## Compound Boolean Expressions

To be eligible to graduate from Loyola University Chicago, you must have 120 credits and a GPA of at least 2.0. This translates directly into Python as acompound condition:

credits >= 120 **and** GPA >=2.0

This is true if both credits >= 120 is true and GPA >= 2.0 is true. A short example program using this would be:

credits = float(input('How many units of credit do you have? '))

GPA = float(input('What is your GPA? '))

**if** credits >= 120 **and** GPA >=2.0:

print('You are eligible to graduate!')

**else**:

print('You are not eligible to graduate.')

The new Python syntax is for the operator and:

condition1 and condition2

The compound condition is true if both of the component conditions are true. It is false if at least one of the conditions is false.

See [Congress Exercise](http://anh.cs.luc.edu/python/hands-on/3.1/handsonHtml/ifstatements.html#congressex).

In the last example in the previous section, there was an if-elif statement where both tests had the same block to be done if the condition was true:

**if** x < xLow:

dx = -dx

**elif** x > xHigh:

dx = -dx

There is a simpler way to state this in a sentence: If x < xLow or x > xHigh, switch the sign of dx. That translates directly into Python:

**if** x < xLow **or** x > xHigh:

dx = -dx

The word or makes another compound condition:

condition1 or condition2

is true if at least one of the conditions is true. It is false if both conditions are false. This corresponds to one way the word “or” is used in English. Other times in English “or” is used to mean exactly one alternative is true.

**Warning**

When translating a problem stated in English using “or”, be careful to determine whether the meaning matches Python’s or.

It is often convenient to encapsulate complicated tests inside a function. Think how to complete the function starting:

**def** isInside(rect, point):

*'''Return True if the point is inside the Rectangle rect.'''*

pt1 = rect.getP1()

pt2 = rect.getP2()

Recall that a Rectangle is specified in its constructor by two diagonally oppose Points. This example gives the first use in the tutorials of the Rectanglemethods that recover those two corner points, getP1 and getP2. The program calls the points obtained this way pt1 and pt2. The x and y coordinates ofpt1, pt2, and point can be recovered with the methods of the Point type, getX() and getY().

Suppose that I introduce variables for the x coordinates of pt1, point, and pt2, calling these x-coordinates end1, val, and end2, respectively. On first try you might decide that the needed mathematical relationship to test is

end1 <= val <= end2

Unfortunately, this is not enough: The only requirement for the two corner points is that they be diagonally opposite, not that the coordinates of the second point are higher than the corresponding coordinates of the first point. It could be that end1 is 200; end2 is 100, and val is 120. In this latter caseval is between end1 and end2, but substituting into the expression above

200 <= 120 <= 100

is False. The 100 and 200 need to be reversed in this case. This makes a complicated situation. Also this is an issue which must be revisited for both the x and y coordinates. I introduce an auxiliary function isBetween to deal with one coordinate at a time. It starts:

**def** isBetween(val, end1, end2):

*'''Return True if val is between the ends.*

*The ends do not need to be in increasing order.'''*

Clearly this is true if the original expression, end1 <= val <= end2, is true. You must also consider the possible case when the order of the ends is reversed: end2 <= val <= end1. How do we combine these two possibilities? The Boolean connectives to consider are and and or. Which applies? You only need one to be true, so or is the proper connective:

A correct but redundant function body would be:

**if** end1 <= val <= end2 **or** end2 <= val <= end1:

**return** **True**

**else**:

**return** **False**

Check the meaning: if the compound expression is True, return True. If the condition is False, return False - in either case return the same value as the test condition. See that a much simpler and neater version is to just return the value of the condition itself!

**return** end1 <= val <= end2 **or** end2 <= val <= end1

**Note**

In general you should not need an if-else statement to choose between true and false values! Operate directly on the boolean expression.

A side comment on expressions like

end1 <= val <= end2

Other than the two-character operators, this is like standard math syntax, chaining comparisons. In Python any number of comparisons can be chainedin this way, closely approximating mathematical notation. Though this is good Python, be aware that if you try other high-level languages like Java and C++, such an expression is gibberish. Another way the expression can be expressed (and which translates directly to other languages) is:

end1 <= val **and** val <= end2