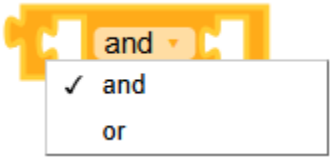
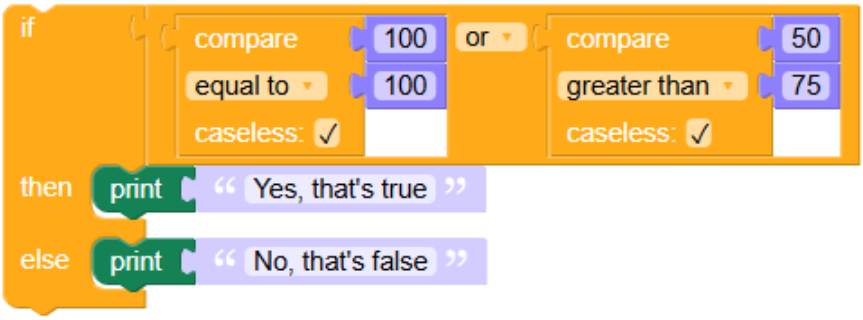
Flow

Use these blocks to control the direction the program goes. ***If/Then*** blocks are covered in the Intermediate Mode tutorial, as are ***compare***, ***repeat***, ***forever***, ***wait***, and ***end program***.

Here are the new blocks introduced in the Advanced Mode.



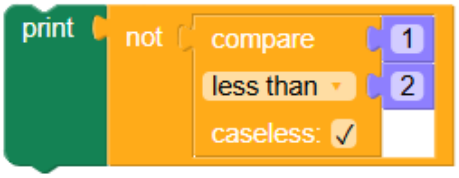
This block, known as a *logical operator*, reports either True or False depending on the information you give it. You can compare two pieces of information and connect them with either ***and*** or ***or***. Here is an example.



If you use ***or***, you'll see "Yes, that's true" because only one of the two comparisons needs to be True for the block to report True.

If you change the ***or*** to ***and***, you will see "No, that's false", because even though 100=100, 50 is not > 75. Both cases would have to be True for the combined result to be True.

This ***not*** block is another *logical operator*. It reports the opposite of the True/False information you give it. For example, in this case, it reports False because the comparison itself is True and the ***not*** that comes before the comparison reports the opposite of that comparison.

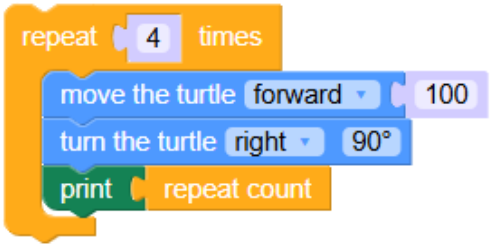


If you change ***less than*** to ***greater than***, it will report True, the opposite of the result of the comparison.

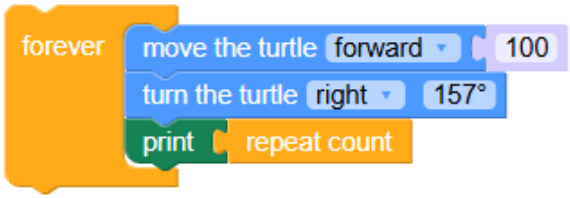
If your head is beginning to hurt as you try to understand these logical blocks, don't worry! You may not need to use them often, but they can come in handy when you do!

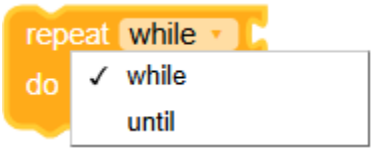
The ***repeat count*** block lets you keep track of the number of times the code is run through the loop. It counts the number of times the code runs a ***repeat*** loop or a ***forever*** command.

In this example, you will see the numbers 1 through 4 display in order, one for each time the repeat loop is run.



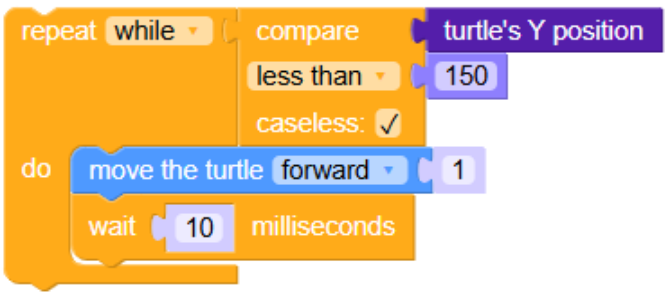
In the next example, a ***forever*** block is used.
Using ***repeat count*** with forever may help you figure out how many times you want to repeat a set of blocks.

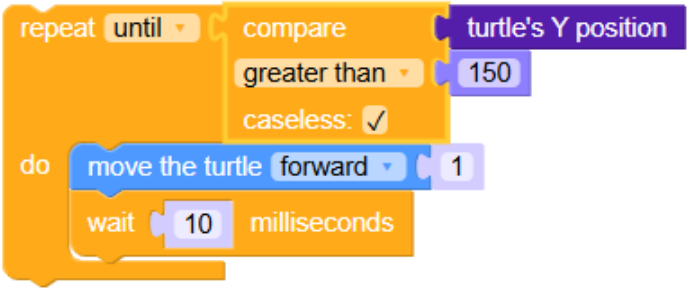


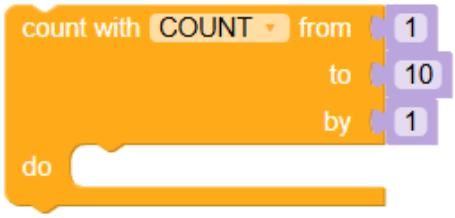


This block can be set to ***while*** or ***until***. You can use it to repeat a set of commands until or while a condition is met.

For example, can you see how both these sets of code produce the same result? They are different in two spots. Look at the code carefully to find them.

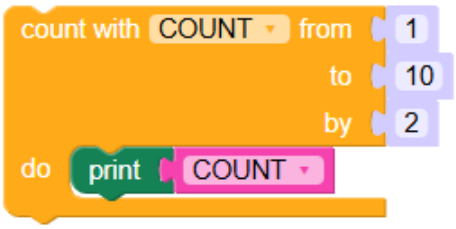




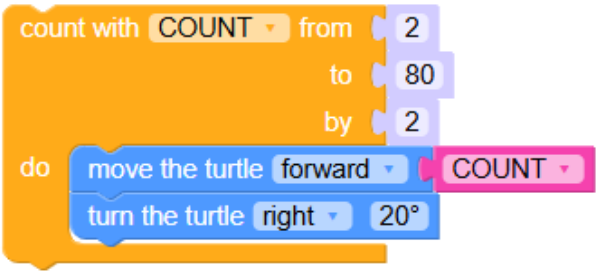


The ***count*** block uses a variable named COUNT to keep track of numbers. You can tell it to count by 1s, 2s, 3s, 10s, 100s, or any number. Tell it what number to start at and when to end.

Here is a simple example to start with.
It simply counts from 1 to 10.
The numbers appear one at a time in an alert box. If you change to a very large ***to*** number and don't want to finish the sequence, you can click the red **Stop** button in the alert box. Otherwise click **OK** to continue counting.

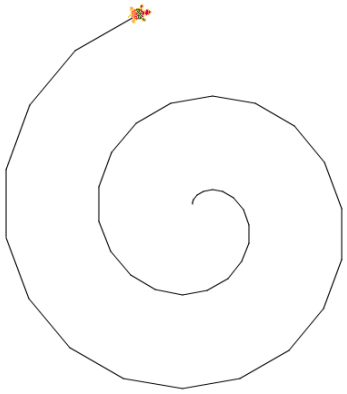


To try this example, you'll find the ***print*** block in the Output section. The **COUNT** variable is in the Variables section.



Next is a more complicated use of this block.

Can you imagine what the code will draw? Each step of the way, the turtle moves forward the value of COUNT, which increases by 2 every time. This is a type of loop. It uses Logo's FOR command.



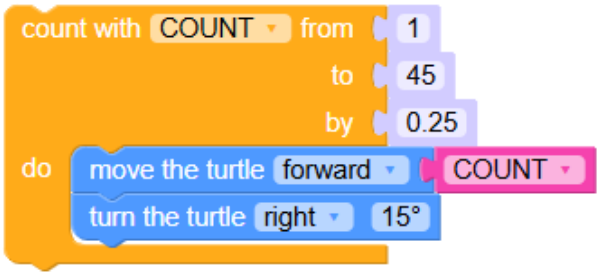
Editor

```
1  FOR [COUNT 2 80 2] [
2    FORWARD :COUNT
3    RIGHT 20
4  ]
```

Experiment with the numbers and create your own version!

Can you figure out how to create this variation?

Notice that you can use a number less than 1 for the ***by*** value. This code uses .25. That means that the forward distance increases by a quarter of a step each time: 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, etc., all the way to 45.



Editor

```
1  FOR [COUNT 1 45 0.25] [
2    FORWARD :COUNT
3    RIGHT 15
4  ]
```