The "single-equals" ("=") is the "assignment operator". It takes whatever value is on its right-hand side, and puts it into the variable ("bucket in memory that can hold a value") on its left-hand side.

The "double-equals" or "equal-equals" ("==") is the "equality comparison operator". It takes whatever value is on its left-hand side and whatever value is on its right-hand side, compares them, and returns the Boolean (true/false) value "true" if they are equal, and "false" if they are not.

Visual Studio will warn you if you use single-equals in an if-statement. Unfortunately, it won't warn you very usefully.

For the code:

int integerValue;

if (integerValue = 3) {

// do something

}

Console.WriteLine(integerValue);

Visual Studio offers the very unhelpful error: "Cannot implicitly convert type 'int' to 'bool'".

What's actually going on here:

1. The expression "integerValue = 3" copies the value "3" into the variable "integerValue".
2. The expression "integerValue = 3" also **has** a value, and that value is "3" (the same value that it copied into the variable).
3. The "integerValue" variable is an "integer" (whole number) variable. It can only hold integers, and can only be used where an integer value is appropriate.
4. The if-statement requires a "Boolean" (true/false) value, not an "integer" (whole number) value, in its conditional statement (the "if (conditional-statement)..." part).
5. Trying to give the if-statement an integer value, when it expects a Boolean value, is an error (it is **always wrong**).
6. Using "=" inside a conditional statement is a warning (it is **sometimes** **wrong**).
7. When a line of code has both warning(s) and error(s), Visual Studio only displays the error(s). The warning(s) are not shown.
8. In this case, the warning ("Assignment in conditional expression; did you mean to use == instead of =?") would have been **much more useful** than the error ("Cannot implicitly convert type 'int' to 'bool'").

So, in this case, by trying to be "too helpful" (by only showing you the Very Important Error[tm] rather than the Possibly Useful Warning[tm]), Visual Studio has suppressed the very information you would have needed to make sense of the problem. Thank you, excessively helpful software!

Side note: the root of this problem is that using a "=" inside a conditional statement is permitted by the language for historical reasons, even though it's a terrible, terrible idea. The correct solution is not to fix Visual Studio's handling of warnings and errors, but to redefine "using '=' inside a conditional statement" as an error. But that would require a change to the C# language definition that would break existing, working code -- and that's not likely to happen, now or ever, regardless of how much it would improve things.