1 Relational operators

The following relational operators allow us to compare (i.e., relate) two different data values to see which is larger, whether they are equal, and so on. The result is either `True` or `False`.

Description	Operator	Example	Result
less than	<	`5 < 6`	`True`
greater than	>	`6.0 > 6`	`False`
equal to	==	`5 == 6`	`False`
not equal to	!=	`5 != 6`	`True`
greater than or equal to	>=	`6.0 >= 6`	`True`
less than or equal to	<=	`6.1 <= 6`	`False`

2 Booleans

Recall that a data type is a set of values and operations on those values. The `bool` data type has just two possible values: `True` and `False`. The operators include `and` (true when both operands are true) and `or` (true when at least one operand is true). The operator `not` takes a single operand and "flips" it: true becomes false and false becomes true. The `bool` type is short for Boolean logic, the logic which underlies modern computing systems and is named after George Boole, a mathematician.

```
(x % 2 == 0 and x % 4 != 0) or (x % 3 == 0)
```

This evaluates to `True` when `x` is a number that is divisible by 3 or `x` is divisible by 2 but not 4. For example, if `x = 8` it evaluates to `False` but when when `x = 9` or `x = 10`, it evaluates to `True`.

3 If statement

An `if` statement has this form. A boolean expression is an expression that evaluates to `True` or `False`.

if boolean_expression:

statements

The indented statements are the **body**. The `if` statement is called a **conditional** statement because the body is only executed under certain conditions. It is executed when the boolean expression evaluates to `True`. When the boolean expression is `False`, python *skips* the body and continues to the next statement in the program.

```
grade = float(raw_input("How did you do on the exam? "))
if grade >= 90:
    print "You got an A!"
print "Keep studying hard!"
```

Suppose the grade is 93. Then the boolean expression `grade >= 90` is `True` and both statements are printed. When the grade is 87, only “Keep studying hard!” is printed.

4 If-Else statement

The following two `if` statements work correctly, but this is bad program design.

```
if grade >= 90:
    print "You got an A!"          # statement 1
if grade < 90:                     # BAD program design!
    print "You did not get an A."  # statement 2
```

This is an “either-or” situation: either the grade is above threshold and we print the first statement, or it is not and we print the second statement. A better program design is to use an `else` clause.

```
if grade >= 90:
    print "You got an A!"          # statement 1
else:                              # GOOD program design!
    print "You did not get an A."  # statement 2
```

This is better program design because the condition is stated only once. If the professor wants to change the threshold for an A to be 93, only one line of code needs to be revised.

5 If-Elif-Else statement

The following nested `if` statement works correctly but it is bad program design.

```
if grade >= 90:
    print "You got an A!"        # statement 1
else:
    if grade >= 80:              # BAD program design!
        print "You got a B."     # statement 2
    else:
        print "You got a C or lower."  # statement 3
```

Nested conditionals can be hard to read. It is better to use an `elif` statement. Below, statement 2 is executed only when $80 \leq$ `grade` < 90. Similarly, statement 3 is executed only when `grade` < 80.

```
if grade >= 90:
    print "You got an A!"          # statement 1
elif grade >= 80:                  # GOOD program design!
    print "You got a B."           # statement 2
else:
    print "You got a C or lower."  # statement 3
```