Lesson 4: Number Guessing Game

In this lesson, you'll learn how to use *random numbers* and *loops* in your programs.

Loops in programming allow us to tell a program to repeat instructions over and over until a certain condition has been met.

Random numbers in programming are a lot like rolling dice.
By using random numbers, we can create games and other interesting programs, because the user doesn't know what a certain number will be.

We'll use both of these concepts to create a number guessing game!

PART 1: Importing Random

First, before we can actually create random numbers, we have to import a *module*.

In Python, a module is a set of functions that can be used in other programs.
Importing a module allows us to use any of its functions later in our program.

Try typing in the line below:

```
import random
```

By itself, using the `import` command won't do anything in our program, but it will allow us to use the module's functions later.

PART 2: Creating Variables

After we've imported the random module, we can start creating variables for our game.

Try adding the following code to your program:

```
import random

won = False

tries = 0

number = random.randint(1, 100)
```

The code `won = False` creates a variable called `won`, and sets the value to `False`.

As well as numbers or strings, variables can also be set to *Boolean values*.
Boolean values are either `True` or `False`, just like a True/ False question on a test.

Setting the variable won doesn't do anything just yet, but we'll use it later in our program, after we create a loop

We'll use the variable `tries` to count the number of guesses the user has made.

The code `number = random.randint(1,100)` creates a variable called number, and sets its value to a random *integer* (whole number) between 1 and 100.

The `.` between random and randint tells the program to use the function randint from the module random, to make a random number.

PART 3: Using Loops

Next, we want to let the user start playing the game.

In order to let the user guess more than once, we'll use a `while` loop in our program.

Try typing in these lines at the end of your program:

```
number = random.randint(1, 100)

print("I'm thinking of a number between 1 and 100...")

while not won:

    guess = int(input())
```

Make sure to include the `:` at the end of the while line. Also make sure to put 4 spaces to the left of the `guess` line (you can use the tab key).

Loops allow us to run a section of code (a block), more than once.
When a block is used with a loop, first the program checks if the condition is true, then if it is, all of the block's code runs, and then this process repeats until the condition fails.

Here, our program will keep asking the user for a number as long as they haven't won.

PART 4: Allowing the User to Win

So far, our program will keep asking for a number forever, even if they guess correctly. This is because the while loop's condition (`not won`) is still `True`.
A `while` loop will check if its condition is `True`, and repeat the next block so long as it is.

In order to start fixing this, we can make a way for the program to exit out of the loop.

Try typing the following code into the while loop's block:

```
while not won:
    guess = int(input())
    tries = tries + 1
    if(guess == number):
        won = True
```

⚠ Make sure the lines you add are indented properly!

In this code, we're first adding one to the current value of the variable `tries`.

In programming (unlike math), when we type `tries = tries + 1`, we aren't saying that the number of tries is equal to `tries + 1` (which would be impossible). Instead we're telling our program to set the variable `tries` to whatever it currently is, plus one.

After this, we're using an `if` statement to ask if the user's guess is equal to the number.

If it is, we'll set the variable `won` to have a value of `True`, which will cause the `while` loop to stop repeating (since we've now won).

PART 5: Giving the User Clues

Right now, even though it's possible to win, it will probably take a very long time, because the program isn't telling the user whether they are guessing too low or too high.

In order to fix this, we can add more code to the `while` loop that compares the value of the guess and the number in a different way.

Try adding the following code (still in the while loop) to the end of your program:

```
while not won:

    .  .  .

    if(guess == number):

         won = True

    if(guess > number):

        print("You guessed too high!")

    if(guess < number):

        print("You guessed too low!")
```

Now, when you test your program, you should be able to win the game more easily!

PART 6: Congratulations Message

All that we're missing now is a message congratulating the player.

We'll also tell the player how many tries it took to guess correctly.

Try adding this code to the end of your program (this time, out of the while loop):

```
while not won:

    .  .  .

    if(guess < number):

        print("You guessed too low!")

print("Correct! The number was " + (str)(number))

print("You guessed the number in " + (str)(tries) + " tries.")
```

Here, we're converting `number` and `tries` to strings inside the `print` function. This is similar to converting the variable `yearly` in Lesson 2.

Now you should have a completed number guessing game that you can play!

PART 7: BONUS: Congratulating the Player

We've already told the player how many tries it took them to guess correctly, but we can also congratulate them based on how quickly they got the right number!

In order to do this, we'll want to look at the variable `tries`, and compare its value against a few different numbers.

The lower the number of tries, the better the message we'll give the player!

Try adding the following code to the end of your program:

```
print("You guessed the number in " + (str)(tries) + " tries.")

if(0 < tries <= 5):

    print("Incredible!")

if(5 < tries <= 8):

    print("Good job!")
```

This code will check if `tries` is more than `0`, and less than/ equal to `5`.

If it is, the program will print "`Incredible!`"

Afterwards, it will check if `tries` is more than `5`, and less than/ equal to `8`.

If that's the case, the program will print "`Good job!`"

You can also give the player a rank – such as “You have received the rank of Exalted Mind-Reader” if the player guesses in 5 or less tries, “Apprentice Mind-Reader” if solving in 5-7, etc.

Unlike many languages, Python is flexible about comparing variables.

```
                        if(0 < tries <= 5)
is the same as saying   if(0 < tries and tries <= 5)
```

Try adding more `if` statements to create even more congratulation messages!