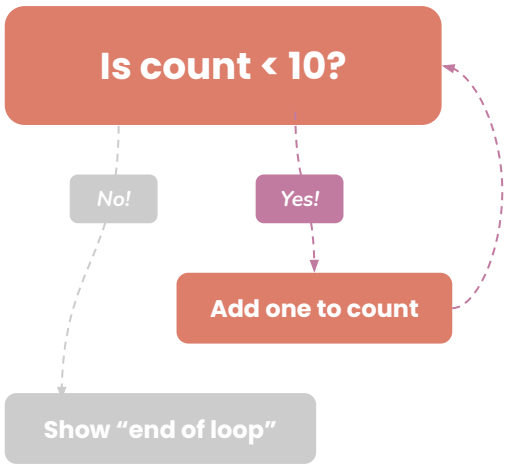
WEEK 4:
ITERATION

Iteration in Python

This week we looked at two forms of iteration: count-controlled iteration and condition-controlled iteration. Iteration is the process in Python of repeating a block of code repeatedly. This can either be based on a condition, or for a specific number of times.

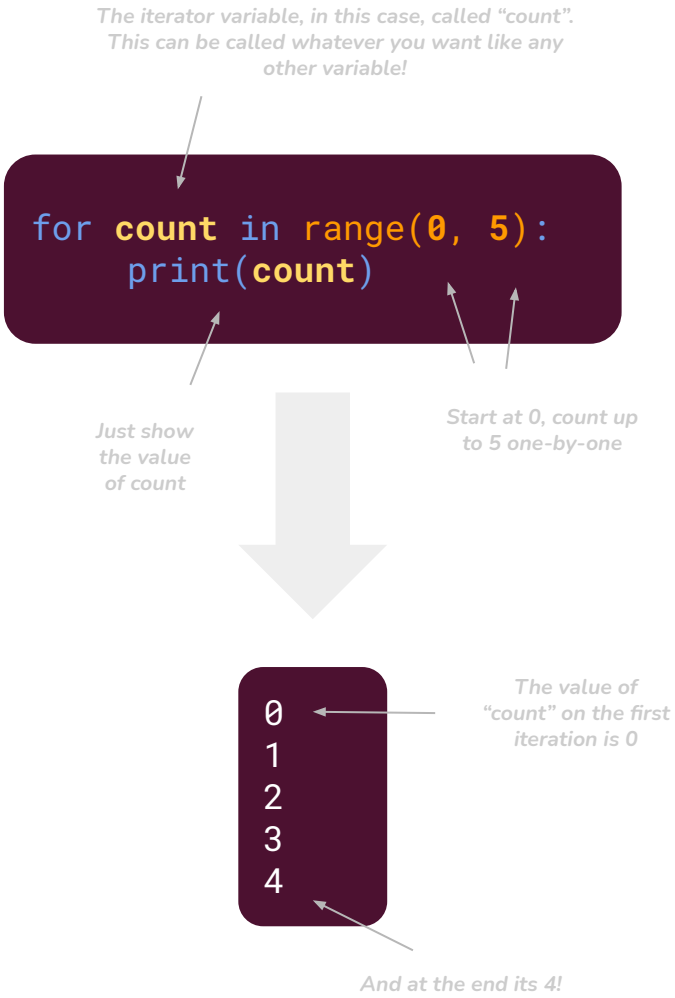
In condition-controlled iteration, a block of code is run WHILE a certain condition is true. An example of this is the ringtone on a phone: it is played whilst the phone is unanswered.

To do condition-controlled iteration, we can use the **while** keyword. We pass a condition, and the code inside is run repeatedly whilst it is true. An example can be seen of this process on the right!



Experiment with while loops in Python:

1. Can you create a loop which counts up to 10 and stops, using ONLY a while loop? Think about what the condition would have to be, and what would have to happen at each iteration.
2. You're saving up for a guitar and you need to write a tool to help you. Create two variables: **savings** and **guitar_price**. Set savings to 0, and guitar_price to something high like 1250. Every iteration, add a value such as 230 to savings. Here each iteration can be thought of as each month. Can you stop the loop when the amount of savings has reached the price of the guitar? Can you also print out the number of months needed to save to buy the guitar? (You might need another variable)
3. Is it possible that a condition is never satisfied? What happens in this case? Try write a while loop where the condition will always be false. (e.g. make a variable **a** = 5, make the condition **a < -3** (which is false, because a = 5). Does it run the code inside?
4. What about a while loop where the condition is always satisfied? Can you think of where this might be the case, and test it out? (You may need to press CTRL+C to forcibly stop the program)



Count-controlled iteration

The other type of iteration that is mainly used in Python is count-controlled iteration. In condition controlled iteration, a condition controls whether they loop continues. In count-controlled iteration however, a count is used to control whether the loop continues. Count-controlled iteration is useful when you want to repeat a certain action a certain number of times. If you know in advance how many times you want the code to loop, the count-controlled iteration is most likely what you're looking for.

In Python, count-controlled iteration is achieved by using the **for** keyword. These are referred to informally as "for loops". For loops offer a way of repeating a block of code a certain number of times -- which can be really useful to us as programmers!

One thing that often confuses new programmers is that for loops have a variable which takes on the value of the current iteration. For loops (usually) work by starting at 0 and counting up in increments of 1, until a certain value is reached. The variable which takes on the iteration number throughout a for loop is normally called the "iterator variable", "iterator" or sometimes just "i".

Experiment with for loops in Python:

1. Write a for loop to print the line "hello world!" 10 times.
2. Use the example showed at the top of this page to loop from 1 to 5 and print out the numbers along the way. Can you modify this to:
    a. Instead of counting 0 to 4, count 1 to 5?
    b. Count backwards, starting at 5, to 0. When 0 is reached, print "blast off!"
    c. What happens if you use range(1, 20, 2)? What about range(1, 20, 5)? What do you think 2 or 5 signifies in this bit of code?
3. Print out the 5 times table, from 5 x 1 all the way up to 5 x 12. Your solution should have the answers too (e.g. 5 x 5 = 25). You will have to make use of string concatenation, the multiplication operator and the iterator variable.

NEXT WEEK: FUNCTIONS