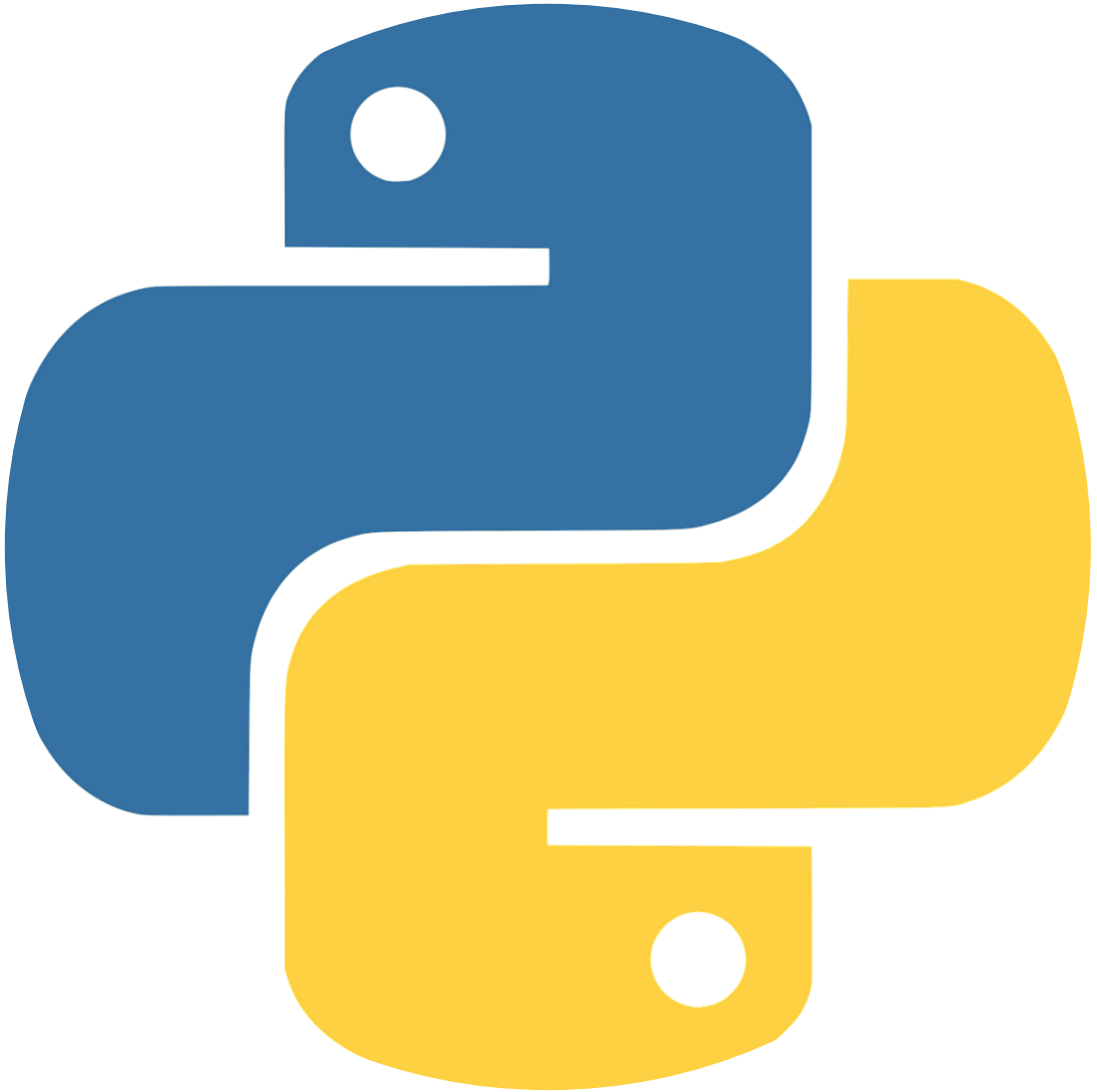
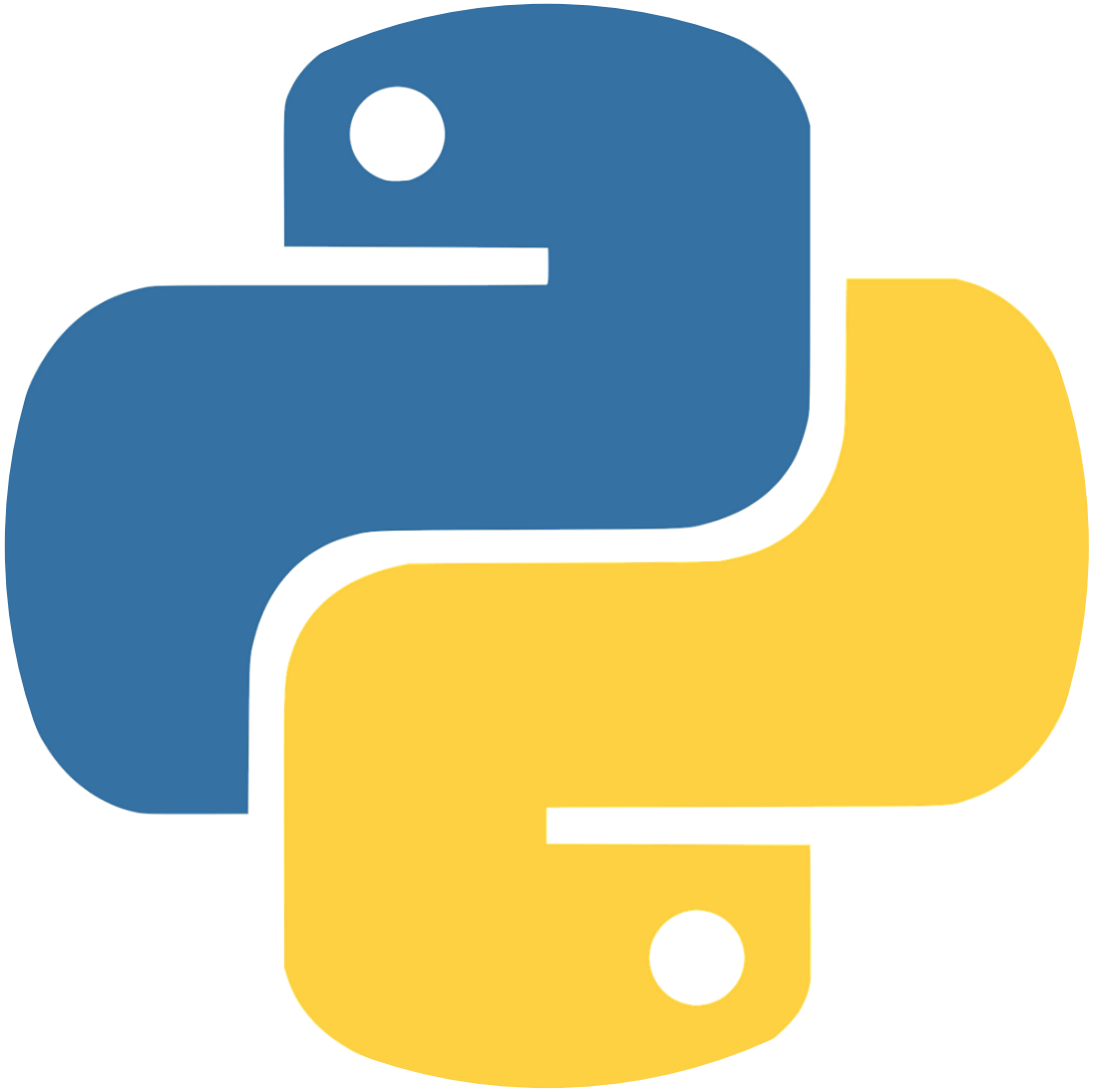
Loops



Loops

Computers are great at doing boring tasks without complaining. Programmers aren't, but they are good at getting computers to do repetitive work for them—by using loops.

A loop runs the same block of code over and over again. There are several different types of loop.



For Loops

For loops

When you know how many times you want to run a block of code, you can use a **for** loop. In this example, Emma has written a program to make a sign for her door. It prints "Emma's Room—Keep Out!!!" ten times. Try out her code for yourself in the shell. (After typing the code and hitting enter/return, press backspace to remove the indent and then hit enter/return again.)

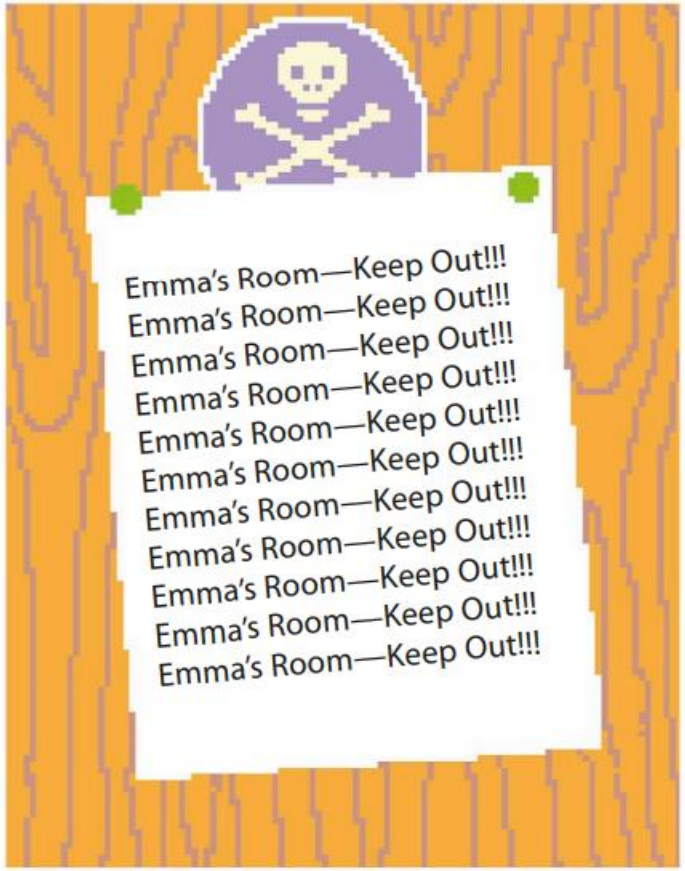
This is the loop variable.

The loop runs 10 times.

```
>>> for counter in range(1, 11):
        print('Emma\'s Room - Keep Out!!!')
```

Indent the commands in the body 4 spaces.

The line that gets repeated is called the loop body.



For loops

▽ **Loop variable**

The loop variable keeps track of how many times we've gone around the loop so far. The first time round it's equal to the first number in the list specified by `range(1, 11)`. The second time around it's equal to the second number in the list, and so on. When we've used all the numbers in the list, we stop looping.

First loop

Loop variable = 1

Second loop

Loop variable = 2

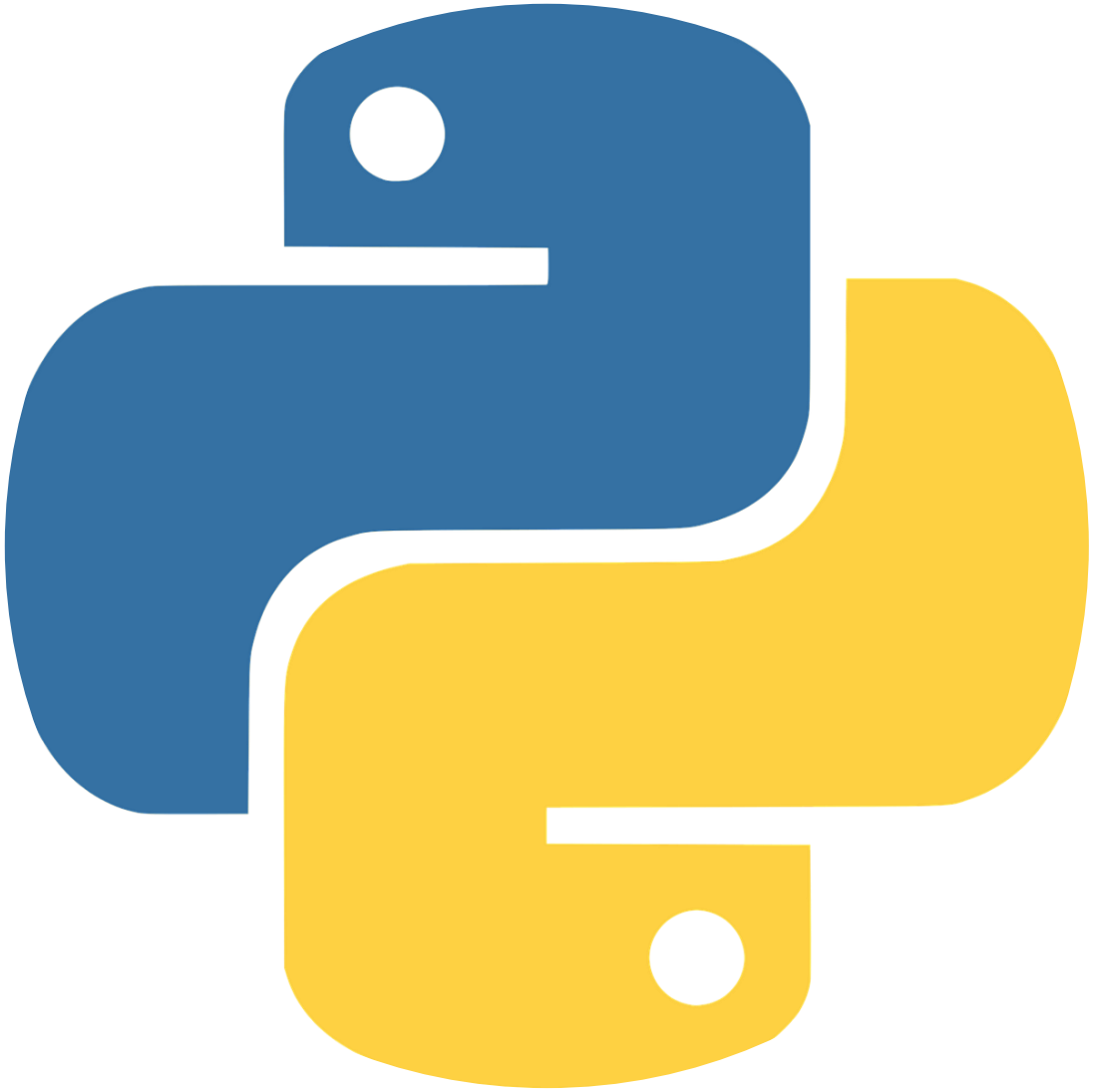
Third loop

Loop variable = 3

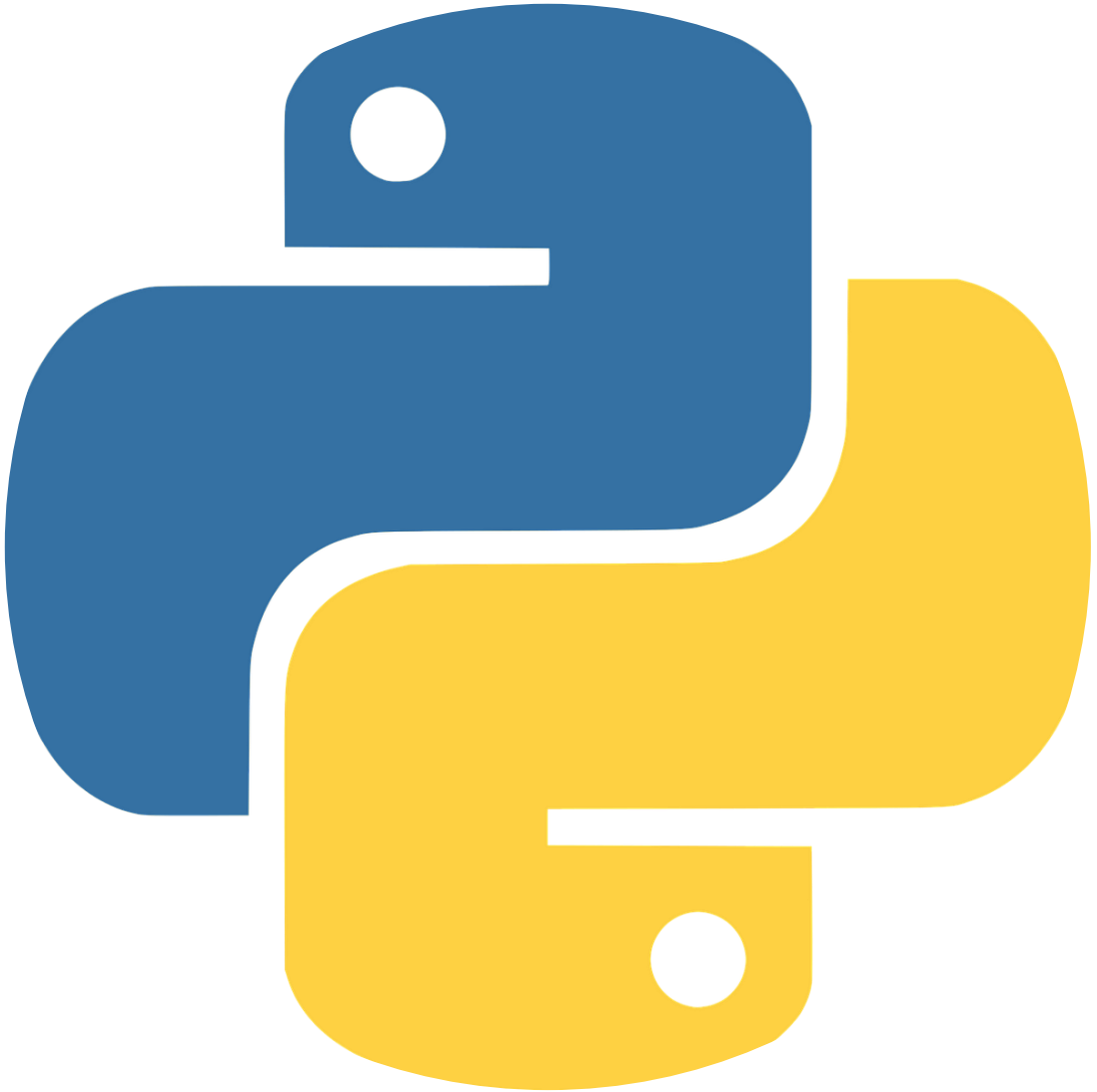
EXPERT TIPS

Range

In Python code, the word "range" followed by two numbers within brackets stands for "all the numbers from the first number to one less than the second number". So `range(1, 4)` means the numbers 1, 2, and 3—but not 4. In Emma's "Keep Out" program, `range(1, 11)` is the numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10.



While Loops



While Loops

A while loop doesn’t have a loop variable that’s set to a range of values. Instead, it has a loop condition. This is a Boolean expression that can be either True or False.

It’s a bit like a bouncer at a disco asking you if you’ve got a ticket.

If you have one (True), head straight for the dance floor; if you don’t (False), the bouncer won’t let you in.

In programming, if the loop condition isn’t True, you won’t get into the loop!

Code this bouncer at disco situation

While loop

▽ **Escaping infinity**

You can deliberately use an infinite loop to get input from the user. This (annoying) program asks if the user is bored. As long as they type "n", it keeps asking the question. If they get fed up and type "y", it tells them they're rude and uses the **`break`** command to leave the loop!

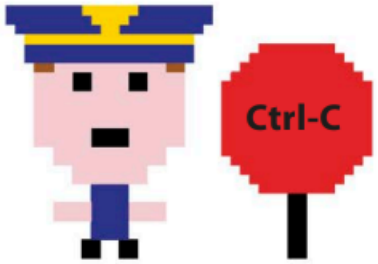
The True condition is that the user is not bored yet (`'n'`).

```
>>> while True:
        answer = input('Are you bored yet? (y/n)')
        if answer == 'y':
            print('How rude!')
            break
```

The False condition (`'y'`) triggers the **`break`** command.

Stopping the loop

If you don't want an infinite loop, it's important to make sure that the body of a **`while`** loop does something that could make the loop condition False. But don't worry—if you accidentally code an infinite loop, you can escape from it by pressing the C key while holding down the Ctrl (control) key. You may have to press Ctrl-C several times before you quit the loop.



While loop

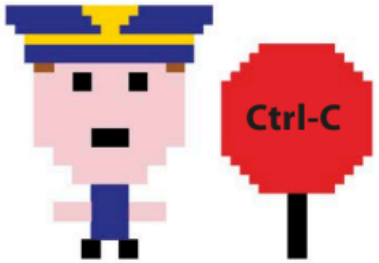
Loops inside loops

Can the body of a loop have another loop within it? Yes! This is called a nested loop. It's like Russian dolls, where each doll fits inside a larger doll. In a nested loop, an inner loop runs inside an outer loop.

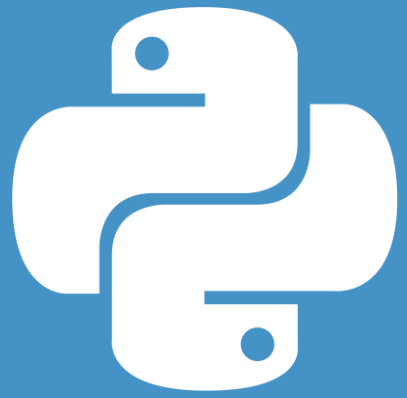
Cheerleader's chant:
1
2
3
Hey
1
2
3
Hey
1
2
3
Hey

Stopping the loop

If you don't want an infinite loop, it's important to make sure that the body of a `while` loop does something that could make the loop condition False. But don't worry—if you accidentally code an infinite loop, you can escape from it by pressing the C key while holding down the Ctrl (control) key. You may have to press Ctrl-C several times before you quit the loop.



Time to code



Guess the number

Make a program that generates a random number between 1 and 100.
Then each time the program asks the user to input a guess number.

If the guess is correct, the user wins. If the guess is lower than the correct number, the program prints a message ”go higher” and if the guess is higher than the correct number the program prints the message “go lower”.

If the user passes the number of tries the game stops, the user looses.