DATA 601

12 September 2018

Learning Python Chapter 8: Lists and Dictionaries

This chapter introduces two important Python data types – Lists and Dictionaries – and gives an overview of their use and syntax, their methods and attributes, and some of the ways they have changed in Python 3 from Python 2.

Lists are mutable, ordered, indexed, and heterogeneous. Technically, lists contain references to objects but, unless explicitly asked for, will always access the object itself but not the reference. Lists remind me of arrays of pointers that always access the object. Unlike strings, lists are mutable meaning they can be changed in place rather than creating a new object with every change. Because lists are ordered, they can be accessed by offset. Like strings, lists can be indexed and sliced. Lists are described as highly flexible and useful data types.

Dictionaries are described as similarly flexible and useful but with different characteristics. Dictionaries are sets of key-value pairs. Unlike lists, dictionaries are unordered collections of arbitrary objects. Like lists, dictionaries are variable length, heterogeneous, arbitrarily nest-able, and mutable. Unlike lists, dictionaries are accessed by key and are unordered which means that sequence operations cannot be performed on dictionaries. Each key in a dictionary can have only one value but this is not a one-to-one relationship; the same value may be assigned to multiple keys. Keys can be any immutable type but are often strings.

The chapter also discusses some important changes in how both lists and dictionaries are handled between Python 2 and 3. For example, lists of mixed type can no longer be sorted nor can magnitude comparisons in dictionaries be directly performed in Python 3. Several dictionary methods now return iterable view objects rather and lists but these objects can be forced to be lists if needed. The chapter does mention some reasons why one might choose a list over a dictionary (if order is important) or vice versa (a sparsely populated data structure or for named data where a key makes more sense than an offset). Both lists and dictionaries are powerful data types and have their various strengths.