Boolean Variables in Python

Previously, we have learned about three Python data types: ints, floats, and strings. Recall that a data type is a piece of information attached to a value or variable that tells the programming language how to interpret that variable or value:

```
x = 42          # x is an integer (int) variable (b/c 42 is an integer value.)
y = 42.7        # y is a floating-point (float) variable (because 42.7 is a
                # floating-point value.)
z = "hello"     # z is a string (str) variable (b/c "hello" is a string value.)
```

Python has many other data types to represent other types of information. One important type is the *Boolean* data type. There are only two possible values for that a Boolean can be, True or False. This might seem strange because ints, floats, and strings have potentially an infinite number of possible values, but Booleans have only two! They are used normally used situations where you need to represent the truth of something.

Aside: The Boolean data type is named after George Boole, a mathematician and logician.

To create and use Boolean variables in Python, you can just use the special Python keywords True and False (they must be capitalized). These are the only two values that may be stored in a Boolean variable.

```
game_over = True      # game_over is a Boolean variable holding the value True
ac_broken = False     # ac_broken is a Boolean variable holding the value False
```

Note: it is very tempting to put quotes around the values True and False, however, this will create string variables, not Boolean variables! "True" and "False" are not the same thing as "True" and "False"!

What are Boolean variables good for?

Boolean variables can be used in the condition part of an if statement:

```
if game_over:
  print("Thanks for playing!")
```

Note that the following works, but is considered poor practice:

```
if game_over == True:           # the == True part is superfluous.
  print("Thanks for playing!")
```

This idea is handy when there are might be many conditions that could end the game, each happening in a different part of your code. Imagine a game that could be won or lost in many different ways, each requiring a different section of code. Each section could test the appropriate condition and set the game_over variable to True if the user happens to win or lose.

Can it be used for other things?

Boolean variables are also commonly used as the return value of a function that is designed to determine whether something is true or not. For instance, consider a function that is designed to test whether a number is even or odd. This can be done by dividing the number by 2 and examining the remainder. An even number, when divided by 2, has a remainder of 0, whereas an odd number has a remainder of one. Consider a function designed to return `True` if a number if even, and `False` if a number is odd:

```
def is_even(num):
  if num % 2 == 0:
    return True
  else:
    return False
```

Here's what happens when you call this function from the Python shell:

```
>>> is_even(6)
True
>>> is_even(7)
False
```

Now this function can be used elsewhere in your program:

```
n = int(input("Enter an integer: "))
even = is_even(n)                     # Capture the Boolean return value.
if even:                              # Test the value of the variable even.
  print("Your number is even.")
else:
  print("Your number is odd.")
```

This code is often shortened because there's no need to create a separate variable to hold the return value:

```
n = int(input("Enter an integer: "))
if is_even(n):                        # Call is_even and run the test together!
  print("Your number is even.")
else:
  print("Your number is odd.")
```

Notice the line of code: `if is_even(n):`. When you call a function in the condition part of an if statement, Python calls the function, then uses *the return value of the function call as the if-condition*. In other words, that single line of code acts as if you called the `is_even` function, passing n as the argument, then captured the return value, saved it in a Boolean variable, and used that variable as the condition in the if statement.