COMS BC1016

Introduction to Computational Thinking and Data Science

Lecture 9: Review and Conditionals

Sept. 3, 2025

February 18, 2026

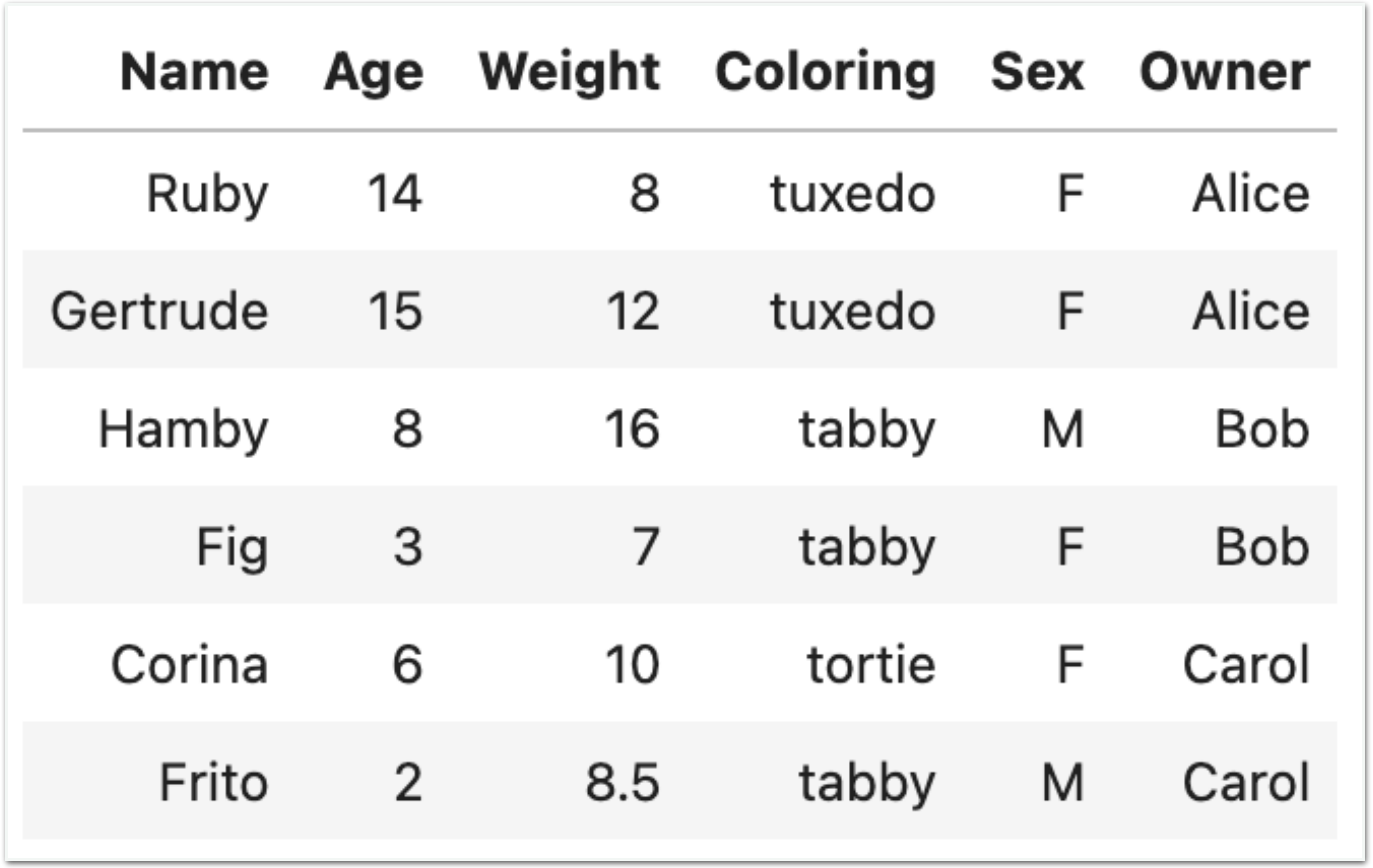
Logistics

- HW 2 is due today at 11:59pm
  - Please remember to submit it as a **.ipynb** file
  - If you are having trouble, please come to one of the office hours (mine, TA, or computing fellow)

Review: Groups and Pivot Tables

Groups and Pivots

Two ways of **summarizing** table data by categorical variables



Name	Age	Weight	Coloring	Sex	Owner
Ruby	14	8	tuxedo	F	Alice
Gertrude	15	12	tuxedo	F	Alice
Hamby	8	16	tabby	M	Bob
Fig	3	7	tabby	F	Bob
Corina	6	10	tortie	F	Carol
Frito	2	8.5	tabby	M	Carol

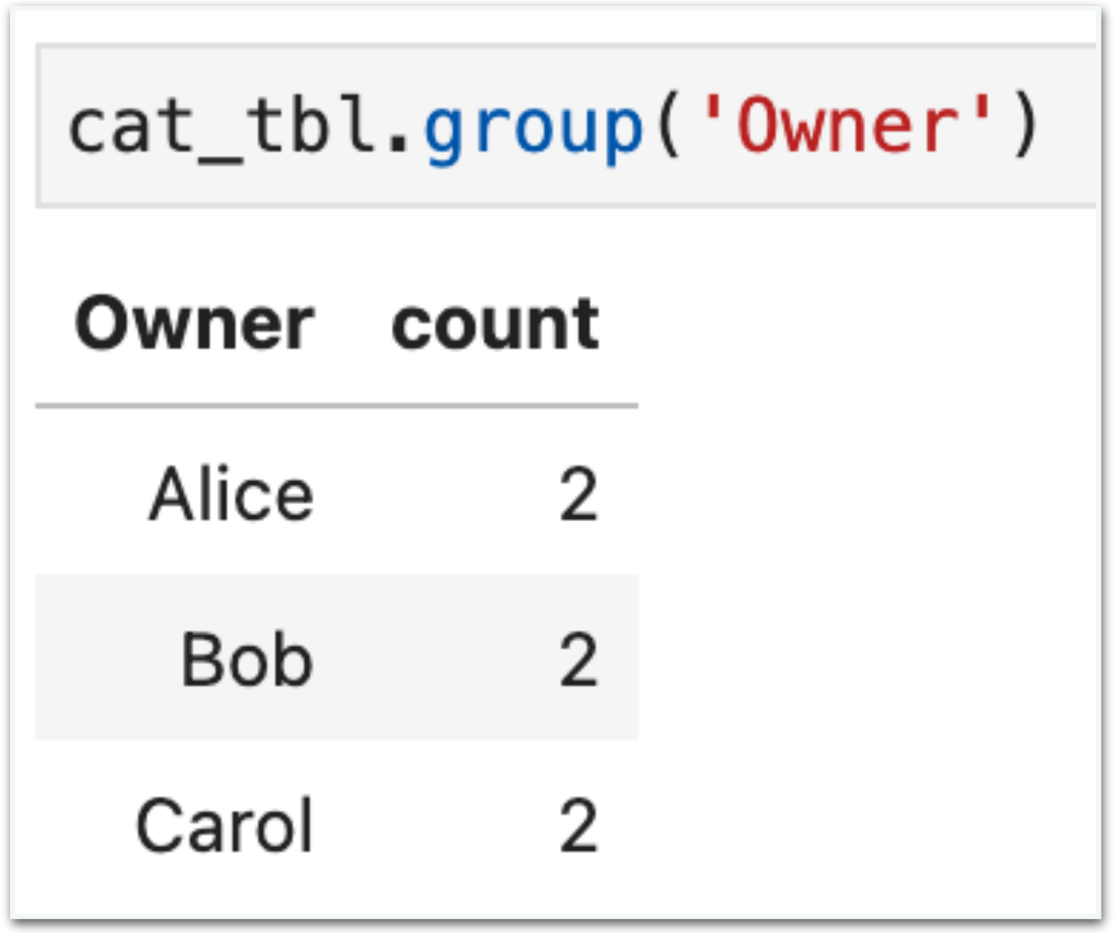
Grouping by a Single Column

The `group` method aggregates all rows with the same value in column `c`

- `tbl.group(c)`
- `tbl.group(c, func)`

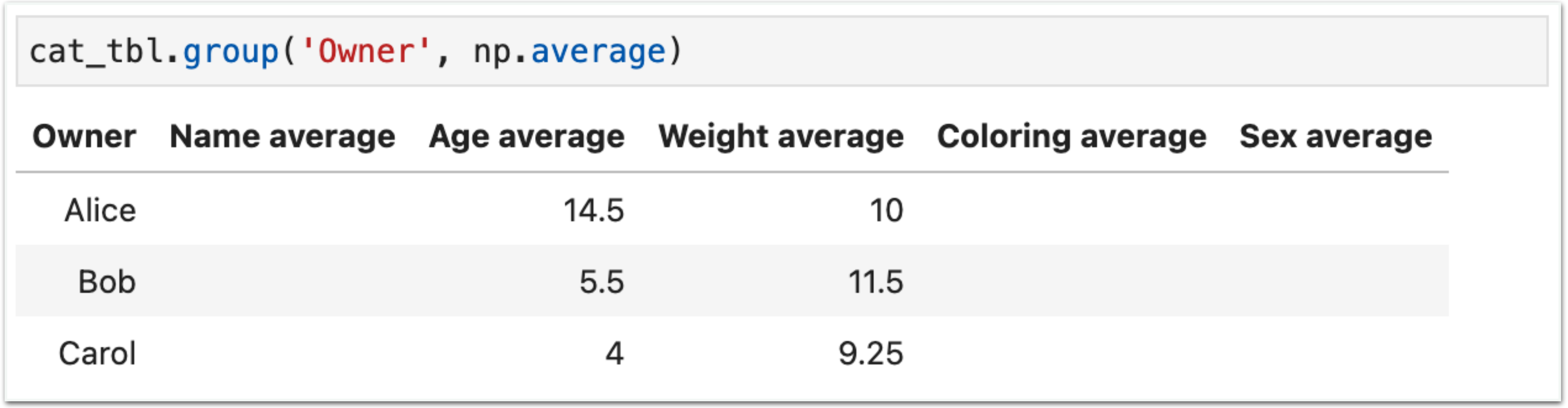
`group` can optionally apply `func` to grouped values, for example:

- `len`: count of grouped values (default)
- `list`: list of all grouped values
- `sum`: total of all grouped values



```
cat_tbl.group('Owner')
```

Owner	count
Alice	2
Bob	2
Carol	2

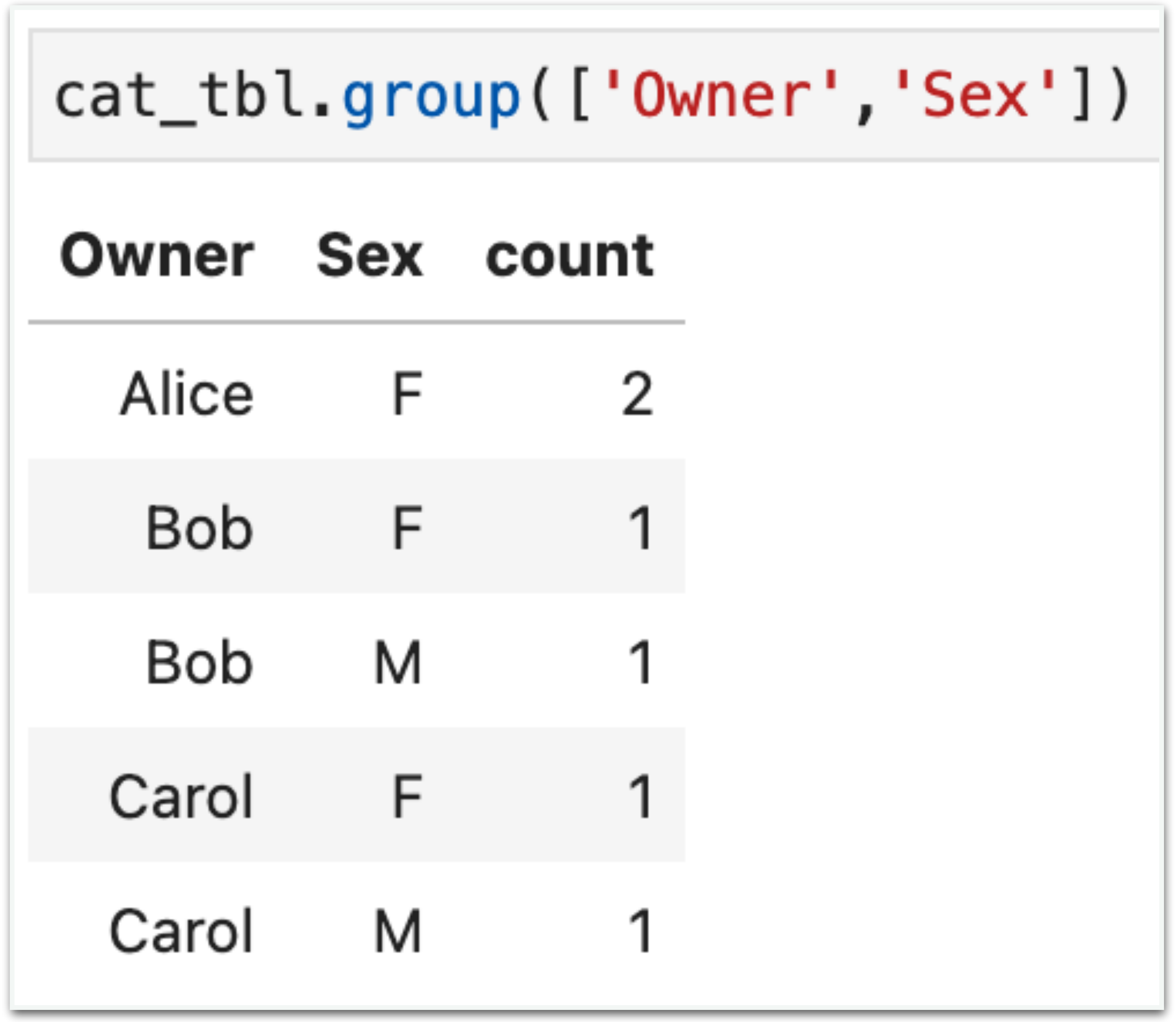


```
cat_tbl.group('Owner', np.average)
```

Owner	Name average	Age average	Weight average	Coloring average	Sex average
Alice		14.5	10		
Bob		5.5	11.5		
Carol		4	9.25		

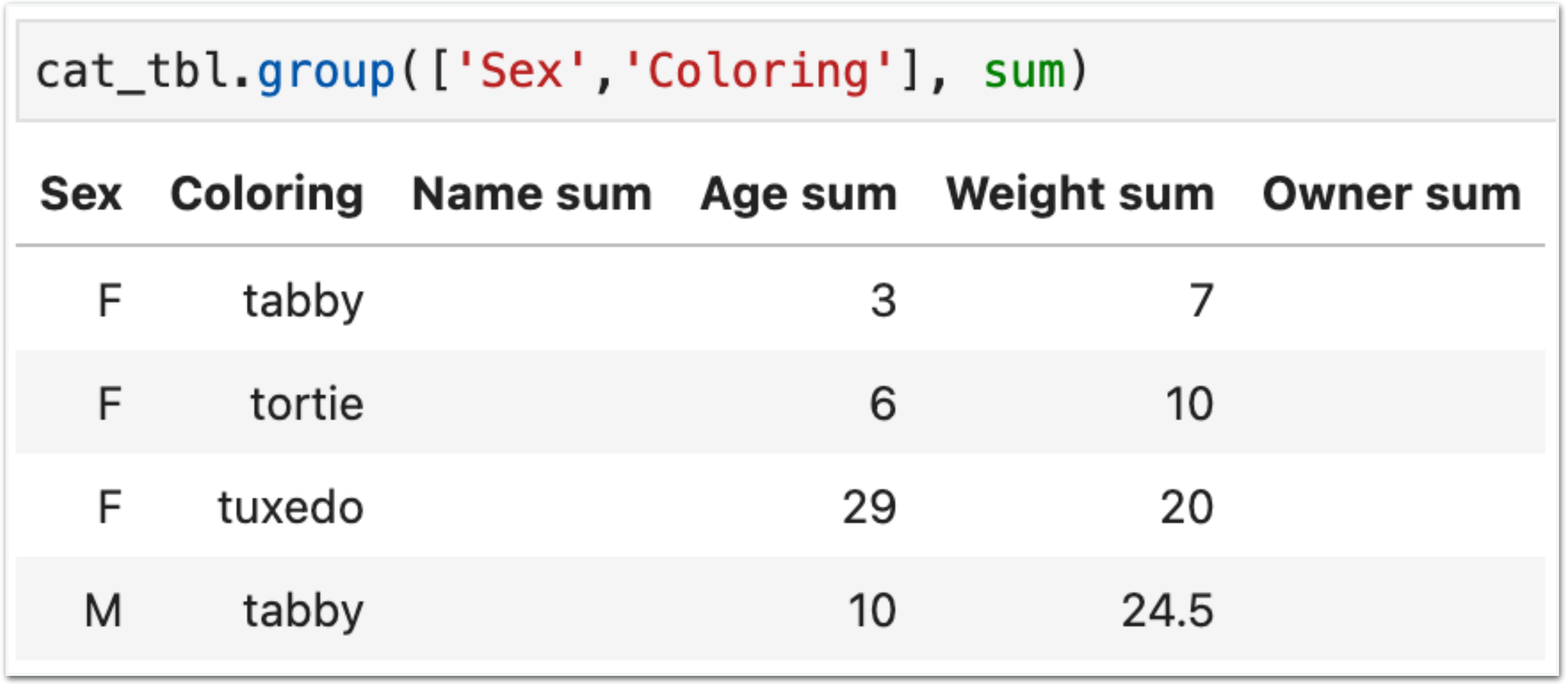
Grouping by Multiple Columns

The `group` method can also aggregate all rows that *share the combination of values* from multiple columns



```
cat_tbl.group(['Owner','Sex'])
```

Owner	Sex	count
Alice	F	2
Bob	F	1
Bob	M	1
Carol	F	1
Carol	M	1



```
cat_tbl.group(['Sex','Coloring'], sum)
```

Sex	Coloring	Name sum	Age sum	Weight sum	Owner sum
F	tabby		3	7	
F	tortie		6	10	
F	tuxedo		29	20	
M	tabby		10	24.5	

Pivot Tables

Cross-classifies according to *two* categorical variables

- Produces a grid of all possible combinations of the two categorical variables
- Grid entries are either counts or aggregated values

Create a pivot table where entries are counts:

```
tbl.pivot(col_var, row_var)
```

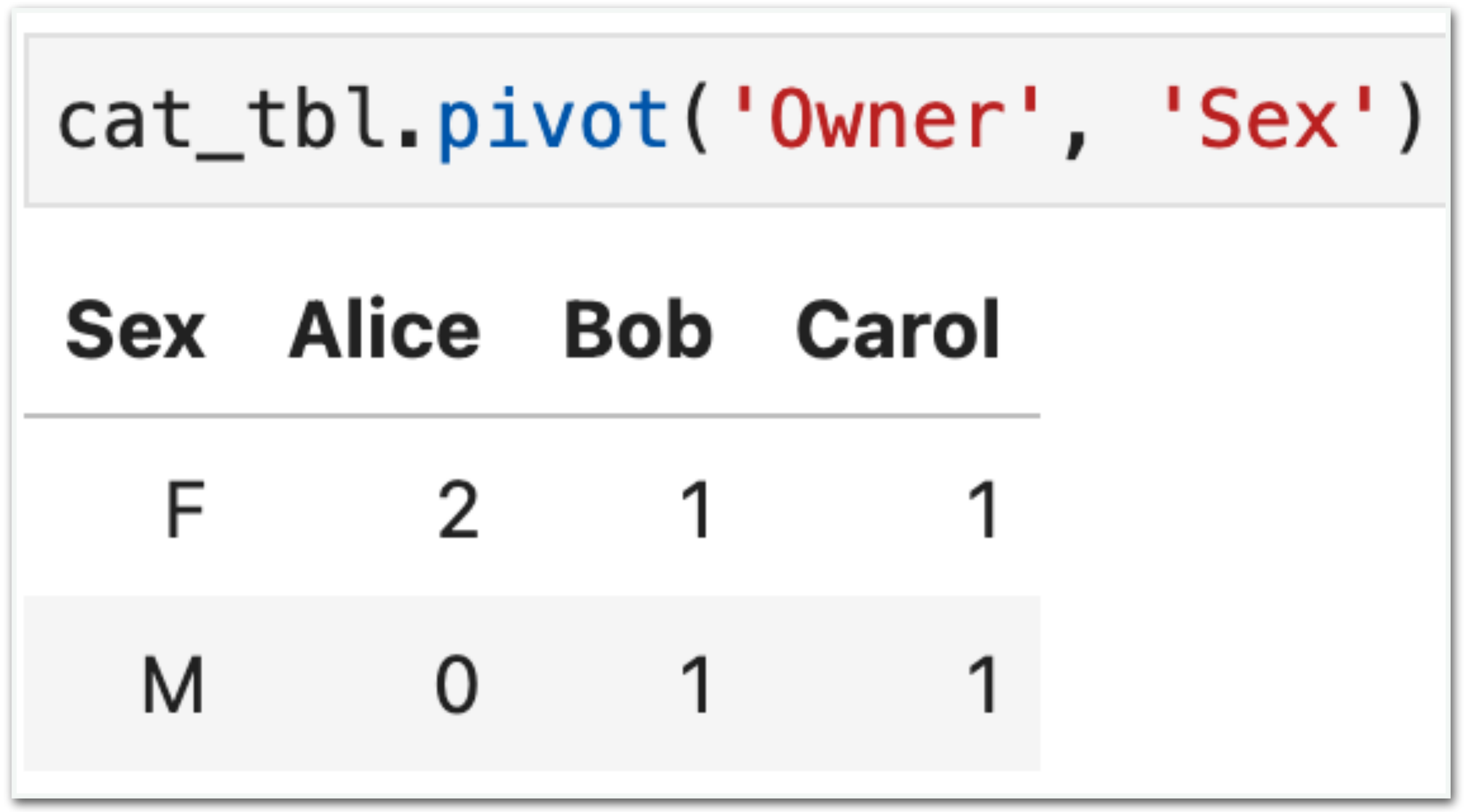
Create a pivot table where entries are aggregated according function `collect` on values in column `values`

```
tbl.pivot(col_var, row_var, values, collect)
```

Pivot Tables

```
tbl.pivot(col_var, row_var)
```

- `col_var`: Variable that forms column labels of grid
- `row_var`: Variable that forms row labels of grid



```
cat_tbl.pivot('Owner', 'Sex')
```

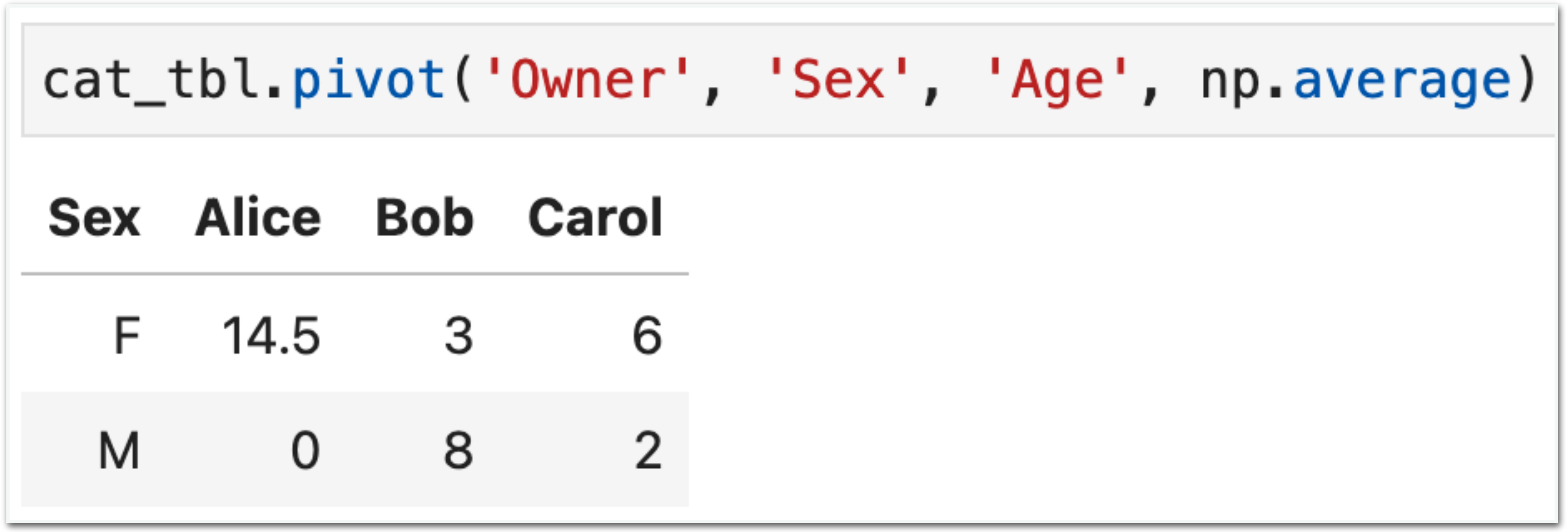
Sex	Alice	Bob	Carol
F	2	1	1
M	0	1	1

Pivot Tables

```
tbl.pivot(col_var, row_var, values, collect)
```

- `values`: Table column to aggregate
- `collect`: Function to aggregate with

Either include **both** `values` and `collect` or **neither**



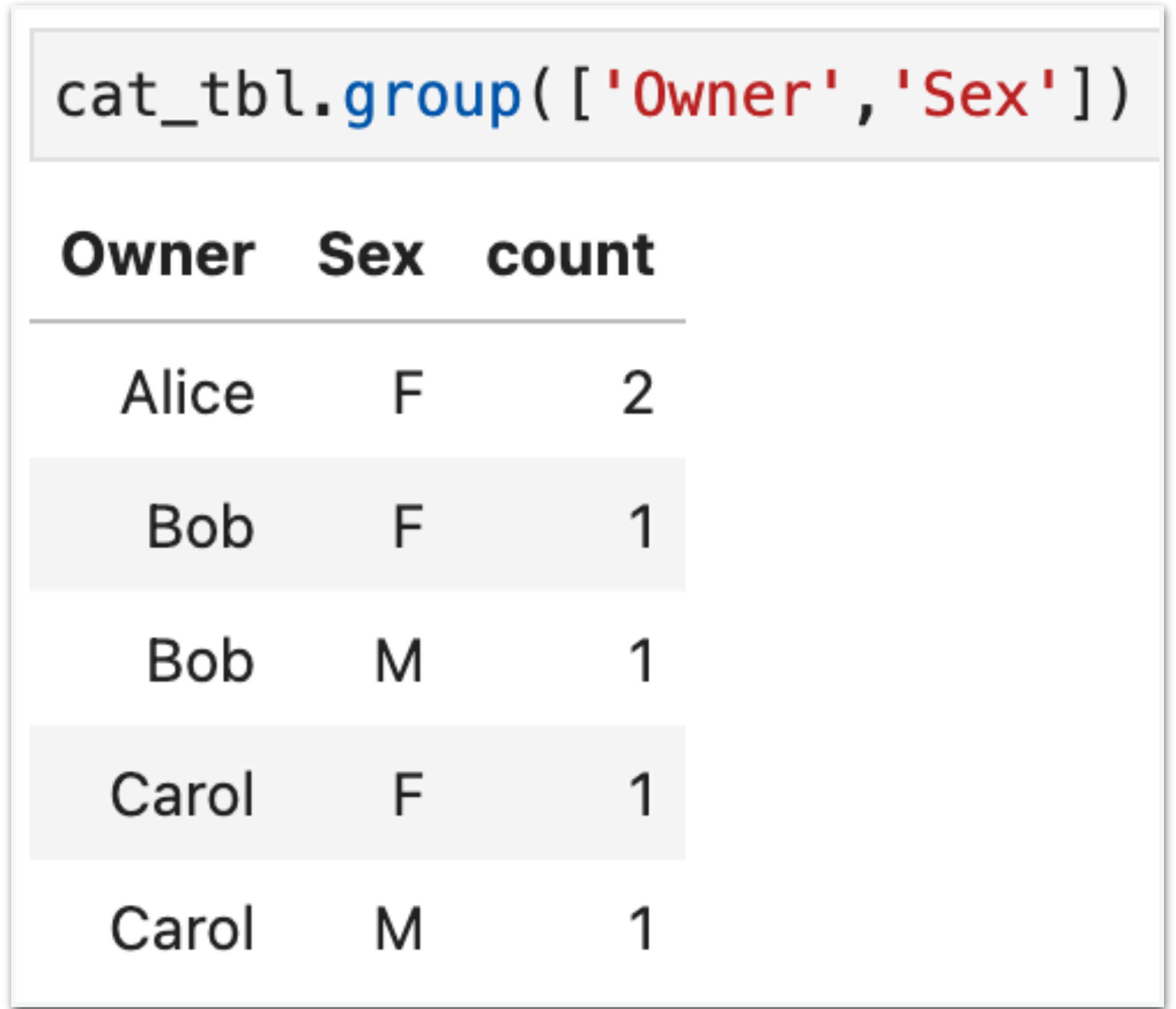
```
cat_tbl.pivot('Owner', 'Sex', 'Age', np.average)
```

Sex	Alice	Bob	Carol
F	14.5	3	6
M	0	8	2

Group vs Pivot

Group

- One combo of grouping variables **per row**
- **Any number** of grouping variables
- Aggregate values of **all other columns** in the table
- Missing combos are **absent**

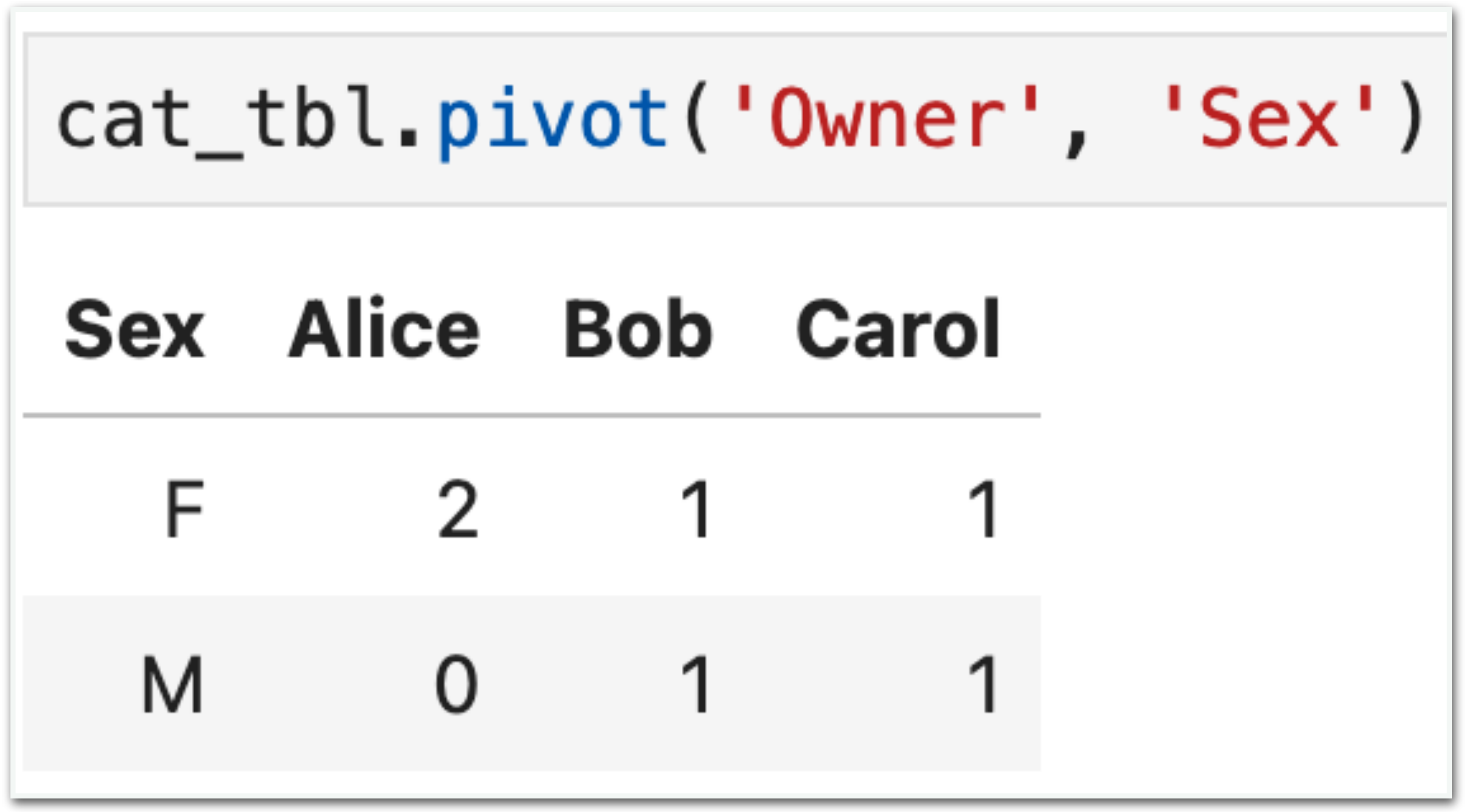


```
cat_tbl.group(['Owner','Sex'])
```

Owner	Sex	count
Alice	F	2
Bob	F	1
Bob	M	1
Carol	F	1
Carol	M	1

Pivot

- One combo of grouping variables **per entry**
- **Two** grouping variables: columns and rows
- Aggregate values of **values column**
- Missing combos **= 0 (or empty string)**



```
cat_tbl.pivot('Owner', 'Sex')
```

Sex	Alice	Bob	Carol
F	2	1	1
M	0	1	1

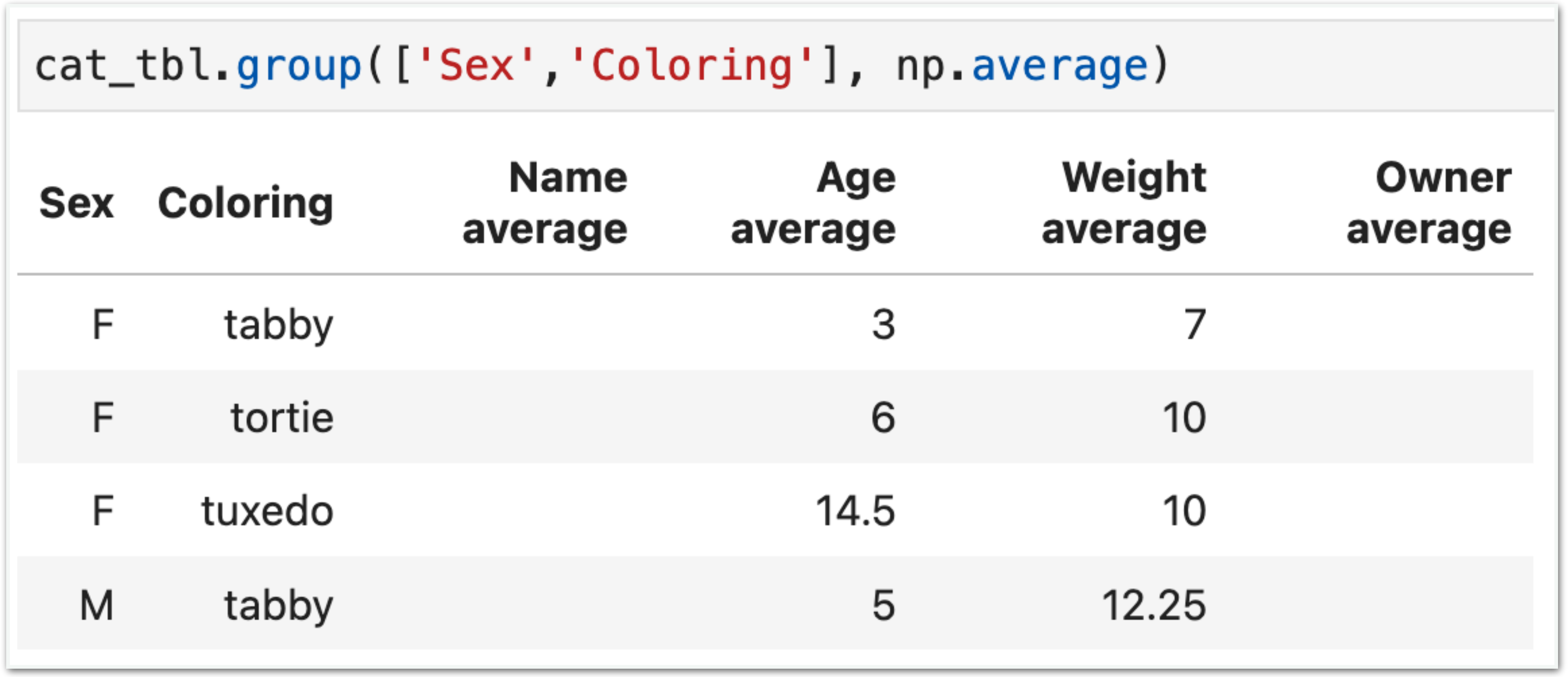
Group vs Pivot

Group

- One combo of grouping variables **per row**
- **Any number** of grouping variables
- Aggregate values of **all other columns** in the table
- Missing combos are **absent**

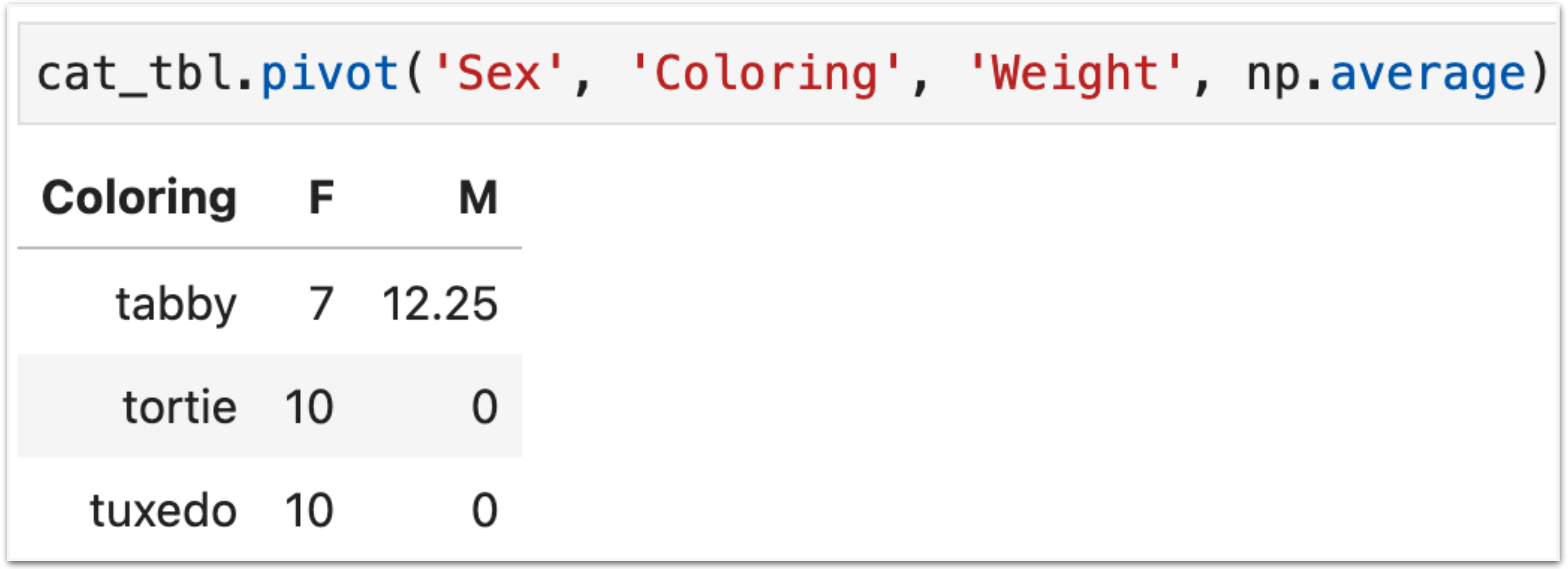
Pivot

- One combo of grouping variables **per entry**
- **Two** grouping variables: columns and rows
- Aggregate values of **values column**
- Missing combos **= 0 (or empty string)**



```
cat_tbl.group(['Sex','Coloring'], np.average)
```

Sex	Coloring	Name average	Age average	Weight average	Owner average
F	tabby		3	7	
F	tortie		6	10	
F	tuxedo		14.5	10	
M	tabby		5	12.25	



```
cat_tbl.pivot('Sex', 'Coloring', 'Weight', np.average)
```

Coloring	F	M
tabby	7	12.25
tortie	10	0
tuxedo	10	0

Control Statements

Control Statements

Control Statements modify *if* and/or *how many times* a block of code is executed in a program

Control Statements

- Two major types are `if` and `for`
  - `if` statements specify code that should be run conditioned on something being true
    - They can also specify if alternative code should be run otherwise
  - `for` loops allow executing code over each element in some sequence of items

`if` statements

- Conditionals begin with an `if` followed by a boolean statement
  - Runs code based on whether a boolean statement evaluates to `True`
- Conditionals can include a combination of `if`, `elif`, and `else` clauses
  - Maximum of one `if` and one `else`

if statements

```
if statement_1:
    first_code_block
```

Runs if statement_1 == True

if statements

```
if statement_1:
    first_code_block
else:
    second_code_block
```

Runs if statement_1 == True

nothing above == True

if statements

Shorthand for "else if"

```
if statement_1:
    first_code_block
elif statement_2:
    second_code_block
else:
    third_code_block
```

Runs if statement_1 == True

Runs if statement_1 != True AND statement_2 == True

nothing above == True

if statements

Shorthand for "else if"

```
if statement_1:
    first_code_block
elif statement_2:
    second_code_block
elif statement_3:
    third_code_block
else:
    fourth_code_block
```

Runs if statement_1 == True

Runs if statement_1 != True
AND statement_2 == True

statement_1 != True
AND statement_2 != True
AND statement_3 == True

nothing above == True

Booleans and Comparisons

Boolean Data Type

- Booleans are data types for truth values: **`True`** or **`False`**
  - **`True`** is equivalent to `1`
  - **`False`** is equivalent to `0`
- `bool(x)` turns `x` into a boolean
  - e.g., `bool(1)` evaluates to **`True`** and `bool(0)` evaluates to **`False`**

Comparison Operators

Operation	Meaning
`>`	greater than
`>=`	greater than or equal to
`<`	less than
`<=`	less than or equal to
`==`	equal to
`!=`	not equal to

Comparison Operators

Example	Result	Explanation
`3 > 2`	**`True`**	3 is greater than 2
`3 > 3`	**`False`**	3 is not (*strictly*) *greater than* 3
`4 <= 4`	**`True`**	4 is less than or equal to 4

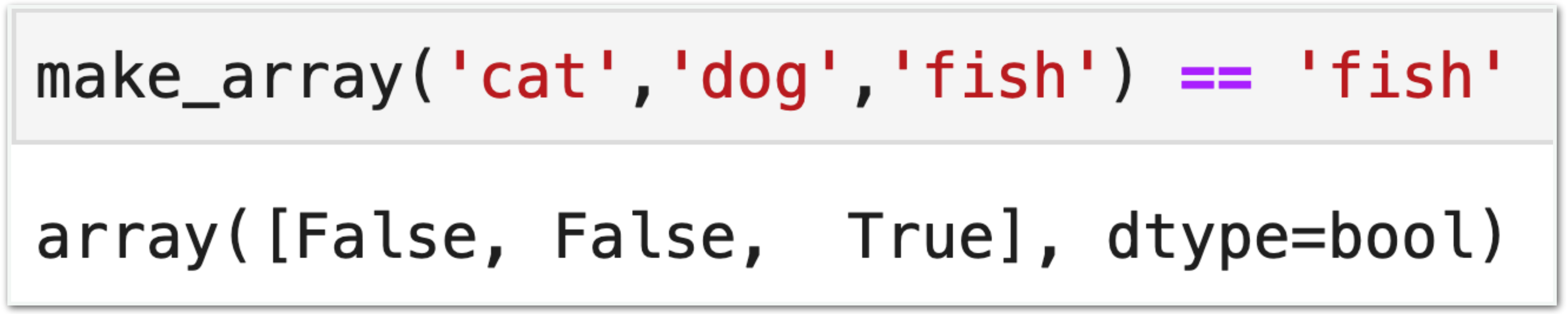
Comparison Operators

Example	Result	Explanation
`'4' == 4`	**`False`**	`'4'` is a string and `4` is an int
`3 - 2 == 4 - 3`	**`True`**	`3-2` equals `1` and `4-3` equals `1`; `1` equals `1`
`2 != 2`	**`False`**	`2` is not *not* equal to `2`

Comparisons with Arrays

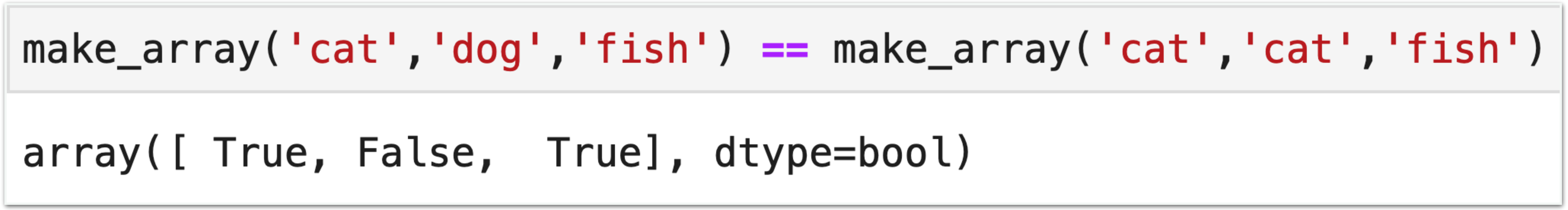
- Single values can be compared against each element in an array

- Comparing two arrays will compare element-by-element



```
make_array('cat','dog','fish') == 'fish'
```

```
array([False, False,  True], dtype=bool)
```



```
make_array('cat','dog','fish') == make_array('cat','cat','fish')
```

```
array([ True, False,  True], dtype=bool)
```

`and`, `or`, and `not`

- You can combine conditional statements using `and` & `or`
  - `and` will return `True` if **all** expressions are `True` (and `False` otherwise)
  - `or` will return `True` if **any** expressions is `True` (and `False` otherwise)
- You can negate a boolean value using `not`
  - `not True` will evaluate to `False`
  - `not False` will evaluate to `True`

`and`, `or`, and `not`

Example	Result
`True and True`	`True`
`True and False`	`False`
`True or False`	`True`
`False or False`	`False`
`not False`	`True`

Aggregating Comparisons

- Summing an array or list of bool values will count the `True` values only

Example	Result
`True + False + True`	2
`1 + 0 + 1`	2
`sum([True, False, True])`	2

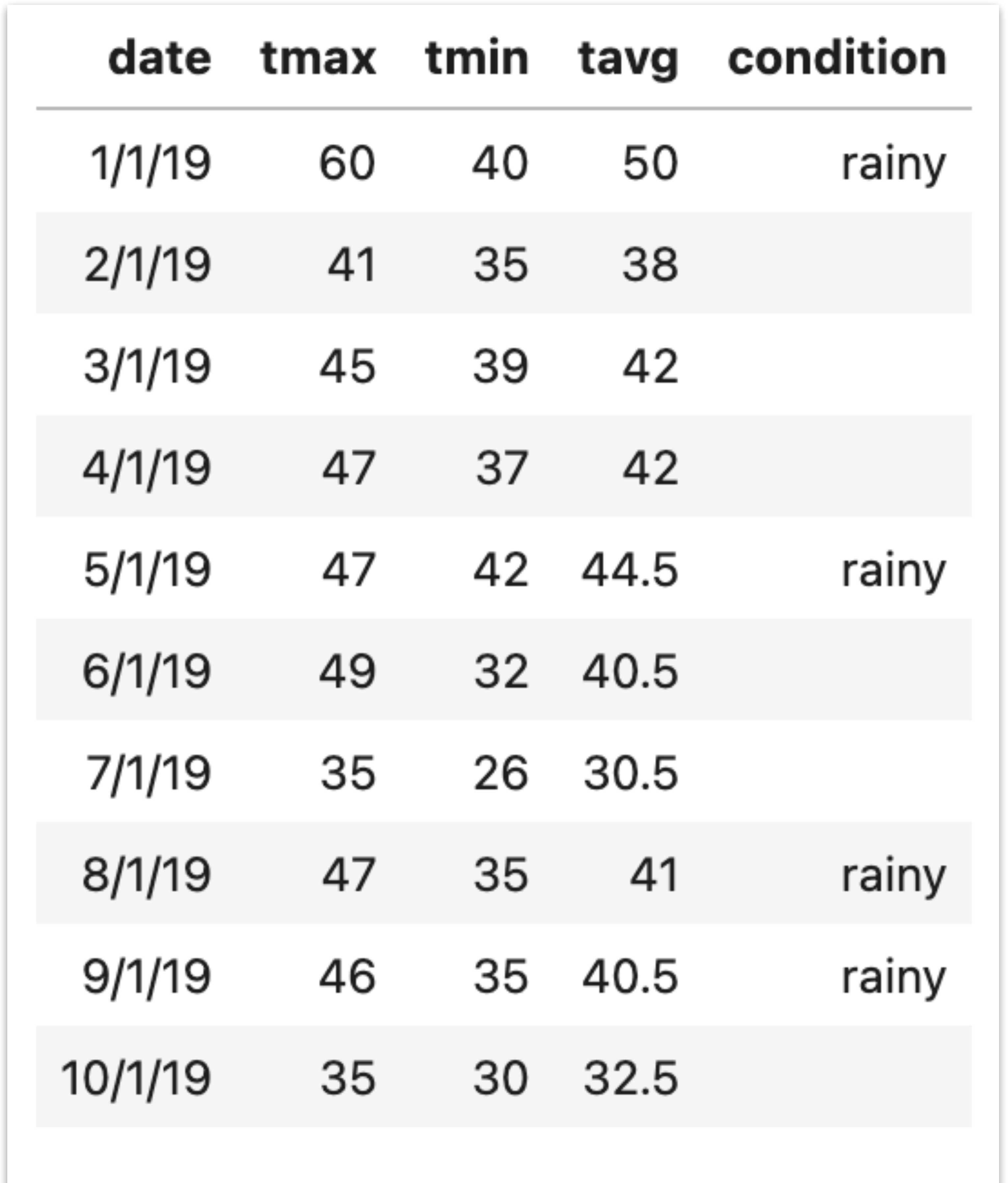
Review: Charts

Chart Selection Exercise

We have NYC weather data from 2019 as shown below (from Kaggle)

Which type of chart (line, scatter, bar, histogram) would best help you answer to each question?

- Do days with hotter highs also tend to have hotter lows?
- How do the number of rainy days compare with the number of snowy days?
- What percent of days have a high of at least 75 degrees?



date	tmax	tmin	tavg	condition
1/1/19	60	40	50	rainy
2/1/19	41	35	38	
3/1/19	45	39	42	
4/1/19	47	37	42	
5/1/19	47	42	44.5	rainy
6/1/19	49	32	40.5	
7/1/19	35	26	30.5	
8/1/19	47	35	41	rainy
9/1/19	46	35	40.5	rainy
10/1/19	35	30	32.5	

Next Class

- Today
  - Group Review
  - Conditionals
- Monday
  - Iteration