Chapter 6 — Boolean

Introduction

The mathematician George Boole studied the algebra of statements that contain only true and false. In Boolean algebra there are three basic operators; 1) addition known as or, 2) multiplication known as and, and 3) the compliment known as not. This chapter will show how to write and understand Boolean expressions in Python.

Objectives

Upon completion of this chapter's exercises, you should be able to:

- Explain the meaning of the Boolean values and the ways that they may be represented in a Python program.
- Compute a Boolean expression using the operations of AND, OR, and NOT.
- Produce a truth table given a Boolean expression with variables.
- Create Boolean expressions in a Python program.
- Use the comparison operators to compare two values.
- Create complex comparisons between several values using comparison operators and Boolean operators in a Python program.

Prerequisites

This chapter required a firm understanding of literal values and variables from Chapter 1. The section on Truth Tables will use For loops from Chapter 5 and Lists from Chapter 4.

Boolean Values

George Boole (1815-1864) was the first to describe the process of logic (true/false) in terms of algebra. This formalization was described in his works *The Mathematical Analysis of Logic* (1847) and T*he Laws of Thought* (1854). (Burris, 2014) It is for his early work and years of research that we use the name Boolean to describe the concept and algebra of true/false values.

Boolean mathematics uses only one (1) representing true and zero (0) representing false. It also has only three operations: 1) addition (also known as OR); 2) multiplication (also known as AND); and 3) compliment (also known as NOT). In Boolean Algebra there is no positional value or place notation

(Positional notation, N.D.), like in other numbering systems (decimal, binary, …). The result of a Boolean operation is always a single Boolean value.

The Three Boolean Operations

Each of the operations are described below using a truth table. In a truth table we place the variables on the left hand side and then show intermediate and the final expression on the right hand side.

Boolean addition: OR +

Boolean Addition is often called the OR operation and is represented in an expression by a plus sign (+). The operation can be stated simply as "zero plus zero is zero, anything else is true". It can also be said that "if one or both are true then the result is true, otherwise the result is false".

The meaning of OR in Boolean Algebra is not typically the meaning of "or" in the English language. We are often told we should do "this or that" but that does not include both being true. In the context of Boolean Algebra we can do both.

X	Y	X + Y
0	0	0
0	1	1
1	0	1
1	1	1

Illustration 9: Truth Table - Addition

Boolean Multiplication: AND *

In Boolean Multiplication "anything multiplied by zero is zero, one multiplied by one or one". The multiplication sign may be omitted between terms and it is assumed.

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

Illustration 10: Truth Table - Multiplication

Compliment: NOT '

The Boolean compliment is just the opposite. True becomes false and false becomes true. Some authors will place a bar above to denote NOT while some will place an apostrophe after.

X	X'
0	1
1	0

Illustration 11: Truth Table - Compliment

Order of Operations

The order of operations, in Boolean Mathematics, is the same as normal algebra if you think of the compliment like we think of negation.

- Parenthesis — ( )
- Compliment — NOT
- Multiplication — AND
- Addition — OR

The Five Postulates

The five postulates of Closure, Identity, Commutativity, Distributivity, and Compliment are the base rules of Boolean algebra that the other rules are based upon. The Laws of Closure, Identity, Commutativity and Distributivity are the same, with respect to multiplication and addition, as in your previous experiences with algebra. The Compliment postulate is defined by the basic operations of OR and AND over the set of Boolean numbers (0 and 1).

Postulate	**Definition**	
1: Closure	IF X and Y are elements then X+Y and XY are also elements	
2: Identity	X + 0 = X	X * 1 = X
3: Commutative Law	X + Y = Y + X	XY = YX
4: Distributive Law	X(Y + Z) = XY + XZ	X + YZ = (X + Y)(X + Z)

5: Compliment Law	X + X' = 1	X * X' = 0

Table 5: The Five Postulates of Boolean Algebra

Please see Bonus Chapter 3 for a more complete discussion of simplification of Boolean expressions.

True and False Values in Python

In Python, we will use Boolean values to control whether to execute a block of code or to skip it. The language includes two keywords for Boolean values: `True`, and `False` (notice that they have their first letter capitalized). They may be used in an expression just like any other constant.

`True False`	Constants
The constants `True` and `False` may be used in your Python programs to represent the two Boolean values. https://docs.python.org/3/library/constants.html#built-in-constants http://bit.ly/2rYf99C	

In an actual conditional statement, there is a list of values that will be interpreted as false, they are:

- `None`;
- `False`;
- `0` — zero (integer, float, complex…);
- `''` or `""` — a string of zero length;
- `[]`, `()`, `{}` — a sequence or map that is empty;
- and special user defined classes.

Other values that are not on the list, above, will be interpreted as true.[1]

1 https://docs.python.org/2/library/stdtypes.html

`None False 0 0.0 '' "" () {} []`	False Values
In Python the values above are considered false in Boolean expressions. Any non-zero number, string with a length 1 or longer, or sequence containing 1 or more values are considered true. User defined classes may be considered true or false based upon their current state or value. https://docs.python.org/3/library/stdtypes.html#truth-value-testing http://bit.ly/2SMtSAk	

Boolean Operations in Python

Python includes the three Boolean operators:

- `and` — multiplication — x and y are true if both are true;
- `or` — addition — x or y is true if either or both are true;
- `not` — compliment — not x is true when x is false.

These operators may be used to create complex comparisons and you mat use parentheses to group operations. Like in normal mathematics, the compliment (or negation) is done first, then operations inside parenthesis are next, then multiplication, and lastly addition.

Order	Operator
First	not
Second	()
Third	and
Fourth	or

Table 6: Order of Boolean Operations

`and or not`	Boolean Operators
The symbols `and`, `or` and `not` are used as the Boolean operators. Values evaluated by the operators do not need to be strictly Boolean. `X or Y` — if X is false then Y else X `X and Y` — if X is false then X else Y `not X` — if X is false then True else False https://docs.python.org/3/reference/expressions.html#boolean-operations http://bit.ly/2Re0MwF	

Below are a couple of complex expressions in Python. Pay special attention to the order of operations when you look at the results.

```
a = True
b = False
c = False
d = a and b or c
print(d)
d = b and c or a
print(d)
d = b and (c or a)
print(d)
d = not b and (c or a)
print(d)
```

```
False
True
False
True
```

Truth Tables

In previous topics in this chapter a truth table has been shown but the details of what they are and how they can be used was not fully explained. A truth table is a simple table that will allow you to go through all of the true/false values of the variables and show the outcomes of a Boolean expression for all of the possibilities.

There will be 2^n lines in a truth table, where "n" is the number of variables. The variables are typically written to the left in alphabetic order, and one or more columns of results are shown to the right.

X	Y	X + Y
0	0	0
0	1	1
1	0	1
1	1	1

Illustration 12: Truth Table - Addition

```
for a in [False, True]:
    for b in [False, True]:
        print(a, b, a or b)
```

```
False False False
False True True
True False True
True True True
```

If you have a particularly complex expression, it is often best to calculate parts separately and to put the final result as the right most column. Take for instance (AB+B'C')(AC+C')

A	B	C	AB	B'C'	AB+B'C'	AC	C'	AC+C'	(AB+B'C')(AC+C')
0	0	0	0	1	1	0	1	1	1
0	0	1	0	1	1	0	0	0	0
0	1	0	0	1	1	0	1	1	1
0	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	0	1	1	1
1	0	1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1	1	1
1	1	1	1	0	1	1	0	1	1

Illustration 13: Truth Table - (AB+B'C')(AC+C')

We can verify our truth table using the `for` statement in Python. This example uses three for loops nested within each other to create the 8 different combinations if the three Boolean variables. Also notice that the for loops are iterating through the two Boolean values.

```
for a in [False, True]:
    for b in [False, True]:
```

```
        for c in [False,True]:
            print(a, b, c, (a and b or not(b and c)) and (a and c or not(c)))
```

```
False False False True
False False True False
False True False True
False True True False
True False False True
True False True True
True True False True
True True True True
```

It is interesting to note in the example above, that the final column is the same as the AC+C' column and that this simpler expression is exactly the same.

Comparing Two Values in Python

Comparison operators look at two values and return True or False depending on the values. The siz most commonly used ones are:

- `==` — Equal.
- `!=` — Not Equal.
- `<=` — Less than or equal.
- `>=` — Greater than or equal.
- `<` — Less than.
- `>` — Greater than.

== != < <= > >= Comparison Operators

The comparison operators of equal, not equal, less than, less than or equal, greater than, and greater than or equal.

https://docs.python.org/3/reference/expressions.html#comparisons
http://bit.ly/2FeNJVk

An example of comparing some values, follows:

```
print(10 < 10)
print(10 <= 10)
```

```
print(10 == 10.0)
print("able" != "unable")
```

```
False
True
True
True
```

NOTE: An error will be thrown if you try to compare expressions that are of different types.

This last example puts it all of the Boolean together, try running this program with several values of a:

```
a = int(input("enter an integer"))
print("a is", a)
print("a is between 10 and 20 (inclusive)", a >= 10 and a <= 20)
print("a is greater than 100 or less than 3", a > 100 or a < 3)
print("a is 2 or 3 or between 20 and 30", a == 2 or a ==3 or (a
   >= 20 and a <= 30))
```

```
a is 2
a is between 10 and 20 (inclusive) False
a is greater than 100 or less than 3 True
a is 2 or 3 or between 20 and 30 True
```


Summary

Goes here

Important Terms

- Boole
- Boolean
- False
- True
- addition
- and
- compliment
- equal
- greater than
- greater than or equal
- less than
- less than or equal
- multiplication
- not
- not equal
- one
- or
- zero

Exercises

Here

Word Search

```
b y i j q v n h z r e q u a l n n u r c n
o w f b s c e h x r r d v t k d o m · y h
o w l s m w c b d p · z y e s o t u m i ·
l d q z n o t · e q u a l o m z x l i · m
e o r j w e a d d i t i o n · · l t n z f
l c l e s s · t h a n g k · b c j i f e i
l o x · c v i i d s j u i d c t f p u r s
f y m h s z u c n c h m n r j d b l l o j
a k q d r b n e z y z g b v u q o i n z v
l g r e a t e r · t h a n · k · o c · w z
s u h c f c r n y y w o a f w i l a c b b
e c w t · s a r v c m w o p u q e t · e q
j u q r z n v x f u · w j w z a a i q x ·
g a x w z a n d i l i g b · k i n o k · g
t n r e g s z h h s t o n m z t i n m e t
g t j d f q m k r · a · u w z m e b k e j
g r e a t e r · t h a n · o r · e q u a l
f z b b · o q x l y g h d s n p b c q v x
x r p h c o m p l i m e n t e k · b q i l
r u l b t r u e l u u u o n e y · d x o v
l e s s · t h a n · o r · e q u a l k · ·
```

Boole, Boolean, False, True, addition, and, compliment, equal, greater than, greater than or equal, less than, less than or equal, multiplication, not, not equal, one, or, zero

References

Burris, Stanley. (2014). George Boole entry in the Stanford Encyclopedia of Philosophy. Retrieved 2016-03-18 from http://plato.stanford.edu/entries/boole/ .

Positional notation. (n.d.). Retrieved 2016-03-1, from http://www.merriam-webster.com/dictionary/positional notation .

Roth, C. (1985). *Fundamentals of Logic Design*, Third Edition. West Publishing Company. St. Paul, Minnesota.