



Lab 115: Bucles

Link: <https://awsrestart.instructure.com/courses/1632/modules/items/886826>

Repo: <https://github.com/francopig/aws-python/tree/main/08>. Trabajo con bucles

Lab overview

A loop is a segment of code that repeats. You will be introduced to two types of loops: the `while` loop and the `for` loop.

In this lab, you will:

- Use a `while` loop
- Use a `for` loop

Exercise 1: Working with a while loop

A `while` loop makes a section of code repeat until a certain condition is met. In this exercise, you will create a Python script that asks the user to correctly guess a number.

Printing the game rules

1. From the navigation pane of the IDE, choose the `.py` file that you created in the previous *Creating your Python exercise file* section.
2. Use the `print()` function to inform the user about your game:

```
print("Welcome to Guess the Number!")
print("The rules are simple. I will think of a number, and you will try to guess it.")
```

3. Save and run the file.
4. Confirm that the script runs correctly and that the output displays as you expect it to.

The screenshot shows an IDE window with a file named `lab115.py`. The code in the editor is:

```
1 print("Welcome to Guess the Number!")
2 print("The rules are simple. I will think of a number, and you will try to guess it.")
```

Below the editor, there is a terminal window showing the output of the script:

```
Welcome to Guess the Number!
The rules are simple. I will think of a number, and you will try to guess it.
```

Importing random and writing a while loop

You will use the `import` command to include code that someone else wrote. Up to now, you have been using built-in functions. Recall that a function is a piece of reusable code.

1. At the top of the file, include the Python module (which is a type of library) called `random`.

Note: By convention, import statements are placed at the top of the script.

```
import random
```

2. Place the cursor on the next line after the second `print()` statement. Next, enter a statement that will generate a random number between 1 and 10 by using the `randint()` function of the `random` module.

```
number = random.randint(1,10)
```

3. Track whether the user guessed your number by creating a variable called *isGuessRight*:

```
isGuessRight = False
```

4. To handle the game logic, create a `while` loop:

```
while isGuessRight != True:
    guess = input("Guess a number between 1 and 10: ")
    if int(guess) == number:
        print("You guessed {}. That is correct! You win!".format(guess))
        isGuessRight = True
    else:
        print("You guessed {}. Sorry, that isn't it. Try again.".format(guess))
```

Note: The while loop will repeat the code inside the loop until the number is guessed correctly, which is represented by the condition `isGuessRight != True` in the code. Additionally, Python uses indentation to determine logic blocks, or what statements are considered to be part of the while loop. You can indent a line by placing the cursor next to a statement and pressing TAB.

5. Save the file.

```
1 import random
2
3 print("Welcome to Guess the Number!")
4 print("The rules are simple. I will think of a number, and you will try to guess it.")
5
6 number = random.randint(1,10)
7 isGuessRight = False
8
9 while isGuessRight != True:
10     guess = input("Guess a number between 1 and 10: ")
11     if int(guess) == number:
12         print(f"You guessed {guess}. That is correct! You win!")
13         isGuessRight = True
14     else:
15         print(f"You guessed {guess}. Sorry, that isn't it. Try again.")
```

programacion/lab115/lab1 x

Run Command: programacion/lab115/lab115.py

Welcome to Guess the Number!
The rules are simple. I will think of a number, and you will try to guess it.
Guess a number between 1 and 10: 1
You guessed 1. Sorry, that isn't it. Try again.
Guess a number between 1 and 10: 2
You guessed 2. Sorry, that isn't it. Try again.
Guess a number between 1 and 10: 3
You guessed 3. That is correct! You win!

Writing pseudocode

Before you run the Python script, write out the logic of the `while` loop in written (non-code) sentences. This technique is called *pseudocoding*.

For example:

- If the user has not guessed the correct answer, enter the loop.
- Ask the user for a guess.

- Is the guess the correct number?
- If the correct guess, tell the user it was the correct guess and exit the loop.
- If the wrong guess, tell the user it was the wrong guess and continue the loop.

Running the script

Now run the Python script and see if it works.

1. Run the file.
2. Confirm that the script runs correctly and that the output displays as you expect it to.

Adding comments

It is helpful to write comments in the code. A comment line is ignored by Python, and it starts with a pound sign (#). On most keyboards, you can create this symbol by pressing SHIFT+3. Add your own comments to the code to help you remember what the code does.

Informing the user about the script

In this activity, you will start a new Python script by creating the initial output that informs the user about what the script will do.

1. From the menu bar, choose **File > New From Template > Python File**.
2. Delete the provided sample code from the template file.
3. Choose **File > Save As...** and save it as *for-loop.py*.
4. To inform the user about your script, use the `print()` function:

```
print("Count to 10!")
```

5. Save and run the file.
6. Confirm that the script runs correctly and that the output displays as you expect it to.

Writing the for loop

In Python, you can include a large amount of functionality in a few words. This feature makes Python relatively easy to write compared to other programming languages, but it can also make Python code more difficult to read. In this activity, you will use the `for` statement, but you will also spend some time analyzing it after you see it run.

1. Return to the Python script. To count to 10, enter the following code.

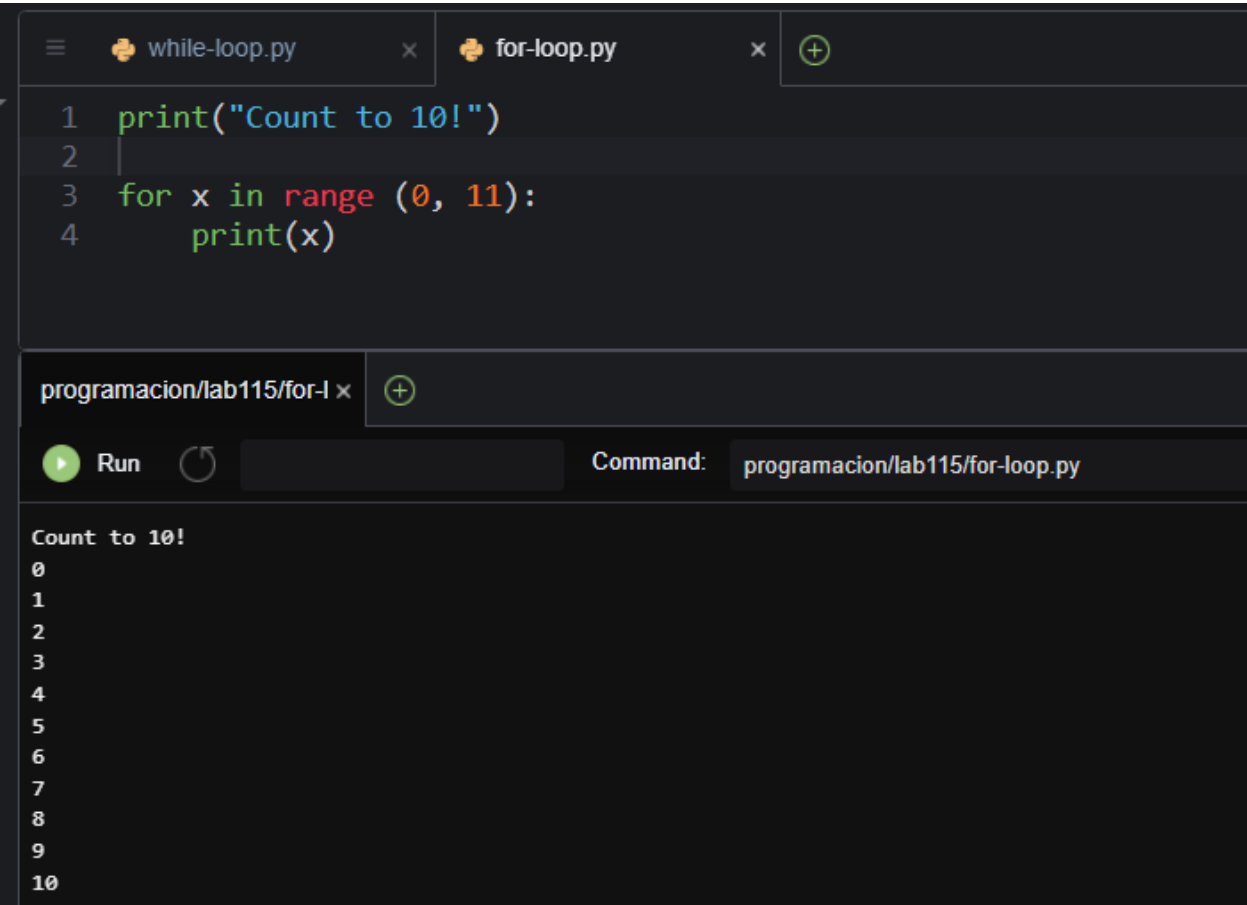
Note: Python uses indentation to determine that the print statement is inside the for loop statement.):

```
for x in range (0, 11):  
    print(x)
```

2. Save and run the file.
3. Confirm that the script runs correctly and that the output displays as you expect it to.

Here is an explanation of what happened in those two lines. The `for` statement uses the `for ... in` keywords to tell the computer to go through the list. A list is generated by the `range()` function. The `range()` function takes a starting number and an ending number, but the ending number is not inclusive. Therefore, you pass in `11` to have the function stop counting at 10. The letter `x` acts as a variable. Each time through the loop, the variable `x` is assigned to the next variable in the loop and is printed out to the screen.

Congratulations! You have worked with `while` and `for` loops in Python.



The screenshot shows a code editor with two tabs: 'while-loop.py' and 'for-loop.py'. The 'for-loop.py' tab is active, displaying the following code:

```
1 print("Count to 10!")
2
3 for x in range (0, 11):
4     print(x)
```

Below the code editor, there is a terminal window with the command 'programacion/lab115/for-loop.py' and a 'Run' button. The output of the command is displayed in the terminal:

```
Count to 10!
0
1
2
3
4
5
6
7
8
9
10
```

End Lab

Congratulations! You have completed the lab.

1. Choose **End Lab** at the top of this page, and then select Yes to confirm that you want to end the lab.
A panel indicates that *DELETE has been initiated... You may close this message box now.*
2. A message *Ended AWS Lab Successfully* is briefly displayed, indicating that the lab has ended.