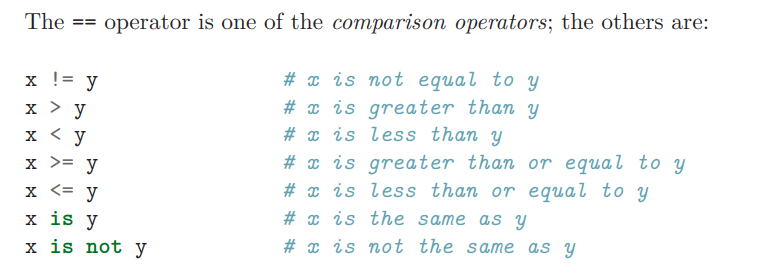
9/3 CSP P4E C3.1-3.2,3.8

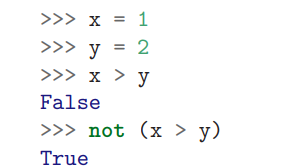
### **Overview**

**Chapter 3: Boolean expressions**

### **3.1 Boolean expressions**

* **Definition of Boolean expressions:** A *boolean expression* is an expression that is either true or false
* True and False are special values that belong to the class bool; they are not strings
* **Example:**
  + The following examples use the operator ==, which compares two operands and produces True if they are equal and False otherwise:
  + 5 == 5 True
  + 5 == 6 False
* 
* **Common Errors**: use a single equal sign (=) instead of a double equal sign (==). Remember that = is an assignment operator and == is a comparison operator. There is no such thing as =< or =>.

### **3.2: Logical operators**

* **3 logical operators: *and, or, and not.***
* **Example:**
  + ‘x > 0 and x < 10’ is true only if x is greater than 0 and less than 10.
  + ‘n%2 == 0 ***or*** n%3 == 0’ is true i***f either of the conditions is true***, that is, if the number is divisible by 2 or 3.
  + Finally, the ***not*** operator negates a boolean expression, so not (x > y) is true if x > y is false
  + 
* Strictly speaking, the operands of the logical operators should be boolean expressions, but Python is not very strict. Any nonzero number is interpreted as “true.” ‘17 and True’ True

### **3.8: Short-circuit evaluation of logical expressions**

* When Python is processing a logical expression such as x >= 2 and (x/y) > 2, it evaluates the expression from left to right. Because of the definition of and, if x is less than 2, the expression x >= 2 is False and so the whole expression is False
* Short-circuiting: When the evaluation of a logical expression stops because the overall value is already known, it is called short-circuiting the evaluation.
* guardian pattern: We can construct the logical expression to strategically place a guard evaluation just before the evaluation that might cause an error as follows:
* 