Booleans

Booleans are either true or false. Python has two constants, cleverly named `True` and `False`, which can be used to assign boolean values directly. Expressions can also evaluate to a boolean value. In certain places (like `if` statements), Python expects an expression to evaluate to a boolean value. These places are called *boolean contexts*. You can use virtually any expression in a boolean context, and Python will try to determine its truth value. Different datatypes have different rules about which values are true or false in a boolean context. (This will make more sense once you see some concrete examples later in this chapter.)

You can use virtually any expression in a boolean context.

For example, take this snippet from `humansize.py`:

```
if size < 0:
    raise ValueError('number must be non-negative')
```

`size` is an integer, 0 is an integer, and `<` is a numerical operator. The result of the expression `size < 0` is always a boolean. You can test this yourself in the Python interactive shell:

```
size = 1
print (size < 0)
#False

size = 0
print (size < 0)
#False

size = -1
print (size < 0)
#True
```

Due to some legacy issues left over from Python 2, booleans can be treated as numbers. `True` is 1; `False` is 0.

```
print (True + True)
#2

print (True - False)
#1

print (True * False)
#0

print (True / False)

#Traceback (most recent call last):
#   File "/usercode/__ed_file.py", line 10, in <module>
# print (True / False)
#ZeroDivisionError: division by zero
```

Ew, ew, ew! Don’t do that. Forget I even mentioned it.