## String Delimiters

A string in Python is a sequence of characters. For Python to recognize a sequence of characters, like hello, as a string, it must be enclosed in quotes to delimit the string.

For this whole section on strings, continue trying each set-off line of code in the Shell. Try

"hello"

Note that the interpreter gives back the string with single quotes. Python does not care what system you use. Try

'Hi!'

Having the choice of delimiters can be handy.

Figure out how to give Python the string containing the text: I'm happy. Try it. If you got an error, try it with another type of quotes, and figure out why that one works and not the first.

**Note**

A string can have any number of characters in it, including 0. The empty string is '' (two quote characters with nothing between them). Many beginners forget that having no characters is legal. It can be useful.

Strings are a new Python type. Try

type('dog')

type('7')

type(7)

The last two lines show how easily you can get confused! Strings can include any characters, including digits. Quotes turn even digits into strings. This will have consequences in the next section....

## String Concatenation

Strings also have operation symbols. Try in the Shell (noting the space after very):

'very ' + 'hot'

The plus operation with strings means concatenate the strings. Python looks at the type of operands before deciding what operation is associated with the +.

Think of the relation of addition and multiplication of integers, and then guess the meaning of

3\*'very ' + 'hot'

Were you right? The ability to repeat yourself easily can be handy.

Predict the following and then test. Remember the last section on types:

7+2

'7'+'2'

Python checks the types and interprets the plus symbol based on the type. Try

'7'+2

With mixed string and int types, Python sees an ambiguous expression, and does not guess which you want - it just gives an error!

This is a traceback error. These occur when the code is being executed. In the last two lines it shows the line where the error was found, and then a reason for the error. Not all reasons are immediately intelligible to a starting programmer, but they are certainly worth checking out. In this case it is pretty direct. You need to make an explicit conversion, so both are strings if you mean concatenation, '7' + str(2), or so both are int if you mean addition, int('7') + 2.

With literal strings these examples are only useful for illustration: There is no reason to write such verbose expressions when you already know the intended result. With variables, starting in the next section, expressions involving these conversions become more important.