**Go With the Flow**

Just like in real life, sometimes we'd like our code to be able to make decisions.

The Python programs we've written so far have had one-track minds: they can add two numbers or print something, but they don't have the ability to pick one of these outcomes over the other.

**Control flow** gives us this ability to choose among outcomes based on what else is happening in the program.

def clinic():

print "You've just entered the clinic!"

print "Do you take the door on the left or the right?"

answer = raw\_input("Type left or right and hit 'Enter'.").lower()

if answer == "left" or answer == "l":

print "This is the Verbal Abuse Room, you heap of parrot droppings!"

elif answer == "right" or answer == "r":

print "Of course this is the Argument Room, I've told you that already!"

else:

print "You didn't pick left or right! Try again."

clinic()

clinic()

**Compare Closely!**

Let's start with the simplest aspect of control flow: **comparators**. There are six:

**Equal to** (==)

>>> 2 == 2 True >>> 2 == 5 False

**Not equal to** (!=)

>>> 2 != 5 True >>> 2 != 2 False

**Less than** (<)

>>> 2 < 5 True >>> 5 < 2 False

**Less than or equal to** (<=)

>>> 2 <= 2 True >>> 5 <= 2 False

**Greater than** (>)

>>> 5 > 2 True >>> 2 > 5 False

**Greater than or equal to** (>=)

>>> 5 >= 5 True >>> 2 >= 5 False

Comparators check if a value is (or is not) equal to, greater than (or equal to), or less than (or equal to) another value.

Note that == compares whether two things are equal, and = assigns a value to a variable.

# Remember that \*\* can be read as 'to the power of'. 3\*\*16 is about 43 million.

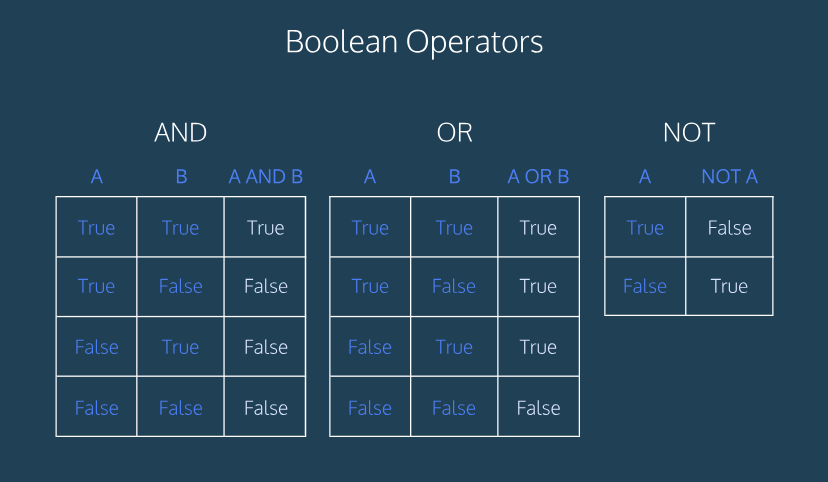
Remember, comparators are: ==, !=, >, >=, <, and <=.

# To Be and/or Not to Be

**Boolean operators** compare statements and result in boolean values. There are three boolean operators:

1. and, which checks if both the statements are True;
2. or, which checks if at least one of the statements is True;
3. not, which gives the opposite of the statement.

We'll go through the operators one by one.



**And**

The boolean operator and returns Truewhen the expressions on both sides of and are true. For instance:

* 1 < 2 and 2 < 3 is True;
* 1 < 2 and 2 > 3 is False.
* Remember that a number raised to the 1/2 power is equal to the square root of that number! (For example, 25 \*\* 0.5 is equal to the square root of 25, which is 5.)
* Be careful with bool\_four—there's a big difference between -1 \*\* 2 and (-1) \*\* 2! This is meant to be tricky.

**Or**

The boolean operator or returns Truewhen at least one expression on either side of or is true. For example:

* 1 < 2 or 2 > 3 is True;
* 1 > 2 or 2 > 3 is False.

**Not**

The boolean operator not returns Truefor false statements and False for true statements.

For example:

* not False will evaluate to True, while not 41 > 40 will return False.

# This and That (or This, But Not That!)

Boolean operators aren't just evaluated from left to right. Just like with arithmetic operators, there's an order of operations for boolean operators:

1. not is evaluated first;
2. and is evaluated next;
3. or is evaluated last.

For example, True or not False and Falsereturns True. If this isn't clear, look at the Hint.

Parentheses () ensure your expressions are evaluated in the order you want. Anything in parentheses is evaluated as its own unit.

**Conditional Statement Syntax**

if is a conditional statement that executes some specified code after checking if its expression is True.

Here's an example of if statement syntax:

if 8 < 9: print "Eight is less than nine!"

In this example, 8 < 9 is the checked expression and print "Eight is less than nine!" is the specified code.

Pay attention to the indentation before the print statement. This space, called *white space*, is how Python knows we are entering a new *block* of code. Python accepts many different kinds of indentation to indicate blocks. In this lesson, we use four spaces but elsewhere you might encounter two-space indentation or tabs (which Python will see as different from spaces).

If the indentation from one line to the next is different and there is no command (like if) that indicates an incoming block then Python will raise an IndentationError. These errors could mean, for example, that one line had two spaces but the next one had three. Python tries to indicate where this error happened by printing the line of code it couldn't parse and using a ^ to point to where the indentation was different from what it expected.

**If You're Having...**

Let's get some practice with ifstatements. Remember, the syntax looks like this:

if some\_function(): # block line one # block line two # et cetera

Looking at the example above, in the event that some\_function() returns True, then the indented block of code after it will be executed. In the event that it returns False, then the indented block will be skipped.

Also, make sure you notice the colons at the end of the if statement. We've added them for you, but they're important.

**Else Problems, I Feel Bad for You, Son...**

The else statement complements the ifstatement. An if/else pair says: "If this expression is true, run this indented code block; otherwise, run this code after the else statement."

Unlike if, else doesn't depend on an expression. For example:

if 8 > 9: print "I don't printed!" else: print "I get printed!"

**I Got 99 Problems, But a Switch Ain't One**

elif is short for "else if." It means exactly what it sounds like: "otherwise, if the following expression is true, do this!"

if 8 > 9: print "I don't get printed!" elif 8 < 9: print "I get printed!" else: print "I also don't get printed!"

In the example above, the elif statement is only checked if the original ifstatement is False.

Make sure the if and elifstatements end with colons :

Your code should look something like:

if EXPRESSION: # do something elif OTHER EXPRESSION: # do something else: # do something

Make sure the if and elifstatements end with colons :

Your code should look something like:

if EXPRESSION: # do something elif OTHER EXPRESSION: # do something else: # do something

# Complete the if and elif statements!

def grade\_converter(grade):

if grade >= 90:

return "A"

elif grade < 90 and grade >=80:

return "B"

elif grade < 80 and grade >=70:

return "C"

elif grade < 70 and grade >=65:

return "D"

else:

return "F"

# This should print an "A"

print grade\_converter(92)

# This should print a "C"

print grade\_converter(70)

# This should print an "F"

print grade\_converter(61)