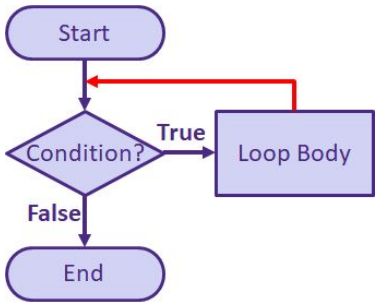
Section 8: Loops

Introduction: A loop allows us to execute the same block of code multiple times until a specified conditional expression becomes `false` (i.e. "do `<something>` until `<condition>` fails"). Similar to multiple function calls, loops tend to be most useful when they are used to execute *similar* (not *identical*) sets of instructions. You may find it helpful to think of a loop as a *condensed* form of repeated, similar code.

---

while-loops: This type of loop repeatedly runs the code inside of it while a conditional expression is `true`:

```
while( condition ) {
    body;  // while-loop body
} // jump back to top of while loop
```



Notice how the code inside of the loop is contained within curly braces, just like the code in a function! In general, curly braces denote a "block" of code.

Example:
```
int x = 1;
while( x < 10 ) {
    x = x * 2;
}
```

The above loop will execute the statement `x = x * 2` four times, with the final value of `x = 16`:

Iteration	x	Condition (`x < 10`)	Result
1	1	`true`	Execute `x = 1 * 2;`
2	2	`true`	Execute `x = 2 * 2;`
3	4	`true`	Execute `x = 4 * 2;`
4	8	`true`	Execute `x = 8 * 2;`
5	16	`false`	Exit loop

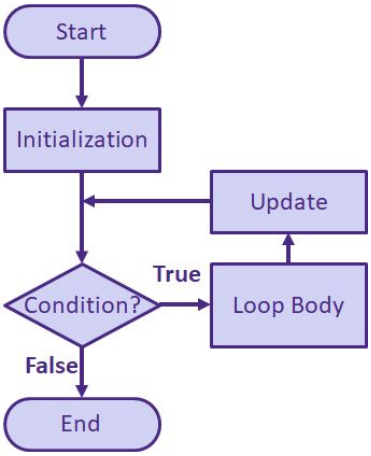
---

for-loops: These are very similar to while-loops, but they allow you to specify additional initialization and update statements, which are separated by semicolons (`;`). The following side-by-side code segments are equivalent:

```
for(init; condition; update)
{
    body; // for-loop body
}
```

⇔

```
init;
while( condition ) {
    body;
    update;
}
```



Examples:
```
int x;
for( x = 1; x < 10; x = x * 2 ) {}
```

This (empty body!) for-loop is equivalent to the while-loop example: we're able to eliminate the entire body of the loop because it's now executed as the update statement!

Note that if you *declare* a variable in the `init` statement, then that variable is *local* to the body of the for-loop, similar to parameters being local to a function body.

```
int sum = 0;
for( int i = 1; i <= 6; i = i + 1 ) {
    sum = sum + i;
}
```

The above loop will sum the numbers from 1 to 6, with the final value of `sum = 21`:

Iteration	i	sum	Condition (i <= 6)	Result
1	1	0	true	Execute sum = 0 + 1;
2	2	1	true	Execute sum = 1 + 2;
3	3	3	true	Execute sum = 3 + 3;
4	4	6	true	Execute sum = 6 + 4;
5	5	10	true	Execute sum = 10 + 5;
6	6	15	true	Execute sum = 15 + 6;
7	7	21	false	Exit loop

Exercises:

1) Describe what the while-loop below does. Then rewrite the code segment using a for-loop.

```
int pos = 0;
while( pos < min(width,height) ) {
    rect(pos, pos, 50, 50);
    pos = pos + 50;
}
```

2) Complete the while-loop below to find the **smallest power of 3 greater than 100**. Your answer should be stored in the variable `answer` *after* the loop has executed:

```
int answer = _____;

while( ____________________ ) {

    answer = ____________________;

}
```

3) Complete the for-loop below that calculates the **sum of all *even* integers from 50 to 100, inclusive**. Your answer should be stored in the variable `sum` *after* the loop has executed:

```
int sum = _____;

for( int i = _____; i <= _____; i = i + _____ ) {

    sum = ____________________;

}
```

4) Find a partner, brainstorm Creativity Project ideas, and get started on “Creativity Planning.” [*partners*]