**Explicit indexes**

In chapter one, you saw that DataFrames are composed of three parts: a NumPy array for the data, and two indexes to store the row and column details.

**dog dataset, revisited**

Here's the dog dataset again.

**.columns and .index**

Recall that dot-columns contains an Index object of column names, and dot-index contains an Index object of row numbers.



**Setting a column as the index**

You can move a column from the body of the DataFrame to the index. This is called "setting an index," and it uses the set\_index method. Notice that the output has changed slightly; in particular, a quick visual clue that name is now in the index is that the index values are left-aligned rather than right-aligned.

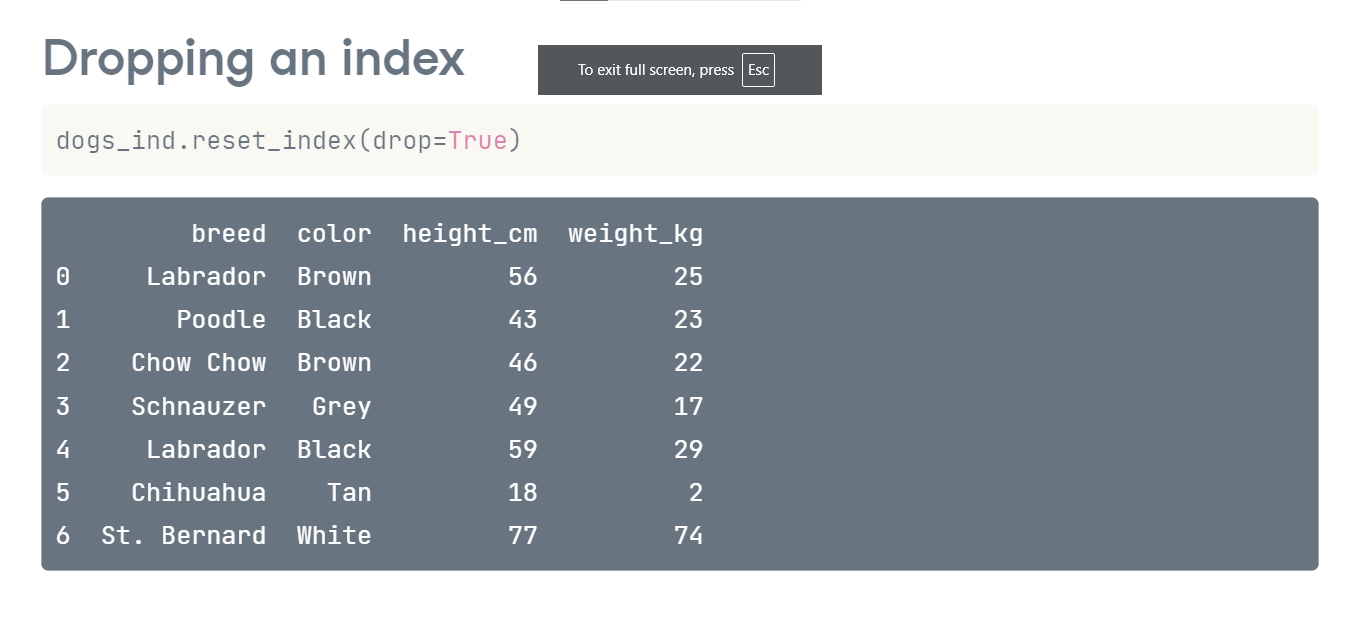


**Removing an index**

To undo what you just did, you can reset the index - that is, you remove it. This is done via reset\_index.

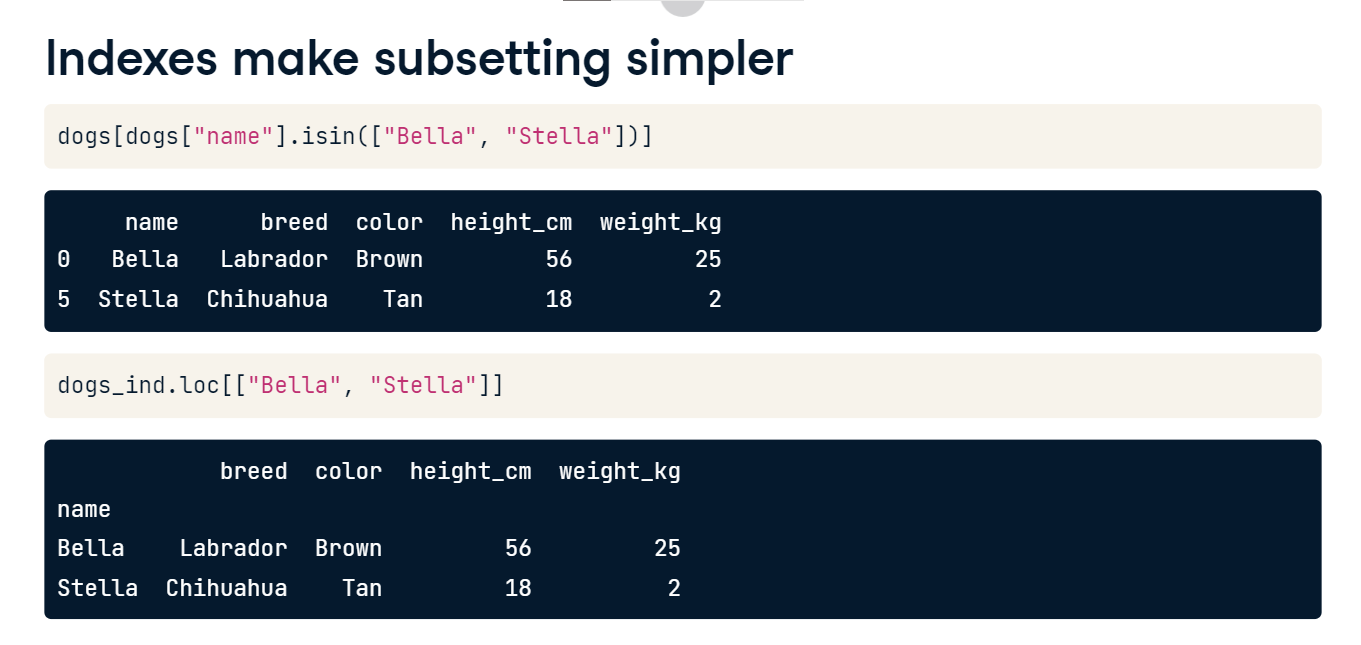
**Dropping an index**

reset\_index has a drop argument that allows you to discard an index. Here, setting drop to True entirely removes the dog names.



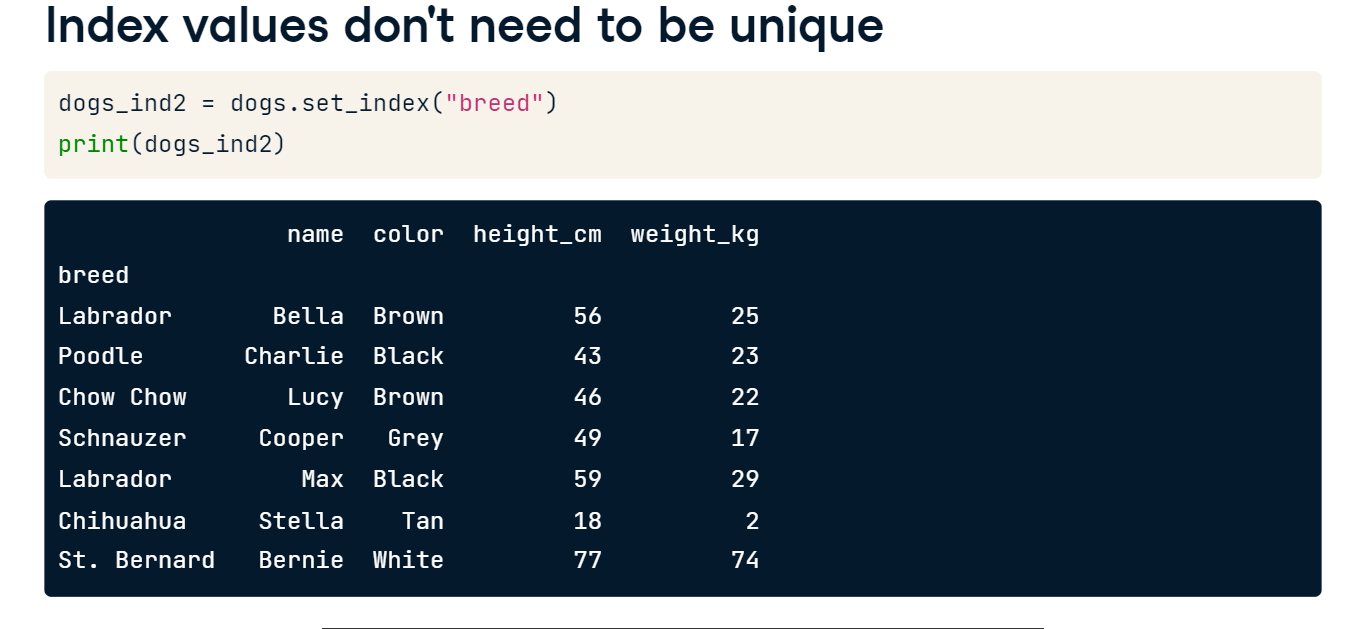
**Indexes make subsetting simpler**

You may be wondering why you should bother with indexes. The answer is that it makes subsetting code cleaner. Consider this example of subsetting for the rows where the dog is called Bella or Stella. It's a fairly tricky line of code for such a simple task. Now, look at the equivalent when the names are in the index. DataFrames have a subsetting method called "loc," which filters on index values. Here you simply pass the dog names to loc as a list. Much easier!



**Index values don't need to be unique**

The values in the index don't need to be unique. Here, there are two Labradors in the index.

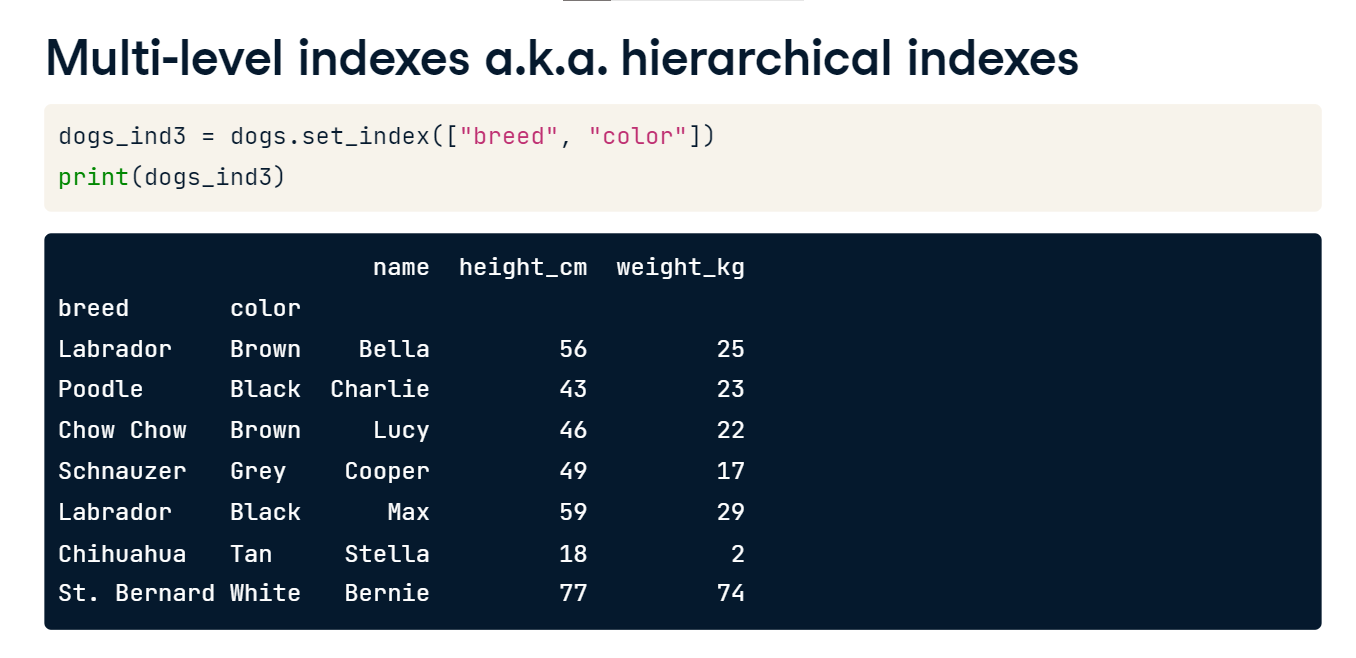


**Subsetting on duplicated index values**

Now, if you subset on "Labrador" using loc, all the Labrador data is returned.

**Multi-level indexes a.k.a. hierarchical indexes**

You can include multiple columns in the index by passing a list of column names to set\_index. Here, breed and color are included. These are called multi-level indexes, or hierarchical indexes: the terms are synonymous. There is an implication here that the inner level of index, in this case, color, is nested inside the outer level, breed.



**Subset the outer level with a list**

To take a subset of rows at the outer level index, you pass a list of index values to loc. Here, the list contains Labrador and Chihuahua, and the resulting subset contains all dogs from both breeds.



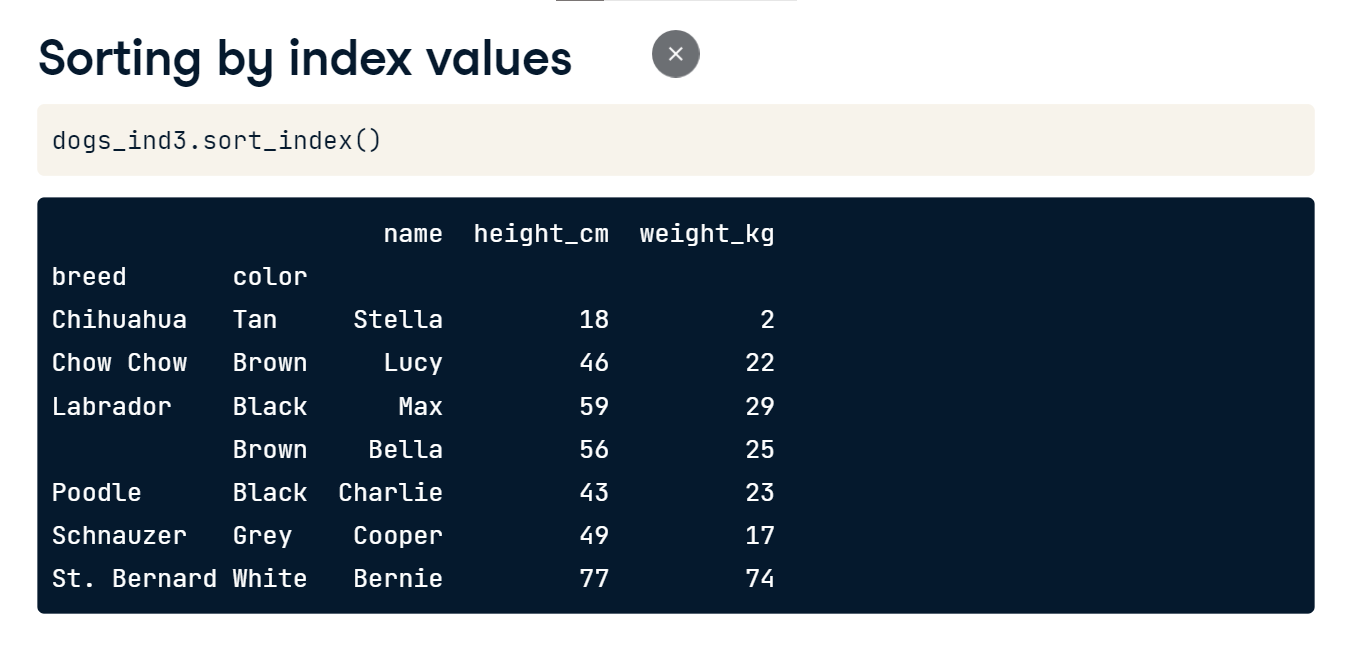
**Subset inner levels with a list of tuples**

To subset on inner levels, you need to pass a list of tuples. Here, the first tuple specifies Labrador at the outer level and Brown at the inner level. The resulting rows have to match all conditions from a tuple. For example, the black Labrador wasn't returned because the brown condition wasn't matched.



**Sorting by index values**

In chapter 1, you saw how to sort the rows of a DataFrame using sort\_values. You can also sort by index values using sort\_index. By default, it sorts all index levels from outer to inner, in ascending order.



**Controlling sort\_index**

You can control the sorting by passing lists to the level and ascending arguments.



**Now you have two problems**

Indexes are controversial. Although they simplify subsetting code, there are some downsides. Index values are just data. Storing data in multiple forms makes it harder to think about. There is a concept called "tidy data," where data is stored in tabular form - like a DataFrame. Each row contains a single observation, and each variable is stored in its own column. Indexes violate the last rule since index values don't get their own column. In pandas, the syntax for working with indexes is different from the syntax for working with columns. By using two syntaxes, your code is more complicated, which can result in more bugs. If you decide you don't want to use indexes, that's perfectly reasonable. However, it's useful to know how they work for cases when you need to read other people's code.

**Temperature dataset**

In this chapter, you'll work with a monthly time series of air temperatures in cities around the world.