Loop de Loop — do & while

Previously we talked about the ability for the processor to execute a branch instruction so that the next instruction it executes isn't just the next one in the ordered list of instructions. We used that be able to jump back in the list to some that had already been executed, creating a loop (though you can, of course, jump forward as well.)
And we saw how the Arduino environment provides us with a basic, so called infinite, loop structure. While that is very handy, sometimes you may want to loop through a particular set of instructions — not forever, but for a specific number of times.

while() {} Loop

Statement Block	Conditional
Runs every loop, but **AFTER** the first check of the conditional.	Is checked **BEFORE** each execution of the block. Once it is **FALSE**, block will not run again.

```
while ( i < 10) {
    Serial.println(i++);
}
```

C/C++/Arduino actually provides a variety of mechanisms to facilitate controlled looping. Lets start with one of the most basic, the **`while`** loop.

The **`while`** construct just tells the cpu to continue to do the same set of instructions as long as a particular condition is met. Many languages use the same term and general organization for such types of loops, though the specifics of the syntax may vary some.

The syntax is really pretty simple, as illustrated in the above graphic. The keyword `while` is followed by a conditional statement enclosed in parenthesis. It is then followed by a statement block, enclosed, as usual, within braces.

As long as the conditional statement evaluates as True, the statements in the statement block be executed over an over again (in order, of course), but as soon as the conditional evaluates to False, then statements in this statement block will be skipped and program flow will continue with the statements after those defined by the `while`.

A variant of the `while` loop is the `do/while` loop. It's syntax is mostly similar to the plain while loop, as described in the bottom-most graphic on the page. After the keyword **do** there's a statement block, and after the closing brace is a comparison operation inside parenthesis. Note that all statements inside the block must be terminated with a semicolon as usual ***and*** the entire do/while statement must also have a terminating semicolon after the closing parenthesis.

Both of these can be nested. That is, more `do/while` or `while` constructs may be contained inside the statement block of a `do/while` or `while`. This will be true of most all of the constructs we explore in this class.

The main thing that sets the `do/while` loop apart from other loop control mechanisms is that the statement block *will always be executed at least one time*. This is because the comparison will not be performed and checked until after the statement block has executed.

That may seem insignificant, but as you start breaking down your desired tasks into subcomponents and building your algorithms and programs, you will find that to be an important difference. Since the regular `while` checks the conditional ***before*** the statement block is first executed, it is thus possible that the block may *never* be executed, if the conditional evaluated to false at the first test.

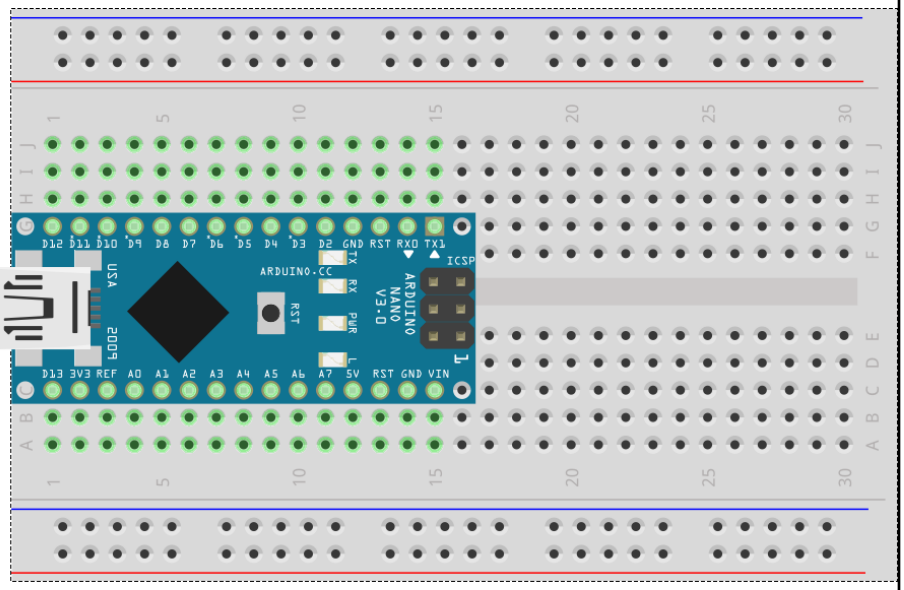
There are times where one of these may be more appropriate than the other, though most problems can be solved using either, provided the appropriate supporting logic exists.

do {} while() Loop

Statement Block	Conditional
Runs every loop, and **BEFORE** the first check of the conditional.	Is checked **AFTER** each execution of the block. Once it is **FALSE**, block will not run again

```
do {
    Serial.println(i);
    i = i + 2;
} while ( i < 50 );
```

Experiment: What to do? While away the time.

Components	Wiring Diagram
✓Computer ✓Arduino IDE	

Connection Instructions
No connections required, we will be using the internal LED for this example.

Sketch(es)	whileInRome.ino

Analysis Questions
Could you also do some or all of this task using for() loops? Given the default 'loop()' in the Arduino environment, could you do this without a while loop at all?

Programming Tasks
Starting with the stub provided, create a program which outputs the roman numeral five (V) via the serial link every 5 seconds. Note that there are a few ways to accomplish this task. Optionally also print out the roman numeral ten (X) every 10 seconds as well. There are even more ways to accomplish this additional task.