# **Number guessing game in python3**

Most of the geeks from a CS (**Computer Science**) background, think of their **very first project** after doing a Programming Language. Here, you will get your very first project and the basic one, in this article.

**Task:** *Below are the steps:*

* Build a Number guessing game, in which the **user selects a range.**
* Let’s say User selected a range, i.e., from**A**to **B**, where **A** and **B**belong to Integer.
* Some **random integer will be selected by the system** and the user has to guess that integer in the **minimum number of guesses**

**Analysis:**

**Explanation 1**: If the User inputs range, let’s say from 1 to 100. And compiler randomly selected 42 as the integer. And now the guessing game started, so the user entered 50 as his/her **first guess**. The compiler shows “Try Again! You guessed too high”. That’s mean the random number (i.e., 42) doesn’t fall in the range from 50 to 100. That’s the importance of guessing half of the range. And again, the user guesses half of 50 (Could you tell me why?). So the half of 50 is 25. The user enters 25 as his/her **second guess**. This time compiler will show, “Try Again! You guessed too small”. That’s mean the integers less than 25 (from 1 to 25) are useless to be guessed. Now the range for user guessing is shorter, i.e., from 25 to 50. Intelligently! The user guessed half of this range, so that, user guessed 37 as his/her **third guess**.  This time again the compiler shows the output, “Try Again! You guessed too small”. For the user, the guessing range is getting smaller by each guess. Now, the guessing range for user is from 37 to 50, for which the user guessed 43 as his/her **fourth guess**. This time the compiler will show an output “Try Again! You guessed too high”. So, the new guessing range for users will be from 37 to 43, again for which the user guessed the half of this range, that is, 40 as his/her **fifth guess**.  This time the compiler shows the output, “Try Again! You guessed too small”. Leaving the guess even smaller such that from 41 to 43. And now the user guessed 41 as his/her **sixth guess**. Which is wrong and shows output “Try Again! You guessed too small”. And finally, the User Guessed the right number which is 42 as his/her **seventh guess**.

**Total Number of Guesses = 7**

**Explanation 2**: If the User inputs range, let’s say from 1 to 50. And compiler randomly selected 42 as the integer. And now the guessing game started. So the half of 50 is 25. The user enters 25 as his/her **First guess**. This time compiler will show, “Try Again! You guessed too small”. That’s mean the integers less than 25 (from 1 to 25) are useless to be guessed. Now the range for user guessing is shorter, i.e., from 25 to 50. Intelligently! User guessed half of this range, so that, user guessed 37 as his/her **second guess**.  This time again the compiler shows the output, “Try Again! You guessed too small”. For the user, the guessing range is getting smaller by each guess. Now, the guessing range for user is from 37 to 50, for which the user guessed 43 as his/her **third guess**. This time the compiler will show an output “Try Again! You guessed too high”. So, the new guessing range for users will be from 37 to 43, again for which the user guessed the half of this range, that is, 40 as his/her**fourth guess**.  This time the compiler shows the output, “Try Again! You guessed too small”. Leaving the guess even smaller such that from 41 to 43. And now the user guessed 41 as his/her fifth guess. Which is wrong and shows output “Try Again! You guessed too small”. And finally, the User Guessed the right number which is 42 as his/her sixth guess.

**Total Number of Guesses =** 6

So, the minimum number of guesses depends upon range. And the compiler must calculate the minimum number of guessing depends upon the range, on its own. For this, we have a formula:-

*Minimum number of guessing = log2(Upper bound – lower bound + 1)*

**Algorithm: Below are the Steps:**

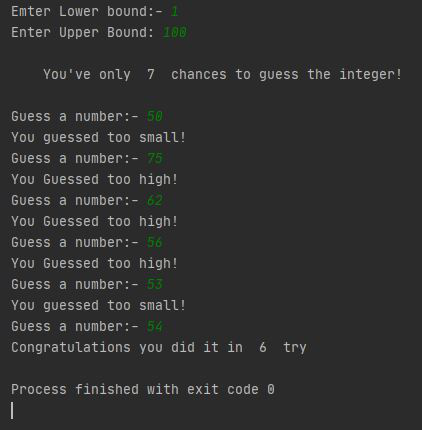
* User inputs the **lower bound** and **upper bound** of the range.
* The compiler generates a random integer between the range and store it in a variable for future references.
* For repetitive guessing, a while loop will be initialized.
* If the user guessed a number which is greater than a randomly selected number, the user gets an output “*Try Again! You guessed too high*“
* Else If the user guessed a number which is smaller than a randomly selected number, the user gets an output “*Try Again! You guessed too small”*
* And if the user guessed in a minimum number of guesses, the user gets a “*Congratulations!* ” Output.
* Else if the user didn’t guess the integer in the minimum number of guesses, he/she will get “*Better Luck Next Time!*” output.

**Below is the Implementation of the Algorithm:**

**Python3**

|  |
| --- |
| import random  import math  # Taking Inputs  lower = int(input("Enter Lower bound:- "))    # Taking Inputs  upper = int(input("Enter Upper bound:- "))    # generating random number between  # the lower and upper  x = random.randint(lower, upper)  print("\n\tYou've only ",         round(math.log(upper - lower + 1, 2)),        " chances to guess the integer!\n")    # Initializing the number of guesses.  count = 0    # for calculation of minimum number of  # guesses depends upon range  while count < math.log(upper - lower + 1, 2):      count += 1        # taking guessing number as input      guess = int(input("Guess a number:- "))        # Condition testing      if x == guess:          print("Congratulations you did it in ",                count, " try")          # Once guessed, loop will break          break      elif x > guess:          print("You guessed too small!")      elif x < guess:          print("You Guessed too high!")    # If Guessing is more than required guesses,  # shows this output.  if count >= math.log(upper - lower + 1, 2):      print("\nThe number is %d" % x)      print("\tBetter Luck Next time!")    # Better to use This source Code on pycharm! |

**OUTPUT:***Below is the output of the above Program*



*OUTPUT FOR THE GUESSING GAME*

**Time Complexity:**   
The time complexity of this code is **O(n)** as the number of iterations of the loop is not fixed and depends on the number of guesses taken by the user to get the right answer.

**Space Complexity:**  
The space complexity of this code is **O(1)** as all the variables used in this code are of the same size and no extra space is required to store the values.