Explicit indexes

DATA MANIPULATION WITH PANDAS

Richie Cotton

Learning Solutions Architect at DataCamp

1. 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.

The dog dataset, revisited

```
print(dogs)
```

```
      name        breed  color  height_cm  weight_kg
0    Bella     Labrador  Brown         56         25
1  Charlie       Poodle  Black         43         23
2     Lucy    Chow Chow  Brown         46         22
3   Cooper    Schnauzer   Gray         49         17
4      Max     Labrador  Black         59         29
5   Stella    Chihuahua    Tan         18          2
6   Bernie  St. Bernard  White         77         74
```

.columns and .index

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

```
dogs.columns
```

```
Index(['name', 'breed', 'color', 'height_cm', 'weight_kg'], dtype='object')
```

```
dogs.index
```

```
RangeIndex(start=0, stop=7, step=1)
```

Setting a column as the index

```
dogs_ind = dogs.set_index("name")
print(dogs_ind)
```

```
            breed  color  height_cm  weight_kg
name
Bella    Labrador  Brown         56         25
Charlie    Poodle  Black         43         23
Lucy    Chow Chow  Brown         46         22
Cooper  Schnauzer   Grey         49         17
Max      Labrador  Black         59         29
Stella  Chihuahua    Tan         18          2
Bernie St. Bernard White         77         74
```

4. 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

```
dogs_ind.reset_index()
```

```
      name        breed  color  height_cm  weight_kg
0    Bella     Labrador  Brown         56         25
1  Charlie       Poodle  Black         43         23
2     Lucy    Chow Chow  Brown         46         22
3   Cooper    Schnauzer   Grey         49         17
4      Max     Labrador  Black         59         29
5   Stella    Chihuahua    Tan         18          2
6   Bernie  St. Bernard  White         77         74
```

5. 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

```
dogs_ind.reset_index(drop=True)
```

```
          breed  color  height_cm  weight_kg
0      Labrador  Brown         56         25
1        Poodle  Black         43         23
2     Chow Chow  Brown         46         22
3     Schnauzer   Grey         49         17
4      Labrador  Black         59         29
5     Chihuahua    Tan         18          2
6    St. Bernard  White         77         74
```

6. 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

```
dogs[dogs["name"].isin(["Bella", "Stella"])]
```

```
     name      breed  color  height_cm  weight_kg
0   Bella   Labrador  Brown         56         25
5  Stella  Chihuahua    Tan         18          2
```

```
dogs_ind.loc[["Bella", "Stella"]]
```

```
            breed  color  height_cm  weight_kg
name
Bella    Labrador  Brown         56         25
Stella  Chihuahua    Tan         18          2
```

7. 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

```
dogs_ind2 = dogs.set_index("breed")
print(dogs_ind2)
```

```
                name  color  height_cm  weight_kg
breed
Labrador       Bella  Brown         56         25
Poodle       Charlie  Black         43         23
Chow Chow       Lucy  Brown         46         22
Schnauzer     Cooper   Grey         49         17
Labrador         Max  Black         59         29
Chihuahua     Stella    Tan         18          2
St. Bernard   Bernie  White         77         74
```

Subsetting on duplicated index values

```
dogs_ind2.loc["Labrador"]
```

```
           name  color  height_cm  weight_kg
breed
Labrador  Bella  Brown         56         25
Labrador    Max  Black         59         29
```

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

```
dogs_ind3 = dogs.set_index(["breed", "color"])
print(dogs_ind3)
```

```
                        name  height_cm  weight_kg
breed       color
Labrador    Brown      Bella         56         25
Poodle      Black    Charlie         43         23
Chow Chow   Brown       Lucy         46         22
Schnauzer   Grey      Cooper         49         17
Labrador    Black        Max         59         29
Chihuahua   Tan       Stella         18          2
St. Bernard White     Bernie         77         74
```

Subset the outer level with a list

```
dogs_ind3.loc[["Labrador", "Chihuahua"]]
```

```
                    name  height_cm  weight_kg
breed     color
Labrador  Brown    Bella         56         25
          Black      Max         59         29
Chihuahua Tan     Stella         18          2
```

Subset inner levels with a list of tuples

```
dogs_ind3.loc[[("Labrador", "Brown"), ("Chihuahua", "Tan")]]
```

```
                      name  height_cm  weight_kg
breed     color                                
Labrador  Brown      Bella         56         25
Chihuahua Tan       Stella         18          2
```

Sorting by index values

```
dogs_ind3.sort_index()
```

```
                          name  height_cm  weight_kg
breed        color
Chihuahua    Tan        Stella         18          2
Chow Chow    Brown        Lucy         46         22
Labrador     Black         Max         59         29
             Brown       Bella         56         25
Poodle       Black     Charlie         43         23
Schnauzer    Grey       Cooper         49         17
St. Bernard  White      Bernie         77         74
```

Controlling sort_index

```
dogs_ind3.sort_index(level=["color", "breed"], ascending=[True, False])
```

```
                           name  height_cm  weight_kg
breed       color
Poodle      Black  Charlie         43         23
Labrador    Black      Max         59         29
            Brown    Bella         56         25
Chow Chow   Brown     Lucy         46         22
Schanuzer   Grey    Cooper         49         17
Chihuahua   Tan     Stella         18          2
St. Bernard White   Bernie         77         74
```

Sorting index values is similar to sorting values in columns, exectp that you call .sort_index() instead of .sort_values().

Now you have two problems

- Index values are just data
- Indexes violate "tidy data" principles
- You need to learn two syntaxes

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

	date	city	country	avg_temp_c
0	2000-01-01	Abidjan	Côte D'Ivoire	27.293
1	2000-02-01	Abidjan	Côte D'Ivoire	27.685
2	2000-03-01	Abidjan	Côte D'Ivoire	29.061
3	2000-04-01	Abidjan	Côte D'Ivoire	28.162
4	2000-05-01	Abidjan	Côte D'Ivoire	27.547

Let's practice!

DATA MANIPULATION WITH PANDAS

Slicing and subsetting with .loc and .iloc

DATA MANIPULATION WITH PANDAS

Richie Cotton
Learning Solutions Architect at DataCamp

Slicing lists

Remember that Python positions start from zero, so 2 refers to the third element, Chow Chow.

```
breeds = ["Labrador", "Poodle",
          "Chow Chow", "Schnauzer",
          "Labrador", "Chihuahua",
          "St. Bernard"]
```

```
['Labrador',
 'Poodle',
 'Chow Chow',
 'Schnauzer',
 'Labrador',
 'Chihuahua',
 'St. Bernard']
```

```
breeds[2:5]
```

```
['Chow Chow', 'Schnauzer', 'Labrador']
```

```
breeds[:3]
```

```
['Labrador', 'Poodle', 'Chow Chow']
```

```
breeds[:]
```

Slicing with colon on its own returns the whole list.

```
['Labrador','Poodle','Chow Chow','Schnauzer',
 'Labrador','Chihuahua','St. Bernard']
```

Sort the index before you slice

```
dogs_srt = dogs.set_index(["breed", "color"]).sort_index()
print(dogs_srt)
```

```
                         name  height_cm  weight_kg
breed      color
Chihuahua  Tan         Stella         18          2
Chow Chow  Brown         Lucy         46         22
Labrador   Black          Max         59         29
           Brown        Bella         56         25
Poodle     Black      Charlie         43         23
Schnauzer  Grey        Cooper         49         17
St. Bernard White      Bernie         77         74
```

3. Sort the index before you slice
You can also slice DataFrames, but first, you need to sort the index. Here, the dogs dataset has been given a multi-level index of breed and color; then, the index is sorted with sort_index.

Compared to slicing lists, there are a few things to remember. You can only slice an index if the index is sorted (using .sort_index()).
To slice at the outer level, first and last can be strings.
To slice at inner levels, first and last should be tuples.
If you pass a single slice to .loc[], it will slice the rows.

Slicing the outer index level

```
dogs_srt.loc["Chow Chow":"Poodle"]
```

```
                    name  height_cm  weight_kg
breed     color
Chow Chow Brown     Lucy         46         22
Labrador  Black      Max         59         29
          Brown    Bella         56         25
Poodle    Black  Charlie         43         23
```

The final value `"Poodle"` *is included*

Full dataset

```
                       name  height_cm  weight_kg
breed       color
Chihuahua   Tan      Stella         18          2
Chow Chow   Brown      Lucy         46         22
Labrador    Black       Max         59         29
            Brown     Bella         56         25
Poodle      Black   Charlie         43         23
Schnauzer   Grey     Cooper         49         17
St. Bernard White    Bernie         77         74
```

4. Slicing the outer index level
To slice rows at the outer level of an index, you call loc, passing the first and last values separated by a colon. The full dataset is shown on the right for comparison. There are two differences compared to slicing lists. Rather than specifying row numbers, you specify index values.

Slicing the inner index levels **badly**

```
dogs_srt.loc["Tan":"Grey"]
```

```
Empty DataFrame
Columns: [name, height_cm, weight_kg]
Index: []
```

5. Slicing the inner index levels badly
The same technique doesn't work on inner index levels. Here, trying to slice from Tan to Grey returns an empty DataFrame instead of the six dogs we wanted. It's important to understand the danger here. Pandas doesn't throw an error to let you know that there is a problem, **so be careful when coding.**

Full dataset

```
                          name  height_cm  weight_kg
breed        color
Chihuahua    Tan        Stella         18          2
Chow Chow    Brown        Lucy         46         22
Labrador     Black         Max         59         29
             Brown       Bella         56         25
Poodle       Black     Charlie         43         23
Schnauzer    Grey       Cooper         49         17
St. Bernard  White      Bernie         77         74
```

Slicing the inner index levels correctly

```
dogs_srt.loc[
    ("Labrador", "Brown"):("Schnauzer", "Grey")]
```

```
                       name  height_cm  weight_kg
breed     color
Labrador  Brown       Bella         56         25
Poodle    Black     Charlie         43         23
Schnauzer Grey       Cooper         49         17
```

Full dataset

```
                        name  height_cm  weight_kg
breed        color
Chihuahua    Tan      Stella         18          2
Chow Chow    Brown      Lucy         46         22
Labrador     Black       Max         59         29
             Brown     Bella         56         25
Poodle       Black   Charlie         43         23
Schnauzer    Grey     Cooper         49         17
St. Bernard  White    Bernie         77         74
```

Slicing columns

```
dogs_srt.loc[:, "name":"height_cm"]
```

```
                           name  height_cm
breed        color
Chihuahua    Tan         Stella         18
Chow Chow    Brown         Lucy         46
Labrador     Black          Max         59
             Brown        Bella         56
Poodle       Black      Charlie         43
Schnauzer    Grey        Cooper         49
St. Bernard  White       Bernie         77
```

Full dataset

```
                           name  height_cm  weight_kg
breed        color
Chihuahua    Tan         Stella         18          2
Chow Chow    Brown         Lucy         46         22
Labrador     Black          Max         59         29
             Brown        Bella         56         25
Poodle       Black      Charlie         43         23
Schnauzer    Grey        Cooper         49         17
St. Bernard  White       Bernie         77         74
```

Slice twice

Bidirectional Slicing

```
dogs_srt.loc[
    ("Labrador", "Brown"):("Schnauzer", "Grey"),
    "name":"height_cm"]
```

```
                      name  height_cm
breed     color
Labrador  Brown      Bella         56
Poodle    Black    Charlie         43
Schnauzer Grey      Cooper         49
```

Full dataset

```
                        name  height_cm  weight_kg
breed       color
Chihuahua   Tan       Stella         18          2
Chow Chow   Brown       Lucy         46         22
Labrador    Black        Max         59         29
            Brown      Bella         56         25
Poodle      Black    Charlie         43         23
Schnauzer   Grey      Cooper         49         17
St. Bernard White     Bernie         77         74
```

Dog days

```
dogs = dogs.set_index("date_of_birth").sort_index()
print(dogs)
```

```
                  name        breed  color  height_cm  weight_kg
date_of_birth
2011-12-11      Cooper    Schnauzer   Grey         49         17
2013-07-01       Bella     Labrador  Brown         56         25
2014-08-25        Lucy    Chow Chow  Brown         46         22
2015-04-20      Stella    Chihuahua    Tan         18          2
2016-09-16     Charlie       Poodle  Black         43         23
2017-01-20         Max     Labrador  Black         59         29
2018-02-27      Bernie  St. Bernard  White         77         74
```

Slicing by dates

```
# Get dogs with date_of_birth between 2014-08-25 and 2016-09-16
dogs.loc["2014-08-25":"2016-09-16"]
```

```
                  name      breed  color  height_cm  weight_kg
date_of_birth
2014-08-25        Lucy  Chow Chow  Brown         46         22
2015-04-20      Stella  Chihuahua    Tan         18          2
2016-09-16     Charlie     Poodle  Black         43         23
```

Slicing by partial dates

```
# Get dogs with date_of_birth between 2014-01-01 and 2016-12-31
dogs.loc["2014":"2016"]
```

```
                  name      breed  color  height_cm  weight_kg
date_of_birth
2014-08-25        Lucy  Chow Chow  Brown         46         22
2015-04-20      Stella  Chihuahua    Tan         18          2
2016-09-16     Charlie     Poodle  Black         43         23
```

Subsetting by row/column number

```
print(dogs.iloc[2:5, 1:4])
```

```
       breed  color  height_cm
2  Chow Chow  Brown         46
3  Schnauzer   Grey         49
4   Labrador  Black         59
```

Full dataset

```
      name        breed  color  height_cm  weight_kg
0    Bella     Labrador  Brown         56         25
1  Charlie       Poodle  Black         43         23
2     Lucy    Chow Chow  Brown         46         22
3   Cooper    Schnauzer   Grey         49         17
4      Max     Labrador  Black         59         29
5   Stella    Chihuahua    Tan         18          2
6   Bernie  St. Bernard  White         77         74
```

Let's practice!

DATA MANIPULATION WITH PANDAS

Working with pivot tables

DATA MANIPULATION WITH PANDAS

Richie Cotton

Learning Solutions Architect at DataCamp

A bigger dog dataset

```
print(dog_pack)
```

```
          breed  color  height_cm  weight_kg
0         Boxer  Brown      62.64       30.4
1        Poodle  Black      46.41       20.4
2        Beagle  Brown      36.39       12.4
3     Chihuahua    Tan      19.70        1.6
4      Labrador    Tan      54.44       36.1
..          ...    ...        ...        ...
87        Boxer   Gray      58.13       29.9
88  St. Bernard  White      70.13       69.4
89       Poodle   Gray      51.30       20.4
90       Beagle  White      38.81        8.8
91       Beagle  Black      33.40       13.5
```

Pivoting the dog pack

```
dogs_height_by_breed_vs_color = dog_pack.pivot_table(
        "height_cm", index="breed", columns="color")
print(dogs_height_by_breed_vs_color)
```

```
color            Black   Brown       Gray        Tan      White
breed
Beagle       34.500000  36.4500  36.313333  35.740000  38.810000
Boxer        57.203333  62.6400  58.280000  62.310000  56.360000
Chihuahua    18.555000      NaN  21.660000  20.096667  17.933333
Chow Chow    51.262500  50.4800        NaN  53.497500  54.413333
Dachshund    21.186667  19.7250        NaN  19.375000  20.660000
Labrador     57.125000      NaN        NaN  55.190000  55.310000
Poodle       48.036000  57.1300  56.645000        NaN  44.740000
St. Bernard  63.920000  65.8825  67.640000  68.334000  67.495000
```

3. Pivoting the dog pack
Recall that you create a pivot table by calling dot-pivot_table. The first argument is the column name containing values to aggregate. The index argument lists the columns to group by and display in rows, and the columns argument lists the columns to group by and display in columns. We'll use the default aggregation function, which is mean.

.loc[] + slicing is a power combo

```
dogs_height_by_breed_vs_color.loc["Chow Chow":"Poodle"]
```

```
color          Black   Brown    Gray      Tan      White
breed
Chow Chow  51.262500  50.480     NaN  53.4975  54.413333
Dachshund  21.186667  19.725     NaN  19.3750  20.660000
Labrador   57.125000     NaN     NaN  55.1900  55.310000
Poodle     48.036000  57.130  56.645      NaN  44.740000
```

4. .loc[] + slicing is a power combo
Pivot tables are just DataFrames with sorted indexes. In particular, the loc and slicing combination is ideal for subsetting pivot tables, like so.

1 _

The axis argument

```
dogs_height_by_breed_vs_color.mean(axis="index")
```

```
color
Black     43.973563
Brown     48.717917
Gray      48.107667
Tan       44.934738
White     44.465208
dtype: float64
```

5. The axis argument
The methods for calculating summary statistics on a DataFrame, such as mean, have an axis argument. The default value is "index," which means "calculate the statistic across rows." Here, the mean is calculated for each color. That is, "across the breeds." The behavior is the same as if you hadn't specified the axis argument.

Calculating summary stats across columns

```
dogs_height_by_breed_vs_color.mean(axis="columns")
```

```
breed
Beagle          36.362667
Boxer           59.358667
Chihuahua       19.561250
Chow Chow       52.413333
Dachshund       20.236667
Labrador        55.875000
Poodle          51.637750
St. Bernard     66.654300
dtype: float64
```

6. Calculating summary stats across columns
To calculate a summary statistic for each row, that is, "across the columns," you set axis to "columns." Here, the mean height is calculated for each breed. That is, "across the colors." For most DataFrames, setting the axis argument doesn't make any sense, since you'll have different data types in each column. Pivot tables are a special case since every column contains the same data type.

Let's practice!

DATA MANIPULATION WITH PANDAS