Pandas - 3

Explicit Indexes

- .columns contains the column index object and .index contains the row index object

Setting a column as the index

- **.set_index(col_name) - The set_index method is used to the index to be a new column.**
- **The index is left aligned whereas all the other columns are right aligned.**

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
```

reset_index

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
```

When drop is set to True, we drop the name column which was the index

previously from the dataset

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
```

When we set indexed it is easy to get the subset of the columns, with the use of the loc method

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
```

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
```

There are 2 dogs named Labrador
so when we try to access the row with the index labrador we get both of the columns.

Setting Multiple index

We pass in the index columns as a list to the set_index method, wherein the indexing is nesting.

```
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
```

To get the subset of outer level index, we can specify the outer level indexes

```
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
```

To get the inner level index subset, we need to pass in a tuple. The resulting rows need to be a perfect match of the passed in 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 Indexes using sort_index

```
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
```

We can control the sorting by passing in the levels

```
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
```

Index values are just data.
Indexes violates “tidy data” principles

The killer feature for indexes is .loc[]: a subsetting method that accepts index values. When you pass it a single argument, it will take a subset of rows.
The code for subsetting using .loc[] can be easier to read than standard square bracket subsetting, which can make your code less burdensome to maintain.

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