A Note About Python Objects & Variables

> In Python, *everything* is an **object** in the sense that it can be assigned to a variable or passed as an argument to a function.

An **object** is a block of information, which *has* (or, *is of*) some **data type**. For instance, in this code `a = "Hi"`, the variable `a` is simply a label (or a box) that has a **reference** to the string literal `"Hi"`, and the string literal `"Hi"` is an object. Furthermore, the variable `a` has a value that is of `str` string data type, and the string literal tiself `"Hi"` is of string data type.

```
>>> my_fridge = "my refrigerator"
>>> my_fridge
"my refrigerator"
```

In the above example, `my_fridge` is the sticky note or the label that you put on to your refrigerator, and the string literal `"my refrigerator"` is the actual representation of your refrigerator. Imagine yourself in your kitchen, and you have a sticky note and a pen in your hand. You write "my_fridge" on the sticky note and then you stick it on the door of you kitchen fridge. In this case, the sticky note with the word "my_fridge" is your variable, and the value for the variable is your refrigerator itself. In the same vein, your refrigerator is an **object** that has been **referenced** by your **variable**.

> In Python, all values of all its data types are **objects**, and even all the Python functions and the functions that you define for yourself are **objects**. They can be assigned to a variable or passed as an argument to a function; yes, you can even assign a function to a variable or even pass it to another function as an argument.

Python Data Types

In Python, there are a total of **eleven *built-in basic*** data types: `int`, `float`, `long`, `complex`, `bool`, `str`, `list`, `tuple`, `dict`, `set`, `frozenset`, `bytes`. Python has more data types (higher-level data types), but the data types that are define in this document are the basic building blocks of other higher-level data types are created. In practice, we will rarely encounter objects that are of **long**, **complex**, **frozenset**, and **bytes** data types.

In this document, we will focus on a total of **eight** built-in data types that are most commonly used in many Python applications: `int`, `float`, `bool`, `str`, `list`, `tuple`, `dict`, `set`.

Numeric Data Type

There are a total of **four** numeric data types in Python: **int**, **float**, **long**, and **complex**. Python `int` is written as `I = -12`, `float` as `F = 1.2`, `long` as `l = 123L`, and `complex` as `C = 2.4+0j`. We will rarely encounter objects that are of **long** and **complex** data types.

Boolean Data Type

Booleans represent the truth values `False` and `True`. The two objects reresenting the values `False` and `True` are the only Boolean objects in Python. The Boolean type is a subtype of the **int** data type; which means that the Boolean values behave like the **int**eger values `0` and `1` (`0` for `False` and `1` for `True`). Python booleans are written in the following way: `B1 = True`, `B2 = False`.

String Data Type

An object that is of the string data type can simply be called a "string". Strings can be enclosed in single quotes or double quotes. A string is a sequence of characters (i.e. letters, numbers, symbols). Python strings are written in the following way: `S1 = 'foo'`, `S2 = "bar"`.

List Data Type

An object that is of list data type can simply be called a "list". A list is the most versatile object in Python and it can be written as a sequence of comma-separated objects, all enclosed in square brackets. Lists can contain items of different types. Python lists are written in square brackets like so: `L = [1, 2, 3]`.

Tuple Data Type

An object that is of tuple data type can simply be called a "tuple". A tuple is like a list, but once it is created, none of the items within tuple cannot be modified during the execution of the program. Python tuples are written with parentheses like so: `T = (1, 2, 3)`.

Dictionary Data Type

An object that is of dictionary data type can simply be called a "dictionary". A dictionary is a collection of key-value entries/items. All the entries in a dictionary is unordered, modficable/mutable, and can be indexed by their keys. A dictionary must be written with curly brackets, and for every entry, there has to be a key, followed by a colon `:`, and then the value associated with the key. You can write a dictionary in the following way: `D = {"A": 2.4, "B": 3}`. Both the keys and the values of a dictionary can be of any data type mentioned above.

Set Data Type

An object that is of set data type can simply be called a "set". A set is an unordered collection of items which is *iterable* (i.e. you can iterate through the items of the object using for-loop), mutable, and has no duplicate elements. The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific item is contained in the set. In Python, a set is written with curly brackets: `S = {"Apple", "Banana", "Cherry"}`.