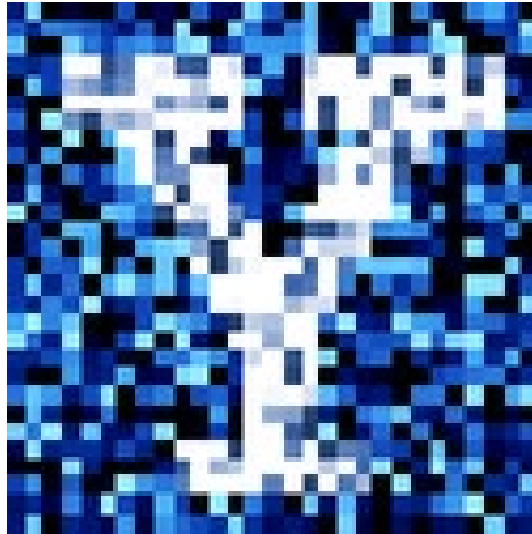


# YData: Introduction to Data Science



Lecture 12: Maps and conditional statements

# Overview

Review of joining tables

Maps

Comparisons

If there is time: Conditional statements

# Announcements

## Homework 4 has been posted

- Due Sunday February 27<sup>th</sup> at 11pm
- Relatively short

## Project 1 has been posted

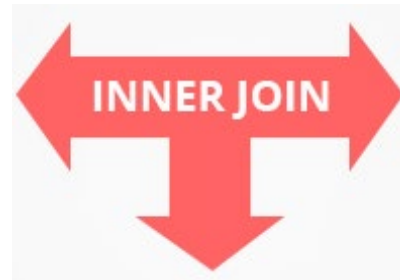
- Due Friday March 4<sup>th</sup>
- Fairly long, start on this soon!!!
- You are allowed to work with one other person on the project
  - You can not discuss the project with anyone else part from the TAs
- Recommendations:
  - Stay organized! There several tables involved, so writing down column names, and steps to solve particular problems on paper prior to coding will make your life much easier.
  - A pdf with functions/methods discussed in class is on the [class Canvas site](#).

# Review of joins

# Review: joining tables

prices

Products	Price
Kiwis	\$6
Onions	\$3
Tomatoes	\$7



quantities

Products	Quantity
Kiwis	10
Onions	6
Broccoli	5

Products	Price	Quantity
Kiwis	\$6	10
Onions	\$3	6

```
prices.join("Products", quantities)
```

Let's explore this in Jupyter!

Maps

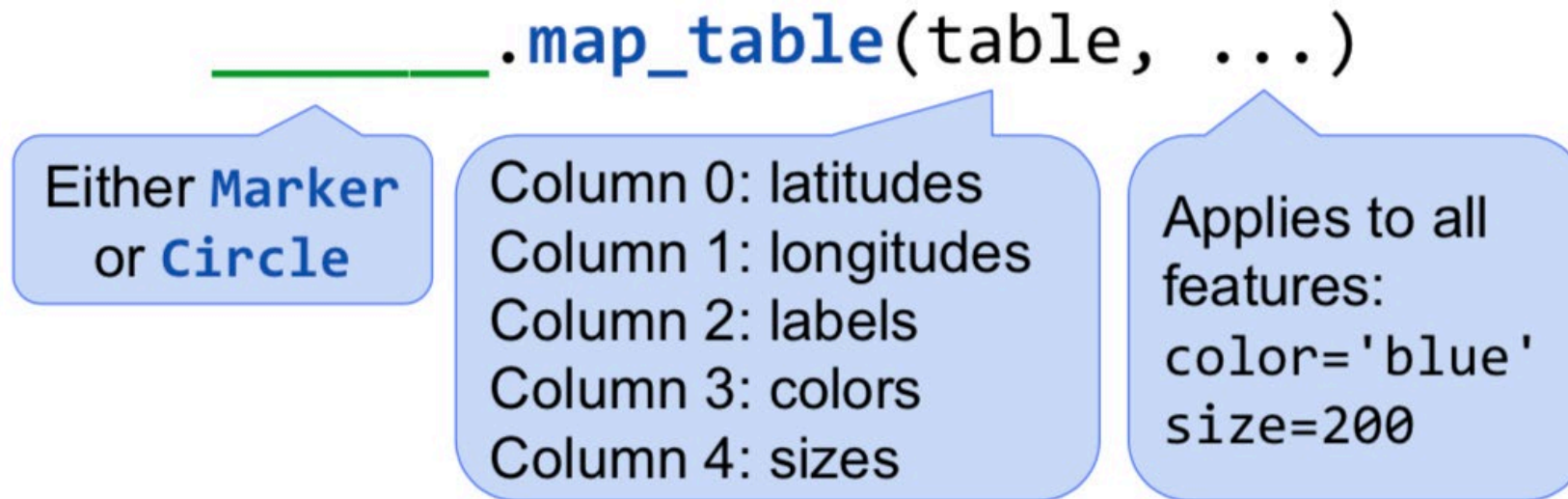
# Maps

Visualizing data on a map can be a powerful way to see spatial trends

- Have we seen any examples of this in YData?

# Maps in the datascience package

We can create maps using the datascience package using a table containing columns of latitude and longitude values



Let's explore this in Jupyter!



# Important Table methods

`t.select(column, ...)` or `t.drop(column, ...)`

`t.take([row, ...])`, or `t.exclude([row, ...])`

`t.sort(column, descending=False, distinct=False)`

`t.where(column, are.condition(...))`

`t.apply(function, column, ...)`

`t.group(column)` or `t.group(column, function)`

`t.group([column, ...])` or `t.group([column, ...], function)`

`t.pivot(cols, rows)` or `t.pivot(cols, rows, vals, function)`

`t.join(column, other table, other table column)`

More documentation can be found at: <http://data8.org/datascience/tables.html>

# Comparisons

# Comparisons

We can use mathematical operators to compare numbers and strings

- Results return Boolean values **True** and **False**

Comparison	Operator	True example	False Example
Less than	<	2 < 3	2 < 2
Greater than	>	3 > 2	3 > 3
Less than or equal	<=	2 <= 2	3 <= 2
Greater or equal	>=	3 >= 3	2 >= 3
Equal	==	3 == 3	3 == 2
Not equal	!=	3 != 2	2 != 2

# Comparisons

Note: when comparing whether two items are equal with use a double equal sign ==

- A single equal sign is used for assigning a value to a name

**x = 2    y = 3** Assignment statements

**x > 1    x > y    y >= 3**  
**x == y    x != 2    2 < x < 5** Comparison expressions

Let's explore this in Jupyter!

# Conditional statements

# Conditional statements

Conditional statements control the sequence of computations that are performed in a program

We use keywords `if` to begin a conditional statement to only execute lines of code if a particular condition is met.

We can use `elif` to test additional conditions

We can use an `else` statement to run code if none of the `if` or `elif` conditions have been met.

```
num = 5
if num == 1:
    print("Monday")
elif num == 2:
    print("Tuesday")
elif num == 3:
    print("Wednesday")
elif num == 4:
    print("Thursday")
elif num == 5:
    print("Friday")
elif num == 6:
    print("Saturday")
elif num == 7:
    print("Sunday")
else:
    print("Invalid input")
```

Let's explore this in Jupyter!