Repetition

One of the greatest strengths of computers is their ability (and willingness) to do the same task over and over again.

In addition, the computer will always perform the task in exactly the same way for the same input, provided the programmer (you!) has done their job correctly.

Repetition – Example

Suppose we wanted to output the numbers from 1 to 10, in order, on the screen. How could we do this? How does efficiency affect how we would do it?

What if we wanted to count to 100? 1000? Would that change our strategy?

Counted Loops

The simplest form of repetition is called a loop, and the simplest type of loop is the counted loop.

In a counted loop, we execute some segment of code a fixed number of times.

In English, we might say:

Do *something* 10 times

Are We Done Yet?

If we are going to perform a task a fixed number of times, we need to keep a count so we know when we're done.

Since counting involves whole numbers, we use an integer variable as our counter. Try to pick a meaningful name for the counter.

The For Loop

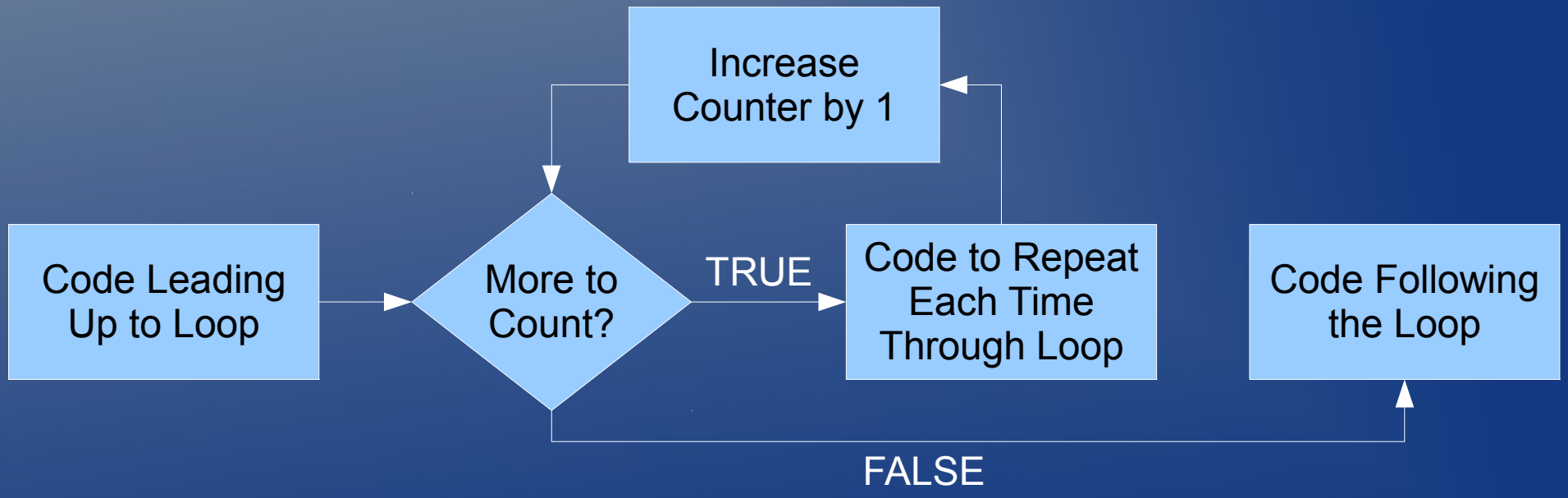
The counted loop is so common that a special type of loop was created to streamline the code.

```
for count : 1 .. 10
  put count
end for
```

Notes:
1. Do NOT declare the variable **count**. The **for** loop will take care of that automatically

Counted Loop – Flowchart

for count from A to B
    statements to execute for each loop
end for



Loop ends when counting is complete

Counting Down with a For Loop

When we count upwards, the counter is incremented (i.e., we add 1 each time). It is also possible to count downwards by decrementing the counter. In Turing:

```
for decreasing count : 10 .. 1
  put count
end for
```

Example – Blast Off!

```
% before the loop
put "Begin Count Down..."

for decreasing count : 10 .. 0
  put count
  % notice that we can use the
  % count variable inside the loop
  delay(1000)
end for

% after the loop
put "Blast Off!"
```

Control Counting with Variables

The upper and lower bounds for the counter can also be variables.

```
var low, high : int
put "Count from? " ..
get low
put "Count to? " ..
get high

for count : low .. high
  put count
end for
```

Changing the Increment/Decrement

So far, we have incremented or decremented by one. It is possible to take larger steps using the "by" command to specify the (integer) step size.

```
for count : 1 .. 10 by 2
  put count
end for

for decreasing count : 10 .. 1 by 3
  put count
end for
```