While Loops

While Loop Syntax

`while` loops, like `for` loops, use curly braces `{}` and indents for all commands that should be repeated. However, `for` loops generally contain 3 elements (an initialized variable, a boolean expression involving that variable, and a change in the value of that variable) while a `while` loop usually contains just a boolean expression.
The `for` and `while` loops below produce the same results.

```
for (int i = 0; i < 5; i++) {
  System.out.println("Loop#: "+i);
}
```

```
int i = 0;
while (i < 5) {
  System.out.println("Loop# "+i);
  i++;
}
```

Note that the variable declaration and initialization happen *before* the start of the `while` loop and any changes that occur to the variable happen *within* the body of the curly braces `{}`. On the other hand, everything happens in one step within parentheses `()` when using a `for` loop.

Here is another example of a `while` loop that prints `Hello` based on the value of the variable `count`.

```
int count = 5; // some random number set by user
while (count > 0) {
  System.out.println("Hello");
  count--;
}
```

Code Visualizer

TRY IT

What happens if you:

- Change the while statement to `while (count > -1 * count)`?
- Replace `count--` in the code above with `count = count - 2`?
- Change the while statement to `while (count < 10)`?

TRY IT

▶ **How does `while (count > -1 * count)` work?**

Infinite Loops

Infinite loops are loops that do not have a test condition that causes them to stop. The following is a common mistake that results in an infinite loop:

```
int count = 5; // some random number set by user
while (count > 0) {
  System.out.println("Hello");
}
```

Since the variable `count` never gets decremented. It remains at 5, and 5 will forever be greater than 0, so the loop will never stop.

Copy the code above and `TRY IT` to see what happens. Java will eventually stop the loop due to an output limit, but it may take some time before this happens.

TRY IT

Why Use a While Loop?

If a `while` loop does the same thing as a `for` loop, then what is the purpose of having both? `while` loops are actually more useful when you are waiting for a certain event to occur. Imagine you are making a video game. The game should continue until the player loses all of their lives. You don't know how long this will take, so a `while` loop is more appropriate. On the other hand, if you have more specific loop parameters, a `for` loop will be better.

```
int player_lives = 3;

while (player_lives > 0) {
  // video game code
  // goes here
}
```

While vs. For Loops

Fill in the blanks below with either `while` or `for`.

A while loop usually contains a boolean expression(s) in its header and nothing else.
A for loop contains a header that specifies where an iterator variable starts, where it ends, and how it is changed per iteration.
A ~~while~~for loop is better if a command needs to be executed a certain number of times.
A counting variable needs to be declared and initialized before a while loop can be executed properly.

One big difference between a `while` loop and a `for` loop is that a `while` loop only contains a boolean expression(s) within its header. That means a counting variable must be declared and initialized before a `while` loop can be used properly. On the other hand, the header of a `for` loop contains an iterator variable which helps to determine how many times the `for` loop will run.