ECMM455 Python Worksheet 6: Logical expressions

Prof Hywel Williams

September 12, 2019

1 Aims

- Practice using logical expressions

2 Logical expressions

A logical expression has an unambiguous truth value: *true* or *false*. There are two main kinds:

- Relations (use relational operators)
- Boolean expressions (use logical operators)

In Python, the two truth values are *True* and *False* . When Python evaluates a logical expression it swaps it for the resulting truth value in all subsequent calculations. Boolean variables are used to store truth values e.g. *x=True* .

2.1 Relational operators

Relational operators assign a truth value based on numerical comparisons. The six relational operators in Python are given below:

Python	Relation	Return value
a<b	Less than	**True** when **a** is less than **b**, **False** otherwise
a>b	Greater than	**True** if **a** is greater than **b**, **False** otherwise
a<=b	Less than or equal to	**True** if **a** is less than or equal to **b**, **False** otherwise
a>=b	Greater than or equal to	**True** if **a** is greater than or equal to **b**, **False** otherwise
a==b	Equal	**True** if **a** is equal to **b**, **False** otherwise
a!=b	Not equal	**True** if **a** is not equal to **b**, **False** otherwise

2.2 Boolean operators

Boolean operators (after the English mathematician who developed their use, George Boole 1815-1864) are used to combine logical statements into more complex statements which also have an unambiguous truth

value. The most common are: *not, and, or* . Truth tables for how logical expressions including Boolean operators are evaluated are given below:

A	B	not A	A and B	A or B
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

There are some precedence rules for the order of evaluation - in short, anything in parentheses is evaluated first. The rules for Python are:

Operator	Description		
( )	Sub-expressions in parentheses	High	First
<, <=, >, >=, ==, !=	Relational operations	Precedence	Evaluated
not	Logical NOT		
and	Logical AND		
or	Logical OR	Low	Last

It is good practice to use brackets to make the meaning clear when writing logic statements.
Note that all kinds of object have truth values in Python:

- Zero values, empty strings, empty lists → False
- Everything else → True

(But you will rarely need to use this!)

3 Exercises

For these logical expressions, first work out the truth value using pen-and-paper. Then check your answer by entering them into Python in interactive mode.

1. "test" != "testing"
2. "test" == 1
3. not (True and False)
4. not (1 == 1 and 0 != 1)
5. not (10 == 1 or 1000 == 1000)
6. not (1 != 10 or 3 == 4)
7. not ("testing" == "testing" and "Zed" =="Cool Guy")
8. 1 == 1 and (not ("testing" == 1 or 1 ==0))
9. "chunky" == "bacon" and (not (3 == 4 or 3== 3))
10. 3 == 3 and (not ("testing" == "testing" or "Python" == "Fun"))

11. 1 == True

12. 0 == True

13. “x” == True

14. “” == True