Today, we are going to be doing a guessing game where the computer generates a random number and then we guess it. The aim of this exercise is to understand what basics of functions and if/else statements and loops.

The first step is to have the computer generate a secret number for us. To do that, we are going to

<code>import random</code>

Whenever we call import random or we import any other library or package, what happens is we are bringing in all the functionality that the library possesses into our current code then we can use the different methods that the package possesses so basically you’re using code that someone else wrote by importing (cool right?). You can see all the python packages and more <a href=<https://docs.python.org>>here</a>

Now when we see the <code>def</code> keyword in python code, it means we are defining a function. A <strong>function</strong> is a block of organized, reusable code. The aim of functions is to write DRY (don’t repeat yourself) code. Python gives you default functions such as <code>print()</code> but you can write your own. To make your own function, just use the syntax

<code>

def function\_name (parameters):</code>

“function\_name” is the name you are giving to your function and the parameters inside of the parenthesis stands for input parameter to the function (It can be more than one). After the parameters parenthesis you have a colon (:) and the indented function block starts right after it. The indent helps to know the block of code that belongs to a function or in some cases (as we would soon see) a statement. (Insert code)

When you define functions, it just sits there really. In order to make use of the function, you have to then <em>call</em> the function. To call a function, all you have to do is write the name of the function and then add the parenthesis after (if the function needs input parameter, add it in the parenthesis). The addition of the parenthesis is what calls it basically.

<code>guess(10) #calls the function which prints out 10</code><br />

That is a lot to take in so you can pause for now and read it again or you can continue.<br />

The second step in our computer guessing game is to make uses of what we discussed above and make use of a function. In this case we are naming our function guess and it takes in a parameter x i.e. <code>def guess(x)</code> Inside this function, we want the computer to generate a random number. To do this, we are going to make use of the random we imported and say <code>random.randint(lower, higher)</code> The randint() is a function basically that has been defined inside of the random package that we imported and it takes in two parameter – the lower value and the higher value. To make use of it, we are calling it hence the parenthesis and parameters. In our case, the lower value is going to be one (1) and our higher value is going to be x meaning when we call the function, we would pass in the higher value. Let’s say when we call our function, we say <code>guess(20)</code> what happens is that the random would generate a number between 1 and 20.

<Insert code>

Alright, now that the computer has generated a random number, we want to be able to guess that number and the computer would tell us if we are right or wrong. For this we would need a loop. There are 2 loops common to different languages which are for loop and while loop. A <strong>for</strong> loop is a definite loop and a <strong>while</while> loop is indefinite.

(For loop code)

The above snippet of code is a for loop. We have a list of items and we loop through it using a for loop. What happens is that the block of code indented under the for loop is run for every single item in the list so each time the code runs “item” changes to the next item in the items list.

(While loop code)

While loop is indefinite. It works kind of as a fact checker that returns true or false. Until the condition becomes false, the indented code below it keeps running.

To be able to check if our number we are guessing is equal to the number generated by the computer, we are going to make use of a while loop because we want to keep guessing until our number equals the computer number. The computer is guessing a random number between 1 and whatever we want so to use the while loop we can set <code>guess = 0</code> and then use <code> while guess != random\_numer:</code> The random number would never be zero (0) so on the first instance of the loop running, the outcome is always false. Then inside the loop, we can ask the user to input a number. The input is wrapped around an <code>int()</code> because input returns a string and we want to convert that input to an integer. After the user guesses a number, the guess variable is no longer zero (0) and becomes the guessed number. The while loop runs and the user continues guessing until the guess == random\_number and then once the condition given is false, the loop breaks and runs the next line of code outside the loop.

(insert code)

As the loop is running, we want to be able to provide some feedback to the user to help them guess. We want them to know if their guess was too high or if it was too low. To do this we are going to make use of <strong>if…else</strong> statements as shown below

<insert if/else code>

If…else statements are used in decision making. The indented block of code under the if statement is run only if the if statement is true. If the statement is false, the block of code would not run. If you want multiple decision, you can make use of if…else statements. If the <code>if</code> statement is false, the <code>else</code> statement runs. You can have more than 2 decisions and then introduce the <code>elif</code> elif is short for “else if”.

(insert victory code)

After we guess correctly, the while loop fails and then the next line of code runs which prints out you guessed right. <br />

It is important to note that the guess function does not run until we call it in the code as show above in line 15.<br />

<img src=”/images/guess.jpg”>

I really hope this helps in your understanding of functions, loops and if…else statements.<br />

Cheers 🥂 <br />

(“=” -> this is an assignment symbol e.g. when naming variables, “==” -> this is a comparison symbol i.e. left side equal to right side, “!=” -> not equal to, “<” -> left less then right, “>” -> left greater than right)