



# Number Guessing Game

This is an exercise in writing a very simple program, a number guessing game. The aim is to get it working, test it, then add a new layer, repeating the process until a more complete program has been produced.

Each step consists of an instruction, followed by a comment on what else is needed, followed by an example solution. The comments refer to the solution, but the comments first in order to put space between the instruction and solution, to prevent you accidentally seeing the solution early.

## Step 1

### Instruction

Choose a number between 1 and 100, e.g., 47. Write a Python program that invites the user to guess a number between 1 and 100. If the user guessed correctly, print, "Well done, you guessed correctly!" Otherwise, print, "Sorry, that is not correct." Regardless of whether the user guessed correctly, print, "Thank you for playing," then exit.

### Comment

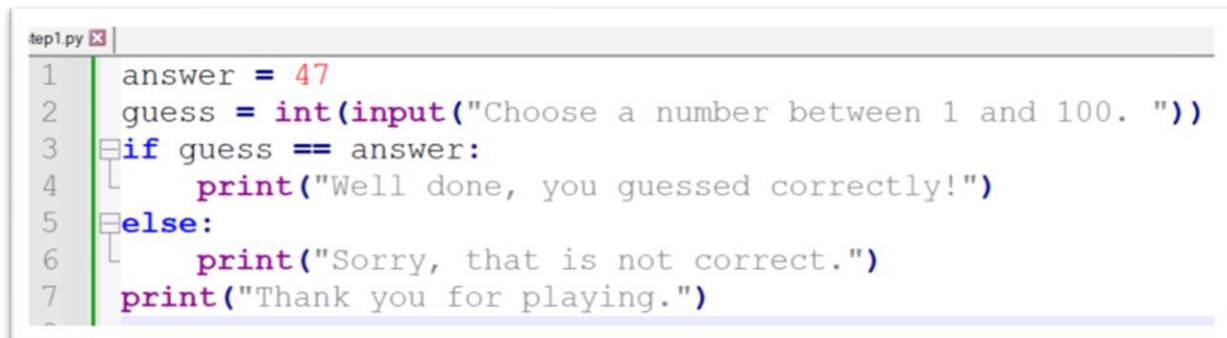
If you got this to work, well done. If you were not able to, don't worry, just check the solution below and have a read through.

Note that the input function returns a string rather than an integer, so on line 2 the call to input is wrapped in the int function. In this way, the string value is cast as an int, which is stored in the variable guess.

Lines 4 and 6 are indented. The indentations must be consistent. If line 4 is indented with a tab, and line 6 is indented with spaces, the program will not run.

Line 7 must not be indented because the print statement does not depend on whether the user guessed correctly.

### Solution



```
step1.py
1  answer = 47
2  guess = int(input("Choose a number between 1 and 100. "))
3  if guess == answer:
4      print("Well done, you guessed correctly!")
5  else:
6      print("Sorry, that is not correct.")
7  print("Thank you for playing.")
```



## Step 2

### Instruction

What's the problem with Step 1? Well, you only get one guess.

Change the code to allow the user an unlimited number of guesses.

### Comment

There are a number of ways of providing multiple guesses. As the instruction states "unlimited", it makes sense to use a while loop rather than, say, a for loop with a range.

There has to be a way of ensuring the loop will end when the correct answer has been guessed. You could use something like

```
while(guess != answer):
```

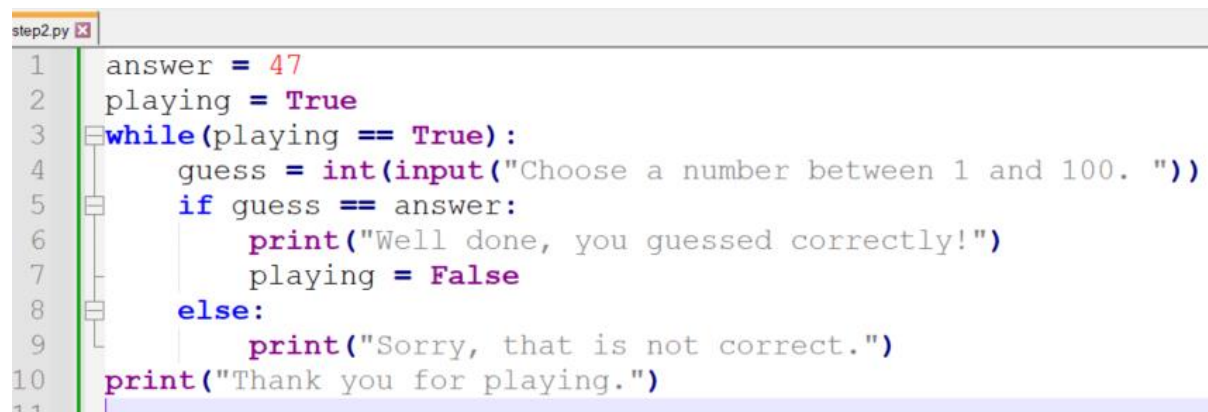
but I chose to introduce a flag variable called playing. This is a boolean with the initial value of True. On line 7, when the user has guessed the correct number, its value is changed to False, so when this iteration of the while loop is finished, the program exits the loop, executes line 10 and exits the program. (The "Thank you for playing" message is only printed when you have guessed correctly now.)

**Note:** There is redundancy in line 3. The value of playing can only be True or False; there is no need to compare it with True. We can shorten it to

```
while(playing):
```

This change will be made in the stages that follow.

### Solution



```
step2.py
1  answer = 47
2  playing = True
3  while(playing == True):
4      guess = int(input("Choose a number between 1 and 100. "))
5      if guess == answer:
6          print("Well done, you guessed correctly!")
7          playing = False
8      else:
9          print("Sorry, that is not correct.")
10 print("Thank you for playing.")
11
```

## Step 3

### Instruction

What's the problem with Step 2? You can't exit until you've guessed correctly! (Not unless you press CTRL-C, anyway.) Guessing a number is not the world's most exciting activity, and you may well get bored and decide to exit.

In the prompt, add

```
("Q to quit")
```



Ensure it is not case sensitive.

### Comment

We need to take the call to input out of the int function, because we need to check if the user has input a letter rather than a numeral. I've introduced a new variable called entry on line 4, as a wish to quit does not really count as a guess, but you don't have to do this; I could have stayed with the variable guess and line 7 would read

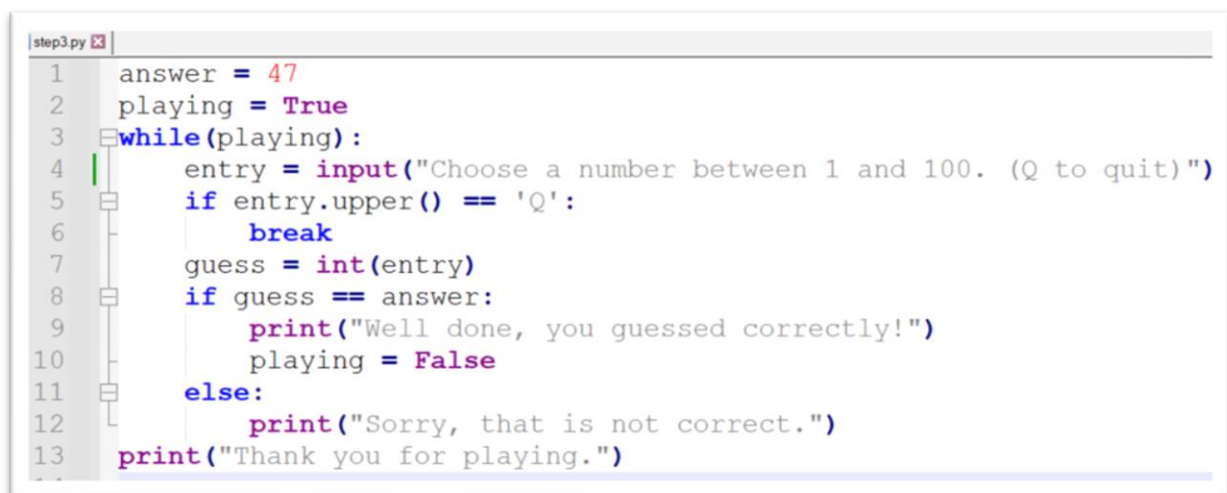
```
guess = int(guess)
```

On line 5 I used the upper method to make the letter a capital regardless of what the user entered. It would make just as much sense to do this on line 5 instead:

```
if entry == 'Q' or entry == 'q':
```

Shouldn't we have an else after the break on line 6? In fact, we don't need one, because if the condition on line 5 is true, we will break out of the loop, so we will skip all of the lines from lines 7-12.

### Solution



```
step3.py
1  answer = 47
2  playing = True
3  while(playing):
4      entry = input("Choose a number between 1 and 100. (Q to quit)")
5      if entry.upper() == 'Q':
6          break
7      guess = int(entry)
8      if guess == answer:
9          print("Well done, you guessed correctly!")
10         playing = False
11     else:
12         print("Sorry, that is not correct.")
13 print("Thank you for playing.")
```

### Step 4

#### Instruction

The problem with Step 3 is that you get no feedback. Wouldn't it be more interesting (or at least less boring) if you knew if your guess was too high or too low?

So, instead of simply saying, "Sorry, that is not correct," say, "Too high," or "Too low."

While you're at it, keep count of the number of guesses. If the user guesses correctly, print, "That took n attempts," where n is the count. (Refinement: What if it takes a single attempt?)

### Comment

The count variable must be initialised to zero before the loop begins.



It is incremented on line 9, because this is the earliest part of the program where we are sure that the user has made a valid guess. The += shorthand has been used, but you might prefer:

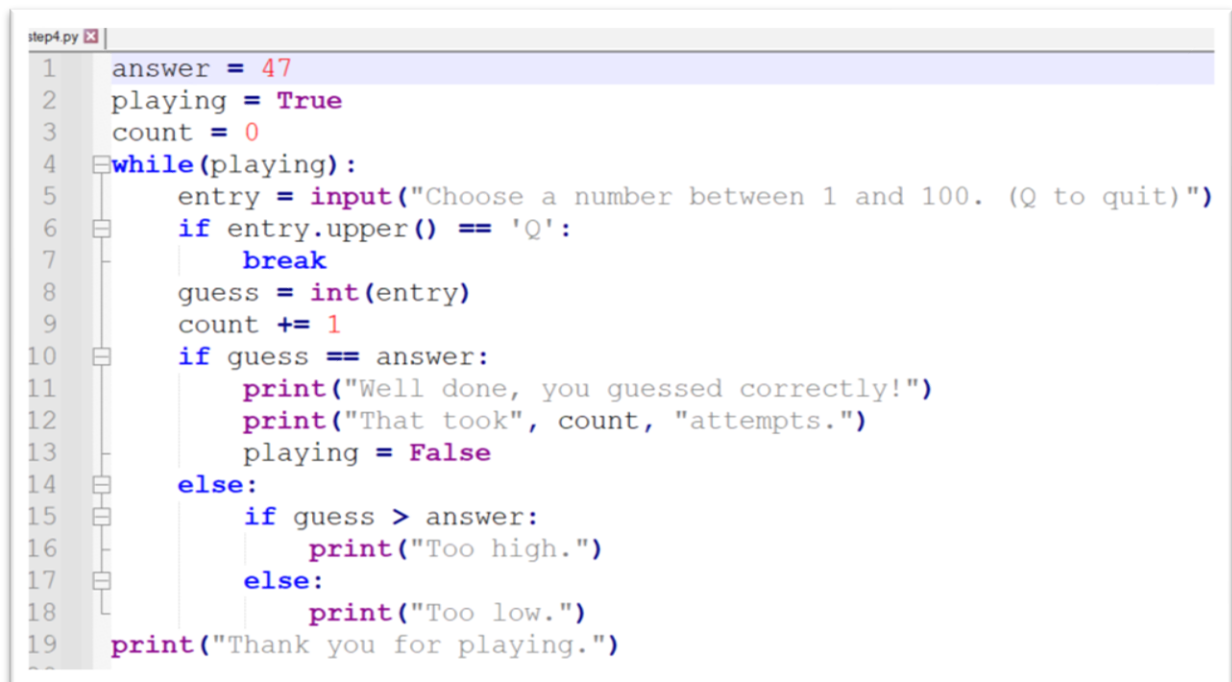
```
count = count + 1
```

On line 15, it checks for answers that are too high using:

```
if guess > answer:
```

but on line 17 it simply uses an else, as too low is the only other possibility; we've already taken care of guess == answer on line 10.

## Solution



```
1  answer = 47
2  playing = True
3  count = 0
4  while(playing):
5      entry = input("Choose a number between 1 and 100. (Q to quit)")
6      if entry.upper() == 'Q':
7          break
8      guess = int(entry)
9      count += 1
10     if guess == answer:
11         print("Well done, you guessed correctly!")
12         print("That took", count, "attempts.")
13         playing = False
14     else:
15         if guess > answer:
16             print("Too high.")
17         else:
18             print("Too low.")
19     print("Thank you for playing.")
```

## Step 5 (stretch)

This exercise goes beyond what is taught in the course.

### Instruction

Once the user has correctly guessed that the number is 47, there's nowhere else to go with this game. Wouldn't it be better if there was a different, and unknown, number each time you played?

Do a search online for "python random numbers" and see if you can achieve this.

### Comment

On line 1 I have imported the random package. Imports must be at the beginning of the file.

To use the functions in the random package, you must precede them with the word "random" followed by a dot.



The arguments in `randrange` are 1 and 101 rather than 1 and 100, because in Python, `range` always deals with one less than the second value.

### Solution

```
1  import random
2
3  answer = random.randrange(1, 101)
4  playing = True
5  count = 0
6  while(playing):
7      entry = input("Choose a number between 1 and 100. (Q to quit)")
8      if entry.upper() == 'Q':
9          break
10     guess = int(entry)
11     count += 1
12     if guess == answer:
13         print("Well done, you guessed correctly!")
14         print("That took", count, "attempts.")
15         playing = False
16     else:
17         if guess > answer:
18             print("Too high.")
19         else:
20             print("Too low.")
21     print("Thank you for playing.")
```