Branching/Conditionals (If statements)

The differences between Python and C++ conditional statements are only syntactical. The semantics --- how they work --- is exactly the same.

Python	C++

if *test*:
  statement
  statement
  …more statements…
	if (*test*)
	{
	  statement
	  statement
	  …more statements…
	}

if *test*:
  statement
  statement
  …more statements…
else*:*
  statement
  statement
  …more statements…
	if (*test*)
	{
	  statement
	  statement
	  …more statements…
	}
	else
	{
	  statement
	  statement
	  …more statements…
	}

if *test1*:
  statement
  statement
  …more statements…
elif *test2*:
  statement
  statement
  …more statements…
…more elif tests…
else*:*
  statement
  statement
  …more statements…
	if (*test1*)
	{
	  statement
	  statement
	  …more statements…
	}
	else if (*test2*)
	{
	  statement
	  statement
	  …more statements…
	}
	…more else if tests…
	else
	{
	  statement
	  statement
	  …more statements…
	}

• Indentation implies which statements form the body of the conditional. • Must use a colon after each test.	• Curly braces imply which statements form the body. If there is only one statement inside the body, the braces can be omitted. • Must surround each test with parentheses.

The relational operators are the same in both languages: == (equality), != (inequality), < (less than), <= (less than or equal to), > (greater than), >= (greater than or equal to).

The logical operators are syntactically different (though they work identically in both languages):

Python: and or not

C++: && || !

Switch statement (only in C++)

A common use of an `if-else if…else` statement is to compare a variable against a sequence of constants with an equality test. For instance:

```
char grade;
cin >> grade;
if (grade == ‘A’) {
   cout << “Your work is excellent!” << endl;
} else if (grade == ‘B’) {
   cout << “Your work is good!” << endl;
} else if (grade == ‘C’) {
   cout << “Your work is average.” << endl;
} else if (grade == ‘D’) {
   cout << “Your work is below average!” << endl;
} else if (grade == ‘F’) {
   cout << “You are failing!” << endl;
} else {
   cout << “You typed an invalid letter grade.” << endl;
}
```

This could be replaced with:

```
char grade;
cin >> grade;
switch (grade)
{
   case ‘A’:
      cout << “Your work is excellent!” << endl;
      break;
   case ‘B’:
      cout << “Your work is good!” << endl;
      break;
   case ‘C’:
      cout << “Your work is average!” << endl;
      break;
   case ‘D’:
      cout << “Your work is below average!” << endl;
      break;
   case ‘F’:
      cout << “You are failing!” << endl;
      break;
   default:
      cout << “You typed an invalid letter grade.” << endl;
      break;
}
```

A `switch` statement can only be used if the variable in question is an `int` or a `char`, and the variable can only be compared against a single constant at a time, and only for an equality test. Notice the use of the `break` statement in each case. When using `switch`, as soon as the first case that matches the variable is found, *all* of the case blocks following the matching one are executed. To avoid this behavior, use a `break` statement inside each case block. This behavior can be used to allow multiple values to trigger execution of a single case block (see the Zybook section 3.5 for an example).