Native Data Types in Python

Every object in Python has a *type*. The `type` function allows us to inspect the type of an object.

```
>>> type(today)
<class 'datetime.date'>
```

So far, the types of objects we have used extensively are relatively few: numbers, functions, Booleans, and now dates. We also briefly encountered sets and strings in Chapter 1, but we will need to study those in more depth. There are many other kinds of objects --- sounds, images, locations, data connections, etc. --- most of which can be defined by the means of combination and abstraction that we develop in this chapter. Python has only a handful of primitive or *native* data types built into the language.

Native data types have the following properties:

1. There are primitive expressions that evaluate to objects of these types, called *literals*.
2. There are built-in functions, operators, and methods to manipulate these types of values.

As we have seen, numbers are native; numeric literals evaluate to numbers, and mathematical operators manipulate number objects.

```
>>> 12 + 3000000000000000000000000
3000000000000000000000012
```

In fact, Python includes three native numeric types: integers (`int`), real numbers (`float`), and complex numbers (`complex`).

```
>>> type(2)
<class 'int'>
>>> type(1.5)
<class 'float'>
```

```
>>> type(1+1j)
<class 'complex'>
```

The name `float` comes from the way in which real numbers are represented in Python: a "floating point" representation. While the details of how numbers are represented is not a topic for this text, some high-level differences between `int` and `float` objects are important to know. In particular, `int` objects can only represent integers, but they represent them exactly, without any approximation. On the other hand, `float` objects can represent a wide range of fractional numbers, but not all rational numbers are representable. Nonetheless, float objects are often used to represent real and rational numbers approximately, up to some number of significant figures.

Python has many native datatypes. Here are the important ones:

1. **Booleans** are either `True` or `False`.
2. **Numbers** can be integers (`1` and `2`), floats (`1.1` and `1.2`), fractions (`1/2` and `2/3`), or even complex numbers.
3. **Strings** are sequences of Unicode characters, *e.g.* an html document.
4. **Bytes** and **byte arrays**, *e.g.* a jpeg image file.
5. **Lists** are ordered sequences of values.
6. **Tuples** are ordered, immutable sequences of values.
7. **Sets** are unordered bags of values.
8. **Dictionaries** are unordered bags of key-value pairs.

Of course, there are more types than these. Everything is an object in Python, so there are types like *module*, *function*, *class*, *method*, *file*, and even *compiled code*. You've already seen some of these: modules have names, functions have `docstrings`, &c. You'll learn about classes in Classes & Iterators, and about files in Files.

Strings and bytes are important enough — and complicated enough — that they get their own chapter. Let's look at the others first.

Source : http://inst.eecs.berkeley.edu/~cs61A/book/chapters/objects.html#native-data-types