3.1. Boolean values and operators

BOOLEAN VALUES:

Boolean values can be tested for truth value, and used for IF and WHILE condition. There are two values True and False. 0 is considered as False and all other values considered as True.

Boolean Operations:

Consider x=True, y= False

Operator	Example	Description
and	x and y- returns false	Both operand should be true
or	x or y- returns true	Anyone of the operand should be true
not	not x returns false	Not carries single operand

Modulus operator

The **modulus operator** works on integers and yields the remainder when the first operand is divided by the second. In Python, the modulus operator is a percent sign (%). The syntax is the same as for other operators:

```
>>> quotient = 7 / 3
>>> print quotient
2

>>> remainder = 7 % 3
>>> print remainder
1
```

So 7 divided by 3 is 2 with 1 left over.

The modulus operator turns out to be surprisingly useful. For example, you can check whether one number is divisible by another—if x % y is zero, then x is divisible by y.

Also, you can extract the right-most digit or digits from a number. For example, x % 10 yields the right-most digit of x (in base 10). Similarly x % 100 yields the last two digits.

Boolean expressions

A **boolean expression** is an expression that is either true or false. 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
```

True and False are special values that belong to the type bool; they are not strings:

```
>>> type(True)
<type 'bool'>

>>> type(False)
<type 'bool'>
```

The == operator is one of the **relational operators**; the others are:

```
x != y # x is not equal to y

x > y # x is greater than y

x < y # x is less than y

x >= y # x is greater than or equal to y

x <= y # x is less than or equal to y
```

Logical operators

There are three **logical operators**: and, or, and not. The semantics (meaning) of these operators is similar to their meaning in English. For 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 if 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, that is, if x is less than or equal to y. The operands of the logical operators should be boolean expressions. Any nonzero number is interpreted as “true.”

```
>>> 17 and True
True
```