**Chapter 3**

**Conditional execution**

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.

True and False are special values that belong to the class bool; they are not

strings:

The == operator is one of the *comparison 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*

x is y *# x is the same as y*

x is not y *# x is not the same as 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.

**Conditional execution**

In order to write useful programs, we almost always need the ability to check conditions

and change the behavior of the program accordingly. *Conditional statements*

give us this ability. The simplest form is the if statement:

The boolean expression after the if statement is called the *condition*. We end the

if statement with a colon character (:) and the line(s) after the if statement are

indented.

If the logical condition is true, then the indented statement gets executed. If the

logical condition is false, the indented statement is skipped.

if statements have the same structure as function definitions or for loops1. The

statement consists of a header line that ends with the colon character (:) followed

by an indented block. Statements like this are called *compound statements* because

they stretch across more than one line.

There is no limit on the number of statements that can appear in the body, but

there must be at least one. Occasionally, it is useful to have a body with no

statements (usually as a place holder for code you haven’t written yet). In that

case, you can use the pass statement, which does nothing.

**Alternative execution**

A second form of the if statement is *alternative execution*, in which there are two

possibilities and the condition determines which one gets executed.

**Chained conditionals**

Sometimes there are more than two possibilities and we need more than two

branches. One way to express a computation like that is a *chained conditional*.

**Nested conditionals**

One conditional can also be nested within another.

The outer conditional contains two branches. The first branch contains a simple

statement. The second branch contains another if statement, which has two

branches of its own. Those two branches are both simple statements, although

they could have been conditional statements as well.

The idea of try and

except is that you know that some sequence of instruction(s) may have a problem

and you want to add some statements to be executed if an error occurs. These

extra statements (the except block) are ignored if there is no error.

You can think of the try and except feature in Python as an “insurance policy”

on a sequence of statements.

**3.10 Glossary**

**body** The sequence of statements within a compound statement.

**boolean expression** An expression whose value is either True or False.

**branch** One of the alternative sequences of statements in a conditional statement.

**chained conditional** A conditional statement with a series of alternative

branches.

**comparison operator** One of the operators that compares its operands: ==, !=,

>, <, >=, and <=.

**conditional statement** A statement that controls the flow of execution depending

on some condition.

**condition** The boolean expression in a conditional statement that determines

which branch is executed.

**compound statement** A statement that consists of a header and a body. The

header ends with a colon (:). The body is indented relative to the header.

**guardian pattern** Where we construct a logical expression with additional comparisons

to take advantage of the short-circuit behavior.

**logical operator** One of the operators that combines boolean expressions: and,

or, and not.

**nested conditional** A conditional statement that appears in one of the branches

of another conditional statement.

**traceback** A list of the functions that are executing, printed when an exception

occurs.

**short circuit** When Python is part-way through evaluating a logical expression

and stops the evaluation because Python knows the final value for the expression

without needing to evaluate the rest of the expression.