Lucky number revisited

Worked example from the last lesson Countdown

The program below displays a sequence of numbers, starting from **10** and counting down to 1. The **count** variable is used to keep track of the current number.

```
count = 10
while count >= 1:
  print(count)
  count = count-1
print("Lift off")
```

Worked example from the last lesson Times tables

The program below asks the user a series of times tables practice questions and provides feedback. The **questions** variable is used to keep track of how many questions have been asked.

```
from random import randint
questions = 0
while questions < 3:
  a = randint(2,12)
  b = randint(2,12)
  print(a, "times", b, "=")
  answer = int(input())
  product = a * b
  if answer == product:
    print("That is correct")
  else:
    print("I am sorry")
    print(a, "times", b, "is", product)
  questions = questions + 1
```

Task Guess the number

Open the Python program below (ncce.io/py-lucky-60) in your development environment. It picks a specific ‘lucky number’ and keeps asking the user to guess it.

```
1  lucky = 13
2  guessed = False
3  while guessed == False:
4    print("Can you guess my lucky number?")
5    guess = int(input())
6    if guess != lucky:
7      print("Sorry, it's not", guess)
8    else:
9      print("Amazing, you guessed it")
10 print("Nice playing with you")from random
11 import randint
```

The program uses a **flag variable** called **`guessed`** to keep track of whether or not the user has guessed the lucky number. The variable is initialised to **`False`** (line 2), but it is never set to **`True`**, so the game never terminates.

Step 1: Ending the game

Insert the following line in your program, **wherever you think it should be**.

```
guessed = True      # raise the flag
```

This assignment sets **`guessed`** to **`True`**. It ‘raises the flag’ to indicate that the user has guessed the number. This should cause the game to end when the condition in `while` is fulfilled.

Tip

Make sure that the `guessed` variable is set to `True` only in the case where the user guesses the number.

Step 2: Counting guesses

Extend the program, so that it keeps track of how many times the user has attempted to guess the lucky number.

At the end of the game, **display** this number to the user.

Example

Note: Use these numbers to test that your program works correctly. In general, the messages displayed will depend on user input and will not always be the same.

The program displays a prompt and waits for keyboard input.	`Can you guess my lucky number?`
The user types in a reply.	`12`
The program displays a message that the user's guess is incorrect.	`Sorry, it's not 12`
The program displays a prompt and waits for keyboard input.	`Can you guess my lucky number?`
The user types in a reply.	`13`
The program displays a message that the user's guess is correct.	`Amazing, you guessed it!`

The program displays the number of attempts.	`It took you 2 guesses`
	`Nice playing with you`

Tip

Introduce a `count` variable to keep track of the number of user guesses.

Look at the `count` and `question` variables in the worked examples: they serve the same purpose. They are assigned an initial value and modified in each iteration.

Step 3: A limit to the guesses

This is the condition currently checked in the `while` statement:

```
guessed == False
```

This means that the game will continue for as long as `guessed` is `False`, i.e. the user still hasn't guessed the lucky number.

Extend this condition, to also **check that the user has not exceeded a certain number of guesses**. For example, the user may only be allowed three guesses.

```
guessed == False and            :
```

Tip

Your program uses the `count` variable to keep track of how many times the user has attempted to guess the lucky number. Check this variable in the condition.

Look at how the `count` and `question` variables are checked in the `while` conditions of the worked examples.

Step 4: Final word

At the end of the game, the current program displays the number of attempts that the user made at guessing the number.

Extend the program so that at the end of the game:

- ☐ If the user managed to guess the lucky number, the program displays the number of guesses required (like it currently does)...
- ☐ ... and otherwise, if the user's guesses were incorrect, the program displays the lucky number to the user

Example

Note: This is an example of the user's **successful final attempt**. In general, the messages displayed will depend on user input and will not always be the same.

The program displays a prompt and waits for keyboard input.	`Can you guess my lucky number?`
The user types in a reply.	13

The program displays a message that the user's guess is correct, and another one with the number of guesses that were required.

```
Amazing, you guessed it!
It took you 2 guesses
```

Example

Note: This is an example of the user's **unsuccessful final attempt**. In general, the messages displayed will depend on user input and will not always be the same.

The program displays a prompt and waits for keyboard input.

```
Can you guess my lucky number?
```

The user types in a reply.

```
12
```

The program displays a message that the user's guess is incorrect, and another one with the actual lucky number.

```
Sorry, it's not 12
My lucky number is 13
```

Explorer task More information

In the current program, when a user's guess is unsuccessful, they are only informed that they didn't guess the lucky number.

It would be great if the program provided some **additional information**, such as whether the user should try a lucky number that is **higher or lower** than their current guess.

Example

Note: Use these numbers to test that your program works correctly. In general, the messages displayed will depend on user input and will not always be the same.

The program displays a prompt and waits for keyboard input.	`Can you guess my lucky number?`
The user types in a reply.	`12`
The program displays a message with additional information about the lucky number.	`My lucky number is larger than 12`

Example

Note: Use these numbers to test that your program works correctly. In general, the messages displayed will depend on user input and will not always be the same.

The program displays a prompt and waits for keyboard input.	`Can you guess my lucky number?`
The user types in a reply.	`16`
The program displays a message with additional information about the lucky number.	`My lucky number is smaller than 16`

Tip

When a user attempts to guess the lucky number, there are now **three possible outcomes**, so you will need multi-branch selection (`if`, `elif`, `else`).

Explorer task Randomness

In the current program, a specific lucky number is always selected.

```
lucky = 13
```

Modify this assignment, so that a random integer between 1 and 20 (inclusive) is selected as a lucky number.

```
from random import randint
lucky = 
```

Tip

You will need to use the `randint` function, from the `random` module.