`while` Loops
Looping Control Statement

Introducing: **`while`** Loops

- General form of a **`while`** loop statement:

```
while (<boolean expression "test">) {
    <repeat block – statements in braces run when test is true>
}
```

- ***Like*** an **`if-then`** statement:
  - the test you place in the parenthesis must be a `boolean` expression
  - if the test evaluates to **`true`**, the computer will move to the first line of code in the repeat block
  - If the test evaluates to **`false`**, the computer will *jump* over the repeat block

- ***Important! Unlike*** an if-then, **after the last statement in the repeat block** completes, the computer will next ***jump backwards* up to the test** and start afresh.

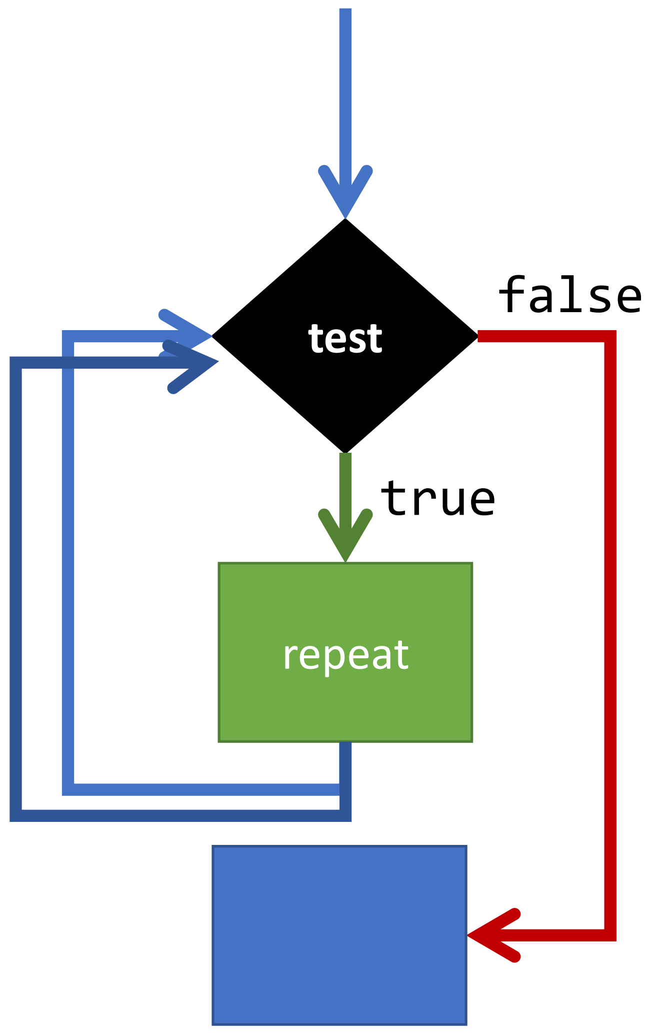
- A **`while`** loop statement can be used *anywhere* you can write a statement.

while loop Flow of Control

1. When a **`while`** statement is encountered, its **`boolean`** **test** expression is evaluated

2. If the **test** is **`true`**,
   a) then the processor will **proceed into the repeat block**.
   b) At the end of the repeat block, the processor jumps back to **step 1**.

3. If the **test** is **`false`**, the processor will jump over the repeat block and continue on.



Example Setup

In VSCode:

1. Start the Development Server
   - View Terminal
   - `npm run pull`
   - `npm start`

2. Open the File Explorer Pane
   - Right click on the src folder
     - Select "New folder"
     - Name it: **`x-while`**
   - Right click on the x-while folder
     - Select "New file"
     - Name it: **`while-app.ts`**

3. In `while-app.ts`, write out the code to the right. It has no errors, so review carefully if yours has any.

```
import { print, promptNumber } from "introcs";

export let main = async () => {

    let n = await promptNumber("How many times?");
    let i = 0;
    while (i < n) {
        print("Loop: " + i);
        i = i + 1;
    }

    print("Done!");

};

main();
```

Writing a **`while`** loop that repeats a specific number of times.

- Repeating a task a specific number of times is a **very** common task in computing.

- You will see this all semester.

- Three keys:
  1) Declare a counter variable and initialize it to 0.
  2) The loops test will check that the counter variable is less than the # of times you want to repeat
  3) Don't forget! The last step inside of the repeat block is incrementing your counter variable.

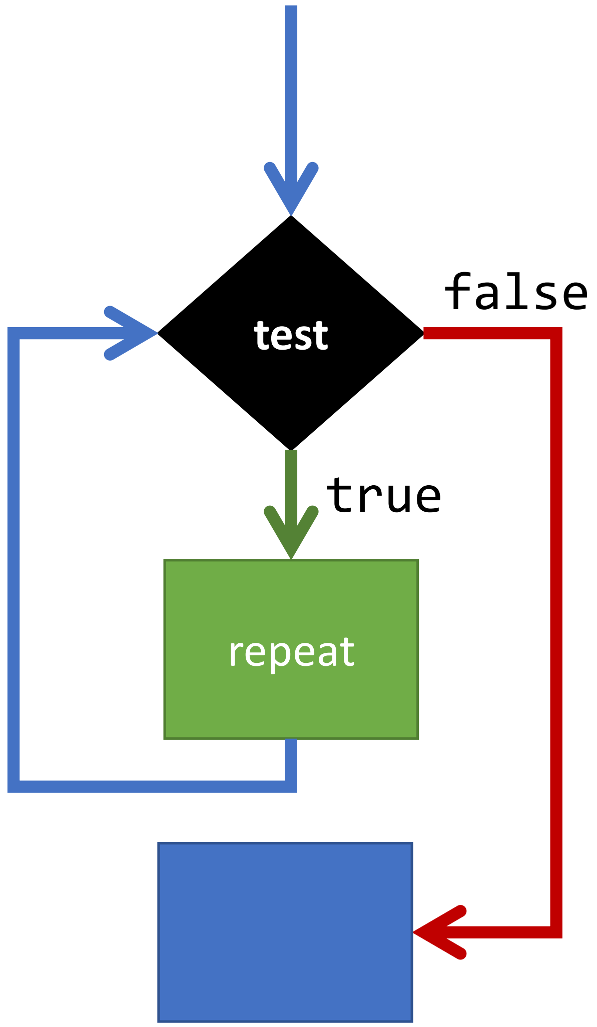
```
1
let i = 0;

while (i < __2__) {

   // Do Something Useful

   i = i + 1; 3

}
```

while loop Statement Notes

- If the test is ***not true*** the first time the while loop is encountered,
  then the computer will jump past the repeat block.

- If the test ***never evaluates to false***,
  then the loop is called an ***infinite loop***.

- The only way to *stop* an *infinite* loop is to force quit/close your browser.



How do you **avoid infinite loops**?

Your **test** condition must eventually evaluate to **`false`**, therefore

a value in the test must be changing inside the repeat block, such that

progress is made toward the test expression evaluating to `false`.

```
let i = 0;
while (i < n) {
    print("Loop!");
}
```

Bad! Nothing is changing inside of the repeat block.

```
let i = 0;
while (i < n) {
    print("Loop!");
    i = i - 1;
}
```

Bad! Subtracting 1 from `i` is not making progress toward `i >= n`.

```
let i = 0;
while (i < n) {
    print("Loop!");
    i = i + 1;
}
```

Good! Adding 1 to `i` *is* making progress toward `i >= n`.