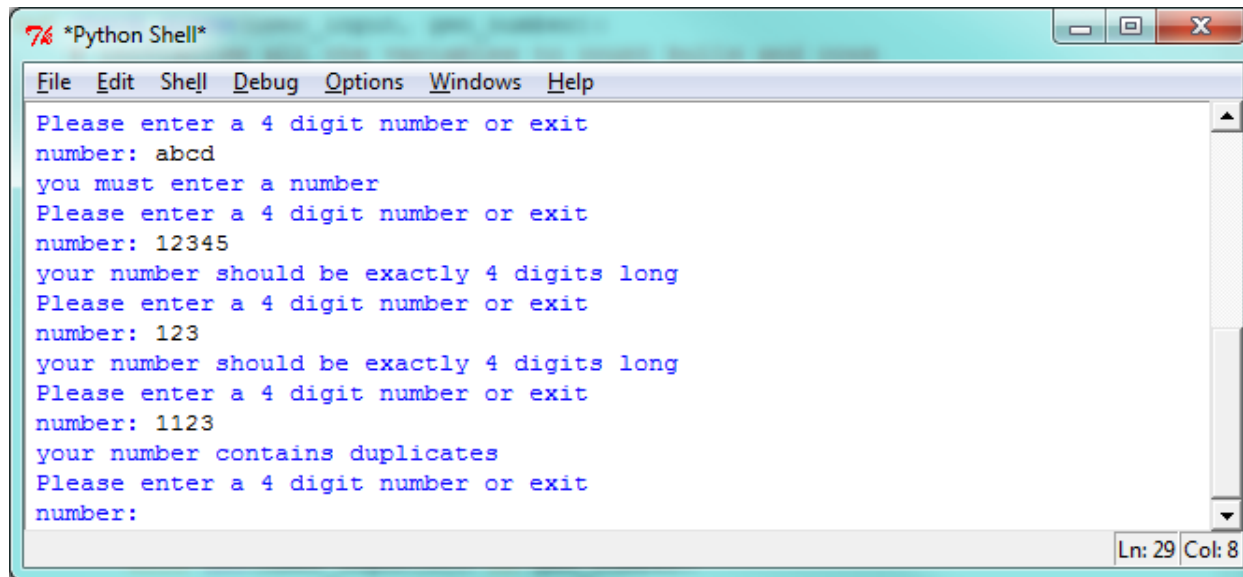
```
while user_input != "exit":  
    print("Please enter a 4 digit number or exit")  
    user_input = raw_input("number: ")
```

Fixed code

Figure 2b.

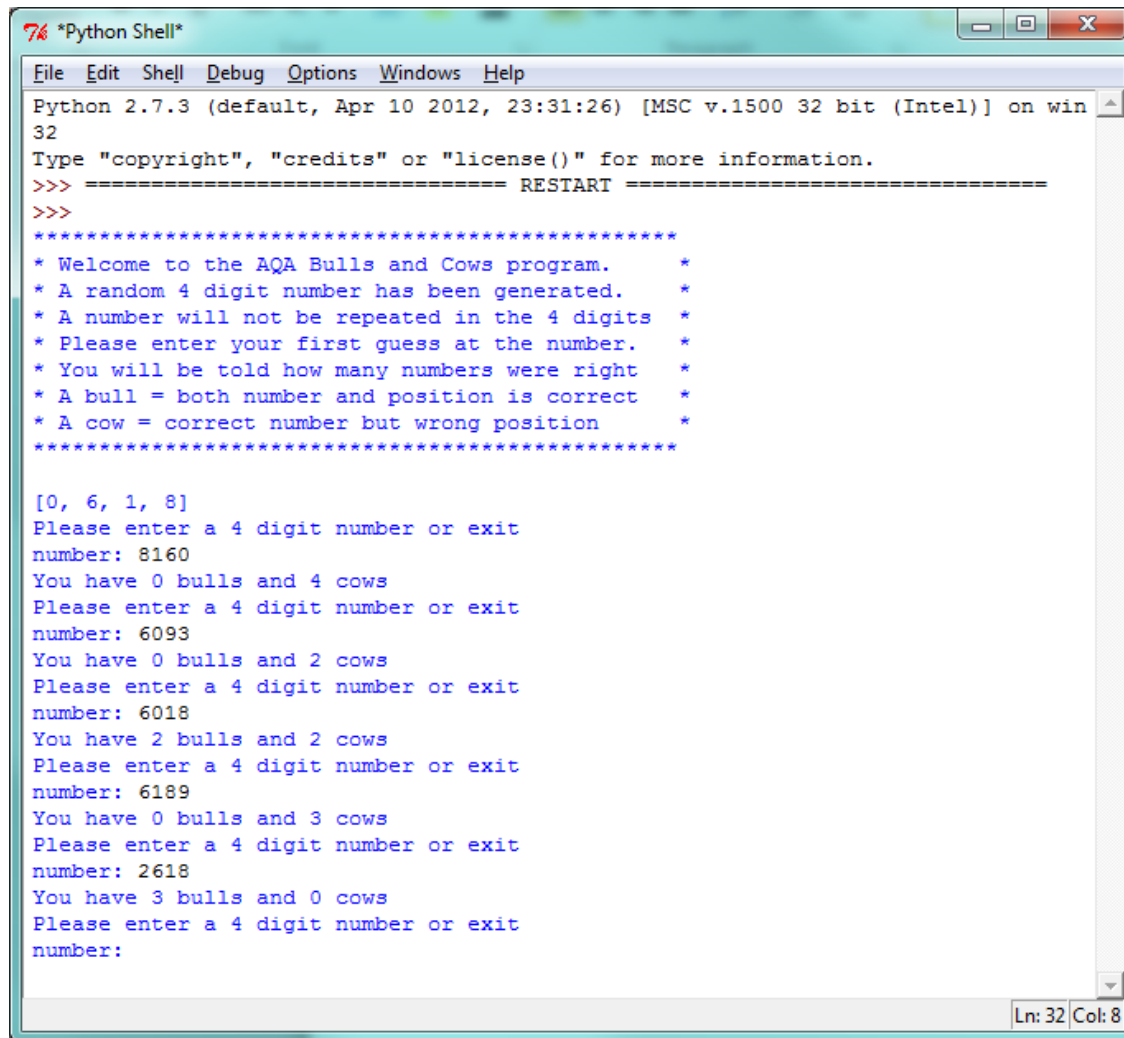
```
* You will be told how many numbers were right *  
* A bull = both number and position is correct *  
* A cow = correct number but wrong position *  
*****  
  
[2, 3, 0, 8]  
Please enter a 4 digit number or exit  
number: 1234  
You have 0 bulls and 2 cows  
Please enter a 4 digit number or exit  
number: |  
  
Ln: 20 Col: 8
```

Figure 3.



```
*Python Shell*  
File Edit Shell Debug Options Windows Help  
Please enter a 4 digit number or exit  
number: abcd  
you must enter a number  
Please enter a 4 digit number or exit  
number: 12345  
your number should be exactly 4 digits long  
Please enter a 4 digit number or exit  
number: 123  
your number should be exactly 4 digits long  
Please enter a 4 digit number or exit  
number: 1123  
your number contains duplicates  
Please enter a 4 digit number or exit  
number:  
Ln: 29 Col: 8
```

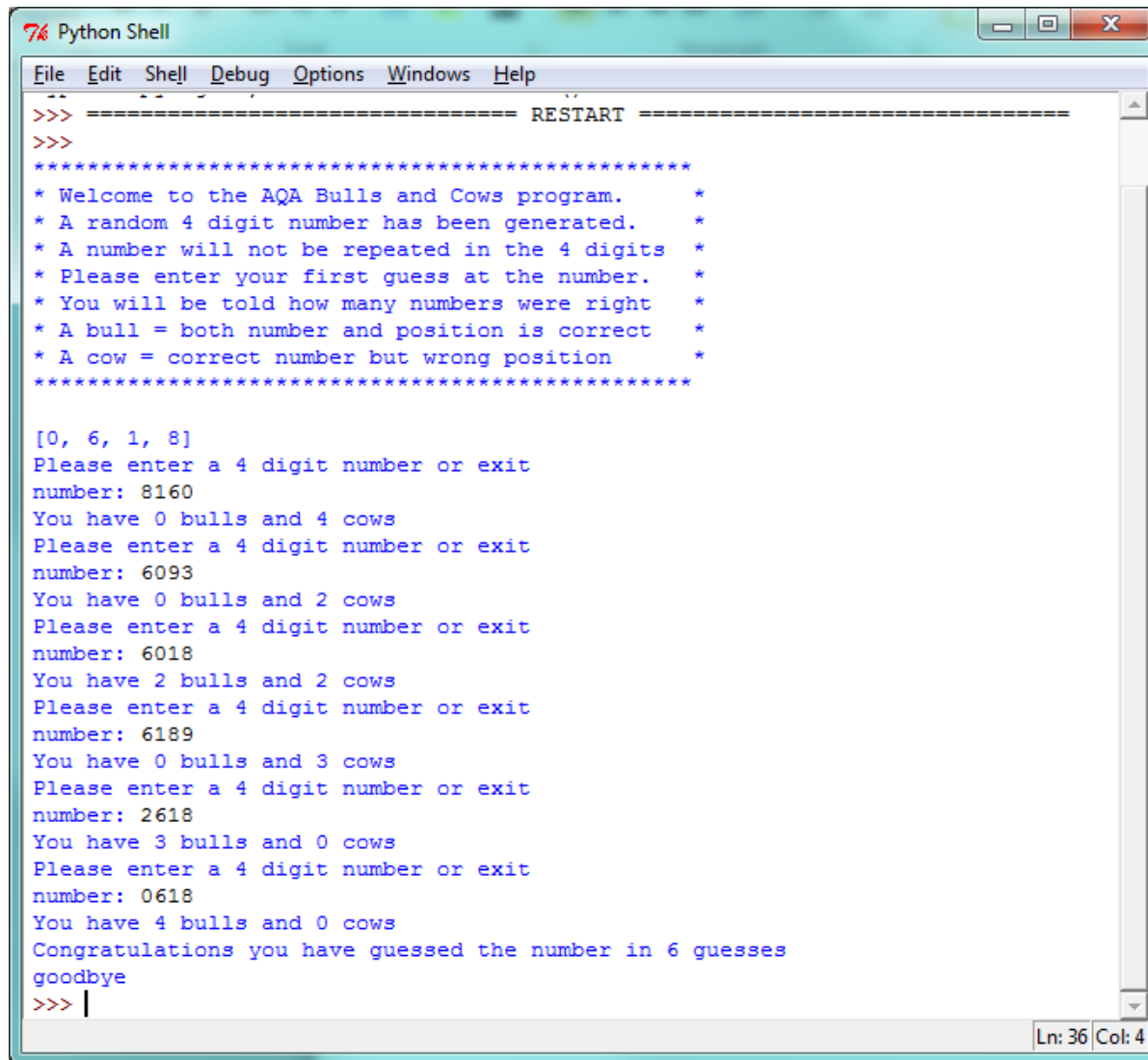
Figure 4.



```
*Python Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
*****
* Welcome to the AQA Bulls and Cows program. *
* A random 4 digit number has been generated. *
* A number will not be repeated in the 4 digits *
* Please enter your first guess at the number. *
* You will be told how many numbers were right *
* A bull = both number and position is correct *
* A cow = correct number but wrong position *
*****

[0, 6, 1, 8]
Please enter a 4 digit number or exit
number: 8160
You have 0 bulls and 4 cows
Please enter a 4 digit number or exit
number: 6093
You have 0 bulls and 2 cows
Please enter a 4 digit number or exit
number: 6018
You have 2 bulls and 2 cows
Please enter a 4 digit number or exit
number: 6189
You have 0 bulls and 3 cows
Please enter a 4 digit number or exit
number: 2618
You have 3 bulls and 0 cows
Please enter a 4 digit number or exit
number:
Ln: 32 Col: 8
```

Figure 5

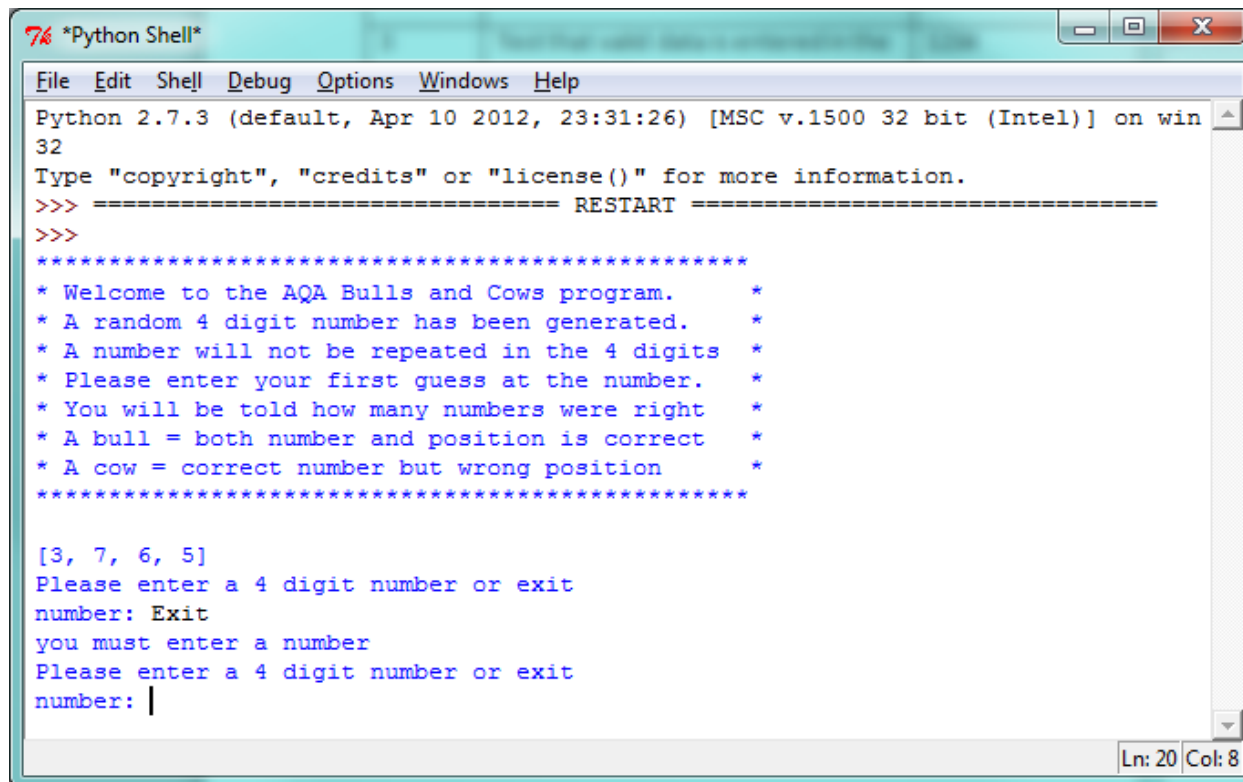


```
>>> ===== RESTART =====
>>>
*****
* Welcome to the AQA Bulls and Cows program.      *
* A random 4 digit number has been generated.     *
* A number will not be repeated in the 4 digits  *
* Please enter your first guess at the number.   *
* You will be told how many numbers were right  *
* A bull = both number and position is correct   *
* A cow = correct number but wrong position      *
*****

[0, 6, 1, 8]
Please enter a 4 digit number or exit
number: 8160
You have 0 bulls and 4 cows
Please enter a 4 digit number or exit
number: 6093
You have 0 bulls and 2 cows
Please enter a 4 digit number or exit
number: 6018
You have 2 bulls and 2 cows
Please enter a 4 digit number or exit
number: 6189
You have 0 bulls and 3 cows
Please enter a 4 digit number or exit
number: 2618
You have 3 bulls and 0 cows
Please enter a 4 digit number or exit
number: 0618
You have 4 bulls and 0 cows
Congratulations you have guessed the number in 6 guesses
goodbye
>>> |
```

Ln: 36 Col: 4

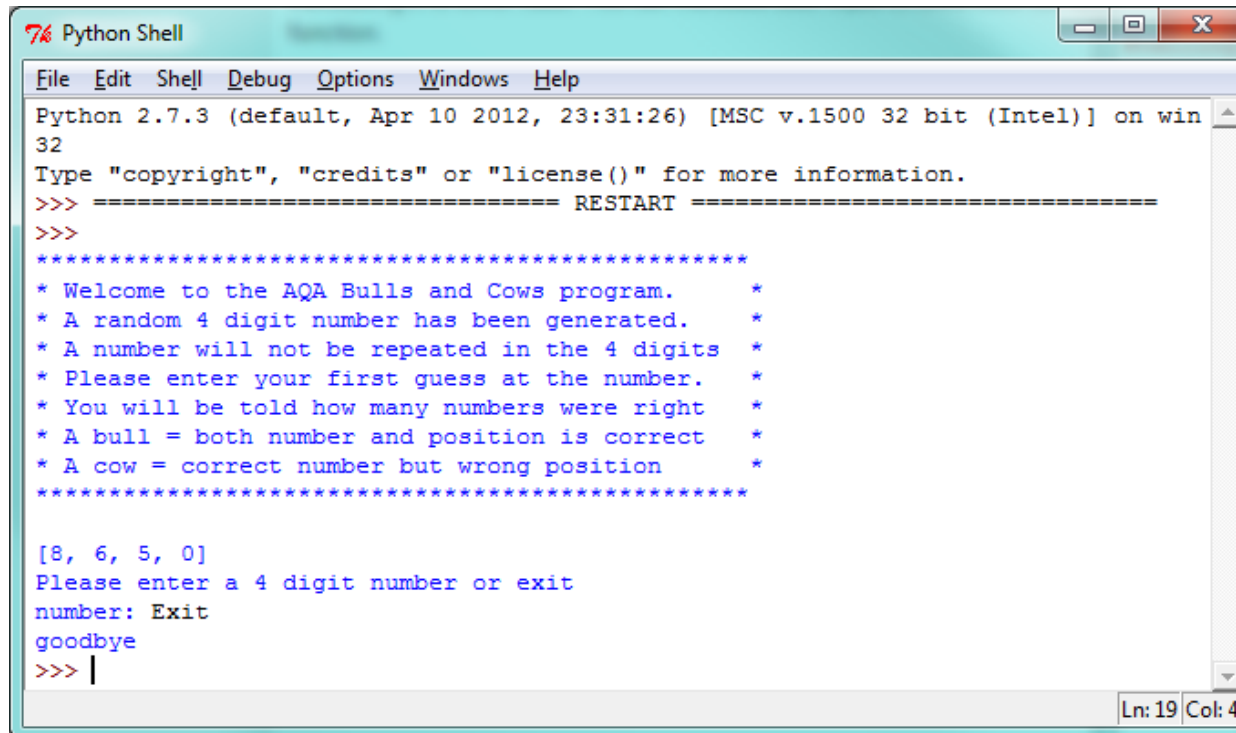
Figure 6a



```
*Python Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
*****
* Welcome to the AQA Bulls and Cows program. *
* A random 4 digit number has been generated. *
* A number will not be repeated in the 4 digits *
* Please enter your first guess at the number. *
* You will be told how many numbers were right *
* A bull = both number and position is correct *
* A cow = correct number but wrong position *
*****

[3, 7, 6, 5]
Please enter a 4 digit number or exit
number: Exit
you must enter a number
Please enter a 4 digit number or exit
number: |
Ln: 20 Col: 8
```

Figure 6b.

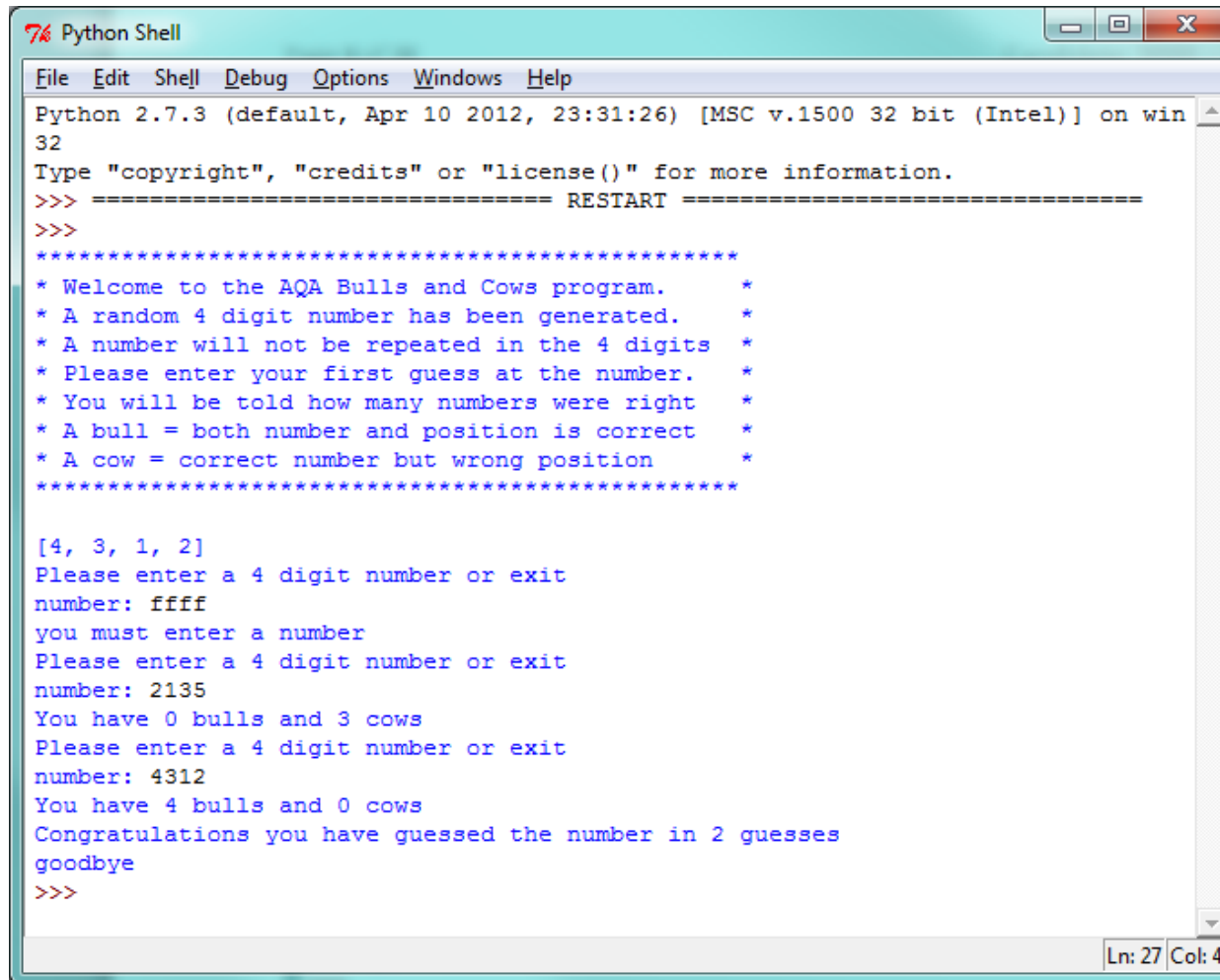


```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
*****
* Welcome to the AQA Bulls and Cows program. *
* A random 4 digit number has been generated. *
* A number will not be repeated in the 4 digits *
* Please enter your first guess at the number. *
* You will be told how many numbers were right *
* A bull = both number and position is correct *
* A cow = correct number but wrong position *
*****
[8, 6, 5, 0]
Please enter a 4 digit number or exit
number: Exit
goodbye
>>> |
Ln: 19 Col: 4
```

```
while user_input != "exit":
    print("Please enter a 4 digit number or exit")
    user_input = raw_input("number: ").lower()
```

Corrected code

Figure 7.



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
*****
* Welcome to the AQA Bulls and Cows program.      *
* A random 4 digit number has been generated.     *
* A number will not be repeated in the 4 digits  *
* Please enter your first guess at the number.    *
* You will be told how many numbers were right   *
* A bull = both number and position is correct   *
* A cow = correct number but wrong position      *
*****

[4, 3, 1, 2]
Please enter a 4 digit number or exit
number: ffff
you must enter a number
Please enter a 4 digit number or exit
number: 2135
You have 0 bulls and 3 cows
Please enter a 4 digit number or exit
number: 4312
You have 4 bulls and 0 cows
Congratulations you have guessed the number in 2 guesses
goodbye
>>>
Ln: 27 Col: 4
```

Potential enhancements and refinements

The solution meets user requirements, allowing the user to play the game and guess the random number correctly. The user input has been validated to ensure that the solution is robust. The validation includes the use of a try: catch when checking the format of the data input. This prevents the program from crashing if the input is not a number, so the user has an opportunity to re-enter the number, rather than restarting the program.

The code has comments that explain the functions and is structured logically, so making an efficient solution that can be easily maintained.

The main problem encountered was how the user input was handled and there are some alternatives that can be considered. In my solution I originally used the input() function. When I tested the solution this caused a problem when validating the input.

The python function input() reads the keyboard input and makes a decision on how to format the data based on what the user has typed. For example if the user types in a number such as 1234 then input() will set the data variable to an integer (so in this case user_input is an integer if 1234 is entered).

```
while user_input != "exit":  
    print("Please enter a 4 digit number or exit")  
    user_input = input("number: ")
```

The advantage of using input() is that python makes a decision without needing code to test if the number is an integer or a string. Code is shorter and less complex. When comparing the randomly generated number in my example the code will be comparing "like with like", as the random number returns an integer. This could have performance benefits as there is no need for extra instructions to the processor to convert data types.

This would be fine if I was comparing the random and user input numbers as a whole. However I needed to compare them digit by digit and an integer data type does not allow this. There are a few ways to do this in python but all involve converting the user input into a string and then mapping into a list. This would make the code more complex and less efficient as data would have to be evaluated twice and potentially converted twice.

Therefore I used the raw_input function that always assumes a string input. This could then be tested once to see if it was an integer, for validation. It could then be compared digit by digit to the randomly generated number that had been formatted as a list originally.

An Improvement to the functionality of the solution could be considered. For example it could be possible to save a game half way through, so that the user could resume guessing the number. This would require a way of saving the random number and number of guesses into some data store.

A history of games could also be created, and a way of entering user names and playing against another user, to compare guesses.

This was not required as part of this solution however.

Appendix – Complete code listing.

```

#-----#
# AQA Cows and Bulls Game #
# Written in Python 3.3   #
# January 2016           #
# Candidate: Harry Potter #
# Centre: Hogwarts       #
# Centre No: 000000      #
# Candidate No: 9999     #
#-----#
#import libraries
import random #imports random for random number

#Function to generate a random 4 digit number
def generate_number():
    #Creates a list to be used to guess the number
    #loop to generate 4 numbers. Continues until all numbers are unique
    gen_number = []
    i = 0
    while i <=3:
        random_value = random.randrange(0, 9)
        #IF statement to check that the number has not been used before
        if gen_number.count(random_value) == 0:
            gen_number.append(random_value)
            i+=1
    return gen_number

def validate_number(user_input):
    #validation to check that the user entry is a number and 4 digits in length
    #the user_input is also checked for duplicate numbers. If validation fails then
    # the function returns False.
    test_input = ""

    #try to convert the user input to an integer. If there are characters
    #in the input then a value error is returned.
    try:
        test_input = int(user_input)
    except ValueError:
        print("you must enter a number")
        return False

    #the length of the number must be 4 digits long. The format of the input
    #has already been checked so it is a number
    try:
        if len(user_input) != 4:
            print("your number should be exactly 4 digits long")
            return False
    except:
        pass
    i = 0
    while i <=3:
        #IF statement to check that the digit has not been used before
        #loop through each digit in the input string and count the number
        #of times it occurs which should only be once, otherwise it is a
        #duplicate
        if user_input.count(user_input[i]) != 1:
            print("your number contains duplicates")
            return False
        i+=1
    # end loop

```

```
    return True

def check_guess(user_input, gen_number):
    # initialise all the variables to count bulls and cows
    i = 0
    bulls = 0
    cows = 0

    #a loop that looks at each digit in the user input and compares it to the generated
    #number.

    while i <=3:
        #if the input digit is the same as the generated digit in the same position
        #then it is a bull

        if gen_number[i] == int(user_input[i]):
            bulls = bulls + 1

        #if the digit is in the generated number but in a different position
        #then it is a cow
        elif int(user_input[i]) in gen_number:
            cows = cows + 1
        i += 1
    # end of loop

    print ("You have " + str(bulls) + " bulls and " + str(cows) + " cows")

    #If there are 4 bulls then True is returned to the main program
    #so that a congratulations message can be printed
    if bulls == 4:
        return True
    else:
        return False

#-----#
# Main program          #
#-----#

#Introduction to the game
print("*****")
print("* Welcome to the AQA Bulls and Cows program.      *")
print("* A random 4 digit number has been generated.      *")
print("* A number will not be repeated in the 4 digits    *")
print("* Please enter your first guess at the number.      *")
print("* You will be told how many numbers were right     *")
print("* A bull = both number and position is correct      *")
print("* A cow = correct number but wrong position         *")
print("*****")
print("")

#calling the function that generates the number
gen_number = generate_number()
print(gen_number) # Just for testing. Comment this line out when playing for real

#sets a data variable for user input
user_input = ""
#an integer is created to keep count of the number of guesses
guesses = 0
```

```
#loop that allows a number to be entered until the program is exited
#or the number is guessed
while user_input != "exit":
    print("Please enter a 4 digit number or exit")
    user_input = input("number: ").lower()

    #breaks the loop if exit is entered, the game ends
    if user_input == "exit":
        break

    #checks to see if the number has been guessed correctly
    #the validation function also checks that a valid number has
    #been entered
    if validate_number(user_input) is True:
        guesses += 1
        if check_guess(user_input, gen_number) is True:
            print ("Congratulations you have guessed the number in " + str(guesses) + "
guesses")
            break

#end of while loop

print ("goodbye")
```