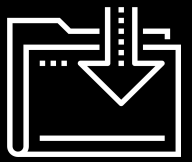
Merging and Data Clean Project

Data Boot Camp

Lesson 4.3



Class Objectives

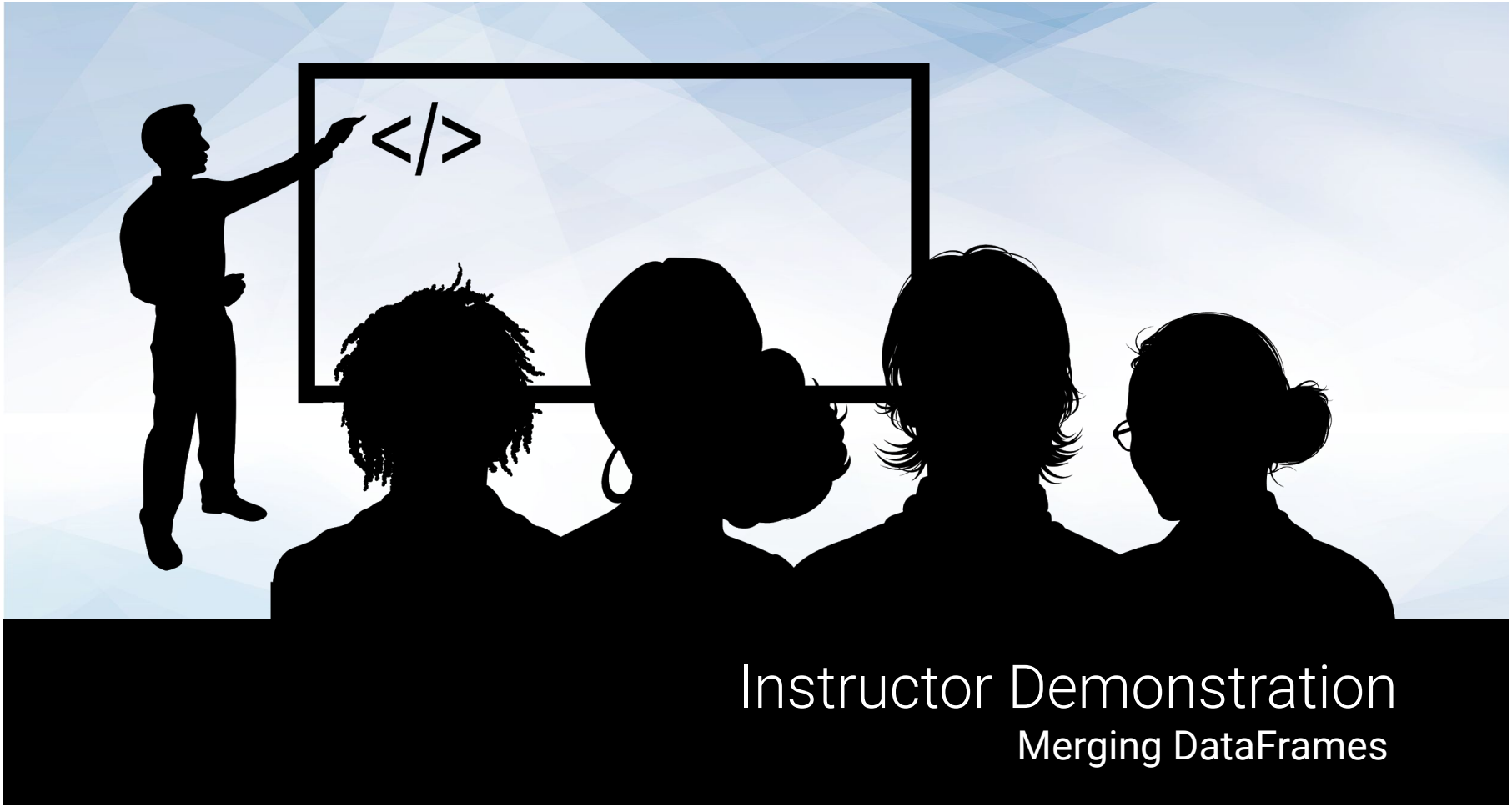
By the end of today's class you will be able to:

Merge DataFrames together whilst understanding the differences between inner, outer, left, and right merges.

Slice data using the method and create new values based upon a series of bins.

Feel more confident in the ability to fix Python/Pandas bugs within Jupyter Notebook.

Use Google to explore additional Pandas functionality.



Instructor Demonstration

Merging DataFrames

Merging DataFrames

What's Merging?

- Sometimes an analyst will receive data split across multiple tables and sources
- Working across multiple tables is error-prone and confusing
- **Merging** is the process of combining two tables based on shared data.
- Shared data can be an identical column in both tables, or a shared index.
- In Pandas, we can merge separate DataFrames using the `pd.merge()` method.

Merging DataFrames

- Inner joins are the default means through which DataFrames are combined using the `pd.merge()` method and will only return data whose values match. Rows that do not include matching data will be dropped from the combined DataFrame.

In [3]:
```
# Create DataFrames
raw_data_items = {
    "customer_id": [403, 112, 543, 999, 654],
    "item": ["soda", "chips", "TV", "Laptop", "Cooler"],
    "cost": [3.00, 4.50, 600, 900, 150]
}
items_df = pd.DataFrame(raw_data_items, columns=[
                        "customer_id", "item", "cost"])
items_df
```

Out[3]:

	customer_id	item	cost
0	403	soda	3.0
1	112	chips	4.5
2	543	TV	600.0
3	999	Laptop	900.0
4	654	Cooler	150.0

Merging DataFrames

Outer Join

- Outer joins will combine the DataFrames regardless of whether any of the rows match and must be declared as a parameter within the `pd.merge()` method using the syntax `how='outer'`.

In [5]:
```
# Merge two dataframes using an outer join
merge_df = pd.merge(info_df, items_df, on="customer_id", how="outer")
merge_df
```

Out[5]:

	customer_id	name	email	item	cost
0	112	John	jman@gmail	chips	4.5
1	403	Kelly	kelly@aol.com	soda	3.0
2	999	Sam	sports@school.edu	Laptop	900.0
3	543	April	April@yahoo.com	TV	600.0
4	123	Bobbo	HeyImBobbo@msn.com	NaN	NaN
5	654	NaN	NaN	Cooler	150.0

Merging DataFrames

Right and Left Joins

- These joins will protect the data contained within one DataFrame like an outer join does whilst also dropping the rows with null data from the other DataFrame.

In [6]:
```
# Merge two dataframes using a left join
merge_df = pd.merge(info_df, items_df, on="customer_id", how="left")
merge_df
```

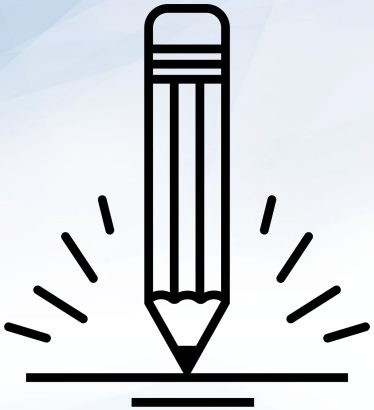
Out[6]:

	customer_id	name	email	item	cost
0	112	John	jman@gmail	chips	4.5
1	403	Kelly	kelly@aol.com	soda	3.0
2	999	Sam	sports@school.edu	Laptop	900.0
3	543	April	April@yahoo.com	TV	600.0
4	123	Bobbo	HeyImBobbo@msn.com	NaN	NaN

In [7]:
```
# Merge two dataframes using a right join
merge_df = pd.merge(info_df, items_df, on="customer_id", how="right")
merge_df
```

Out[7]:

	customer_id	name	email	item	cost
0	112	John	jman@gmail	chips	4.5
1	403	Kelly	kelly@aol.com	soda	3.0
2	999	Sam	sports@school.edu	Laptop	900.0
3	543	April	April@yahoo.com	TV	600.0
4	654	NaN	NaN	Cooler	150.0



Activity: Cryptocurrency Merging

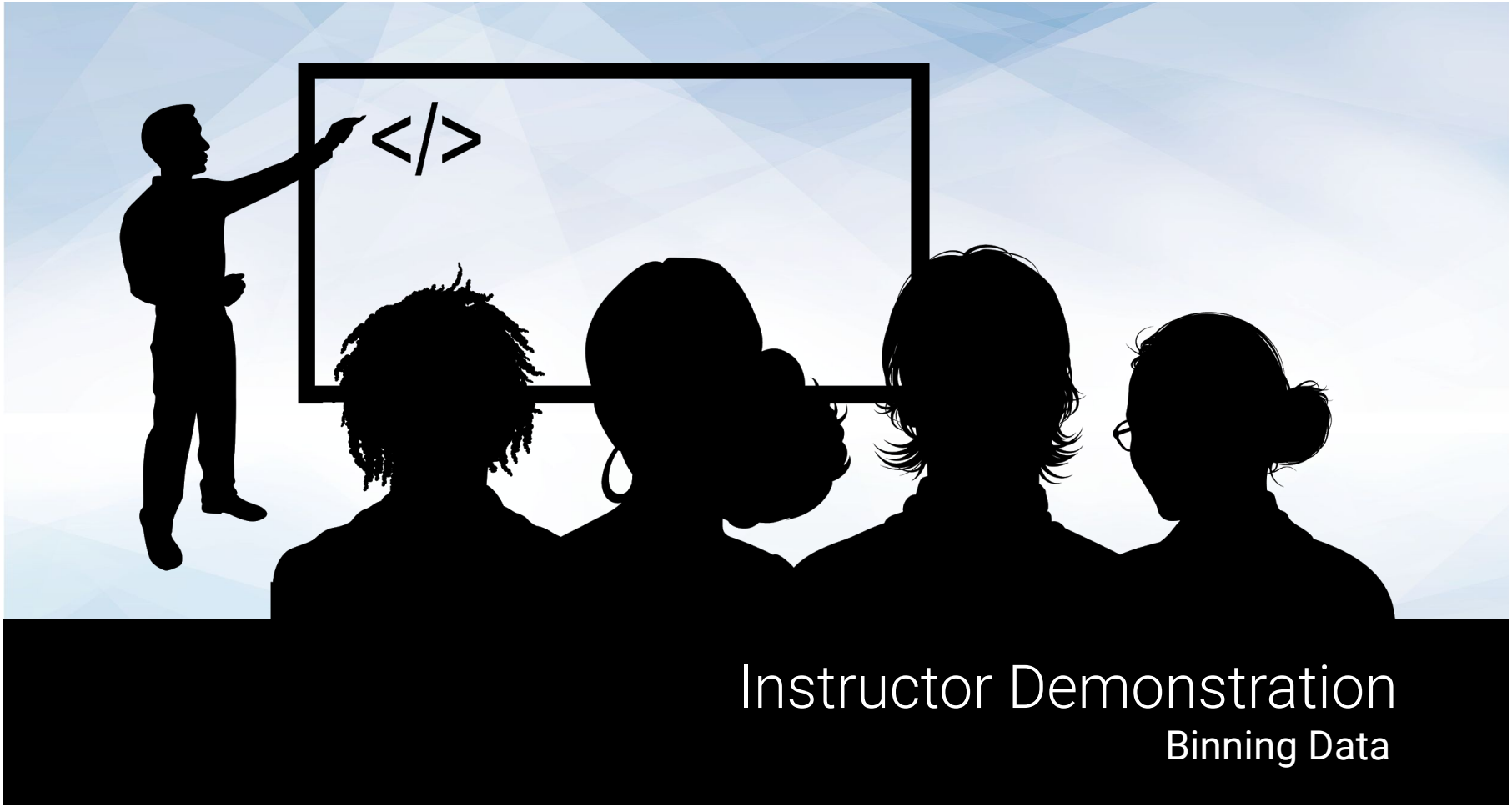
In this activity, you will merge two datasets on cryptocurrencies, one on Bitcoin and the other on Dash. Once fully merged you will clean it to make it more presentable.

Suggested Time:
20 Minutes

Activity: Cryptocurrency Merging

- Read in both of the CSV files and print out their DataFrames.
- Perform an inner merge that combines both DataFrames on the "Date" column.
- Rename the columns within the newly merged DataFrame so that the headers are more descriptive.
- Create a summary table that includes the following information: `Best Bitcoin Open`, `Best Dash Open`, `Best Bitcoin Close`, `Best Dash Close`, `Total Bitcoin Volume`, `Total Dash Volume`.
- `Total Bitcoin Volume` and `Total Dash Volume` should be calculated to have units of "millions" and be rounded to two decimal places.

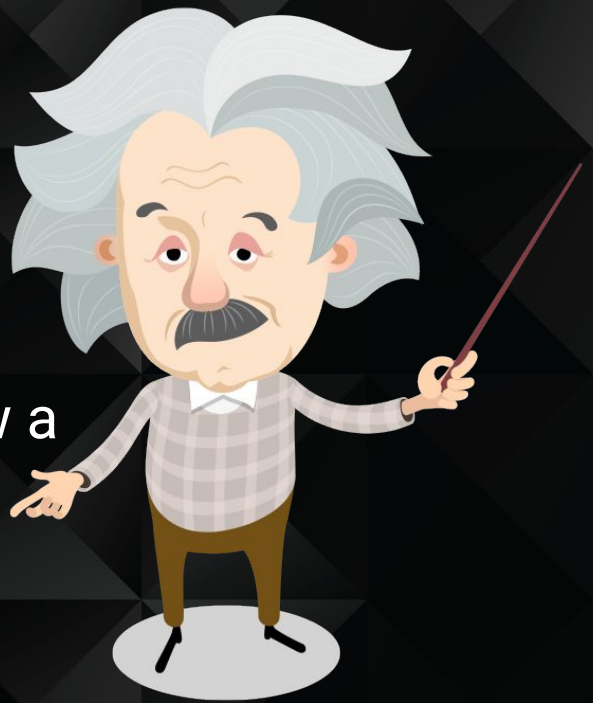
Time’s Up! Let’s Review.



Instructor Demonstration

Binning Data

The "binning" method is an effective function that place values into groups in order to allow a more vigorous customization of datasets.



Binning Data

- Use `pd.cut()` when you need to segment and sort data values into bins. This function is also useful for going from a continuous variable to a categorical variable.

In [18]:
```
# Create the bins in which Data will be held
# Bins are 0, 59.9, 69.9, 79.9, 89.9, 100.
bins = [0, 59.9, 69.9, 79.9, 89.9, 100]

# Create the names for the five bins
group_names = ["F", "D", "C", "B", "A"]
```

In [19]:
```
df["Test Score Summary"] = pd.cut(df["Test Score"], bins, labels=group_names, include_lowest=True)
df
```

Out[19]:

	Class	Name	Test Score	Test Score Summary
0	Oct	Cyndy	90	A
1	Oct	Logan	59	F
2	Jan	Laci	72	C
3	Jan	Elmer	88	B
4	Oct	Crystle	98	A
5	Jan	Emmie	60	D

Binning Data

- What makes binning so powerful is that, after creating and applying these bins, the DataFrame can be grouped according to those values and thus a higher-level analysis can be conducted.

In [19]:
```
df["Test Score Summary"] = pd.cut(df["Test Score"], bins, labels=group_names, include_lowest=True)
df
```

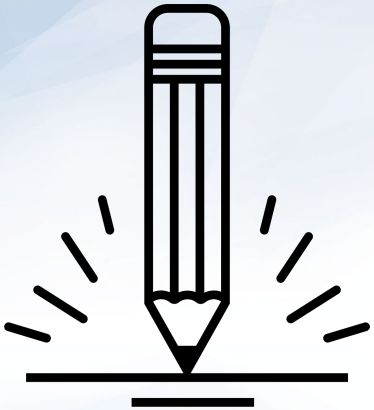
Out[19]:

	Class	Name	Test Score	Test Score Summary
0	Oct	Cyndy	90	A
1	Oct	Logan	59	F
2	Jan	Laci	72	C
3	Jan	Elmer	88	B
4	Oct	Crystle	98	A
5	Jan	Emmie	60	D

In [20]:
```
# Creating a group based off of the bins
df = df.groupby("Test Score Summary")
df.max()
```

Out[20]:

	Class	Name	Test Score
Test Score Summary			
F	Oct	Logan	59
D	Jan	Emmie	60
C	Jan	Laci	72
B	Jan	Elmer	88
A	Oct	Cyndy	98



Activity: Binning TED

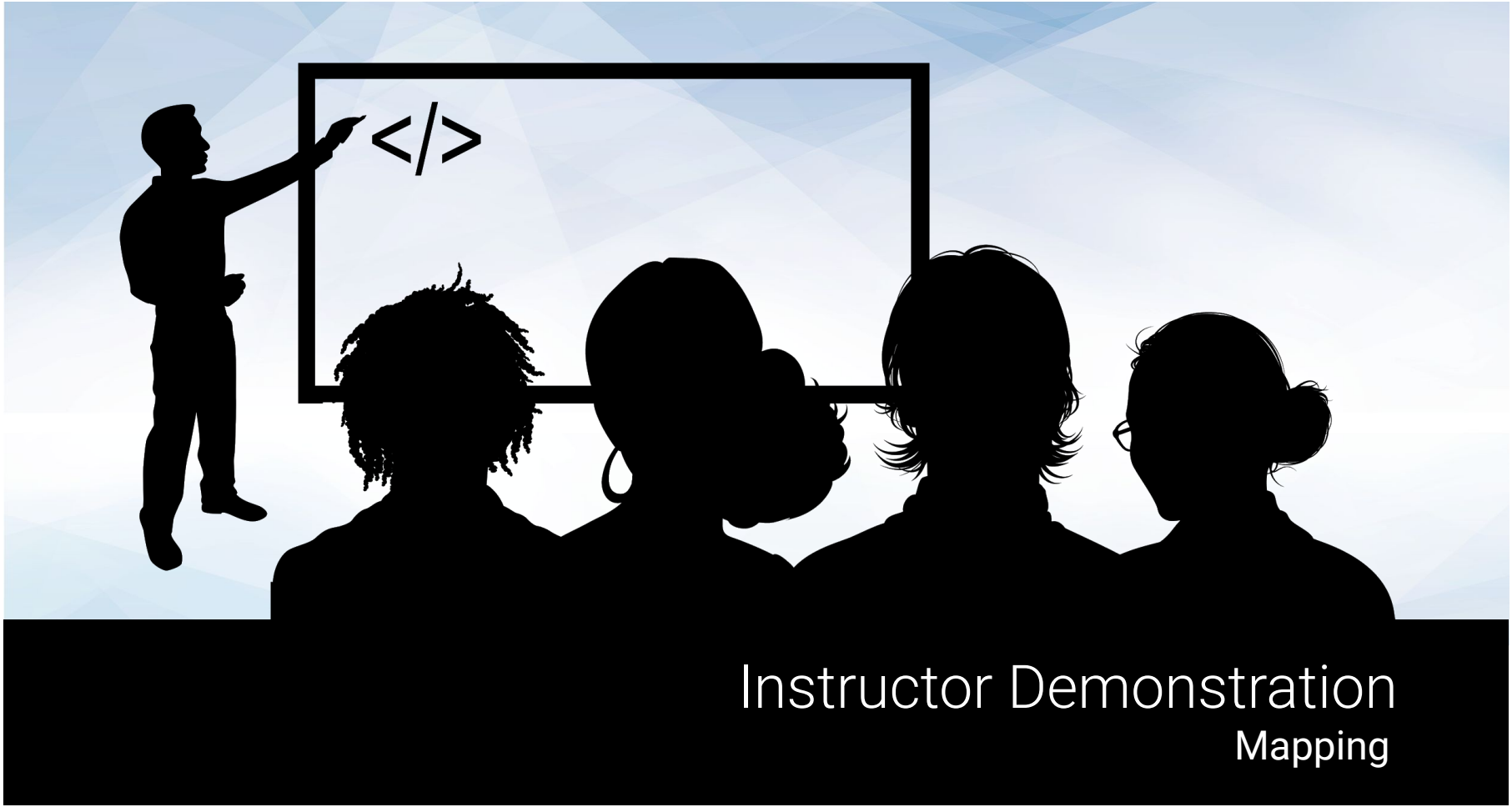
In this activity, you will put your binning skills to the test by creating bins for TED Talks based upon their viewership.

Suggested Time:
25 Minutes

Activity: Binning TED

- Read in the CSV file provided and print it to the screen.
- Find the minimum "views" and maximum "views".
- Using the minimum and maximum "views" as a reference, create 10 bins in which to slice the data.
- Create a new column called "View Group" and fill it with the values collected through your slicing.
- Group the DataFrame based upon the values within "View Group".
- Find out how many rows fall into each group before finding the averages for "comments", "duration", and "languages".

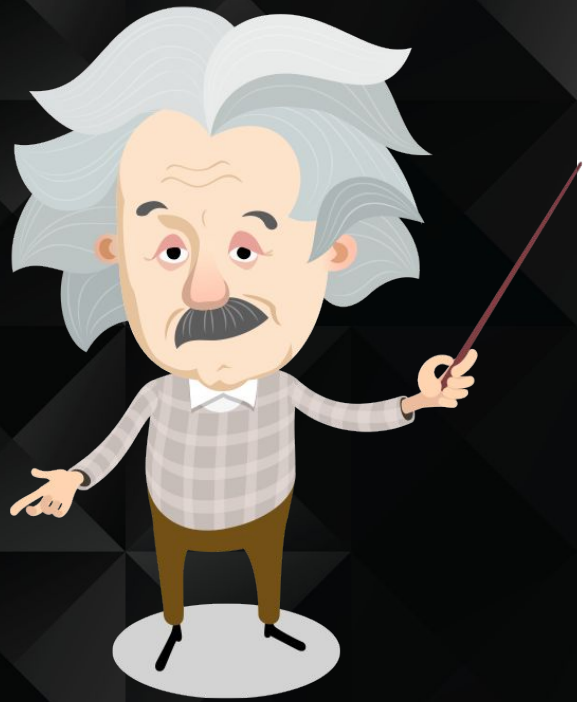
Time’s Up! Let’s Review.



Instructor Demonstration

Mapping

Similar to Excel’s number formats, Pandas unlocks the same functionality using the `df.map()` method, thus allowing users to style columns wholesale.



Mapping

- `df[<COLUMN>].map(<FORMAT STRING>.format)` is the method by which users can modify the styling of an entire column.
- To convert values into a typical dollar format, use `"${:.2f}"`. This places a dollar sign before the value which has been rounded to two decimal points.
- Using `"{:,}"` will split a number up so that it uses comma notation.

In [3]:
```
# Use Map to format all the columns
file_df["avg_cost"] = file_df["avg_cost"].map("${:.2f}".format)
file_df["population"] = file_df["population"].map("{:,}".format)
file_df["other"] = file_df["other"].map("{:.2f}".format)
file_df.head()
```

Out[3]:

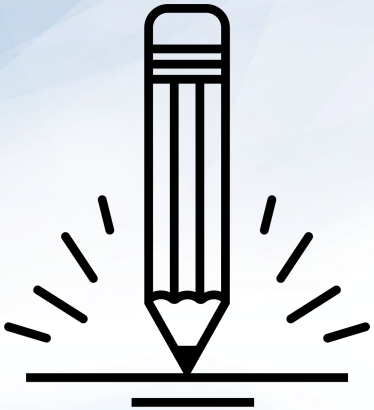
	id	city	avg_cost	population	other
0	1	Houxiang	$55.12	609,458	-15.66
1	2	Leribe	$95.78	601,963	-23.79
2	3	Hengshan	$57.87	589,509	1.31
3	4	Sogcho	$59.22	948,491	-11.38
4	5	Kohlu	$23.09	92,206	7.67

Mapping

- Format mapping only really works once and will return errors if the same code is run multiple times without restarting the kernel. Because of this, formatting is usually applied near the end of an application.
- Format mapping also can change the datatype of a column. As such, all calculations should be handled before modifying the formatting.

```
In [4]: # Mapping has changed the datatypes of the columns to strings
        file_df.dtypes
```

```
Out[4]: id              int64
        city           object
        avg_cost       object
        population     object
        other          object
        dtype: object
```



Activity: Cleaning Kickstarter

In this activity, you will take a dataset similar to your first homework, clean it up, and format it.

Suggested Time:
30 Minutes

Activity: Cleaning Kickstarter

- The instructions for this activity are contained within the Jupyter Notebook.

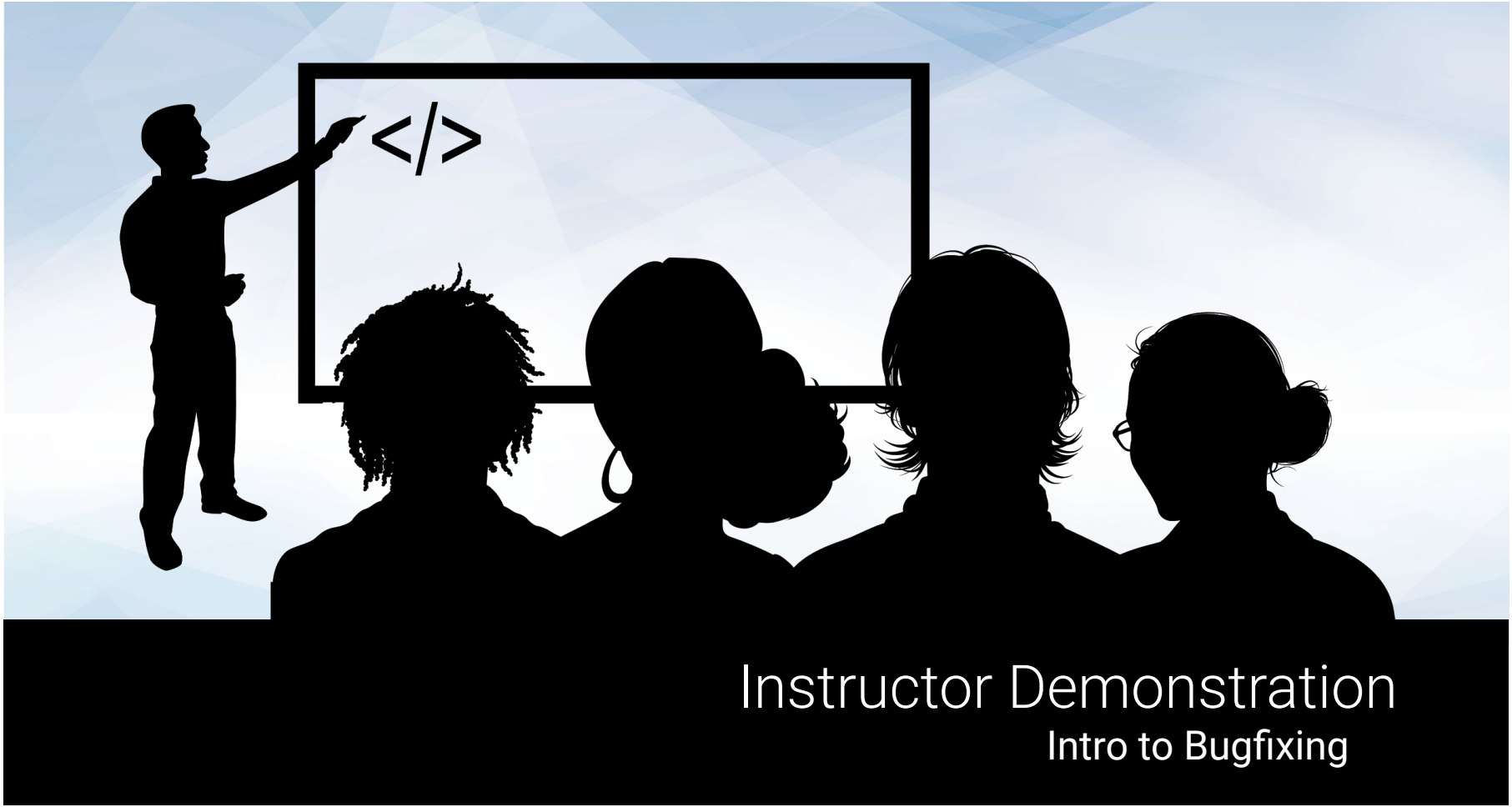
```
In [ ]: import pandas as pd

In [ ]: # The path to our CSV file

        # Read our Kickstarter data into pandas

In [ ]: # Get a list of all of our columns for easy reference

In [ ]: # Extract "name", "goal", "pledged", "state", "country", "staff_pick",
        # "backers_count", and "spotlight"

In [ ]: # Remove projects that made no money at all

In [ ]: # Collect only those projects that were hosted in the US
        # Create a list of the columns
        # Create a new df for "US" with the columns above.

In [ ]: # Create a new column that finds the average amount pledged to a project

In [ ]: # First convert "average_donation", "goal", and "pledged" columns to float
        # Then Format to go to two decimal places, include a dollar sign, and use comma notation

In [ ]: # Calculate the total number of backers for all US projects

In [ ]: # Calculate the average number of backers for all US projects

In [ ]: # Collect only those US campaigns that have been picked as a "Staff Pick"

In [2]: Group by the state of the campaigns and see if staff picks matter (Seems to matter quite a bit)
```

Time’s Up! Let’s Review.

Countdown timer

40:00

(with alarm)



Instructor Demonstration

Intro to Bugfixing

Bugfixing Bonanza

- Note that an error is being returned as the application attempts to collect the average value within the "Cocoa Percent" column.

```
In [5]: # Finding the average cocoa percent
        chocolate_ratings_df["Cocoa Percent"].mean()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/anaconda3/lib/python3.6/site-packages/pandas/core/nanops.py in _ensure_numeric(x)
    818         try:
--> 819             x = float(x)
    820         except Exception:

ValueError: could not convert string to float: '63%70%70%70%70%70%70%70%70%70%70%70%70%70%70%7
0%70%70%70%63%70%63%70%70%60%80%88%72%55%70%70%75%75%75%65%75%75%75%75%70%70%70%70%60%60%60%60%60%
60%60%80%60%60%70%70%70%70%70%70%70%70%70%70%70%85%85%72%73%64%66%75%63%70%68%70%70%75%70%70%70%7
0%70%70%70%63%70%66%75%85%50%75%60%75%75%75%72%75%75%70%70%73%70%70%70%70%70%70%70%70%70%73%70%68%
70%70%70%70%75%70%75%72%72%72%72%100%72%72%72%72%75%72%72%80%75%72%72%72%68%72%70%77%75%70%80%70%7
0%70%70%70%70%70%70%70%70%80%65%70%65%73%72%80%70%70%90%64%64%64%71%70%70%70%83%78%83%74%74%74%
73%72%72%55%64%88%72%72%70%74%64%72%76%76%78%86%72%75%70%65%70%78%75%65%75%65%71%75%68%70%70%70%7
0%70%82%72%82%75%75%75%70%70%75%75%75%65%75%75%75%75%75%75%75%75%75%75%75%75%75%75%75%75%75%75%
75%100%75%75%77%100%70%70%70%70%68%70%70%72%70%75%85%60%80%70%80%80%60%70%72%70%72%68%70%68%72%72%
72%72%60%70%75%75%75%75%65%70%75%72%66%77%75%75%74%75%70%74%71%74%72%64%70%69%70%70%72%72%66%65%7
0%71%71%71%71%71%77%70%70%70%70%70%63%71%82%91%60%70%75%72%70%100%80%78%70%75%73%70%75%70%77%7
7%77%55%55%55%70%70%72%72%72%72%80%65%65%65%65%65%72%72%70%72%72%76%70%70%73%70%70%72%70%72%70%70%
```

Bugfixing Bonanza

- Bugs happen all the time and they are rarely the end of the world. In fact, most bugs that programmers run across are simple enough to solve so long as they know how and where to look for the solution.

Bugfxing Bonanza

The first step is figuring out what the bug is and where it is located

- In Jupyter Notebook is quite easy to find the erroneous block of code since the error will always be returned in the space beneath the erroneous cell.
- Unfortunately Pandas is not known for returning clearly understandable error text. In fact, it often returns large blocks of text that is complex and confusing to those who do not know the library's underlying code. Looking for the line following `KeyError:` is generally a good starting point.
- For example, the text following `TypeError:` within the current code lets the programmer know that Pandas cannot convert the string values in the "Cocoa Percent" column to floats.

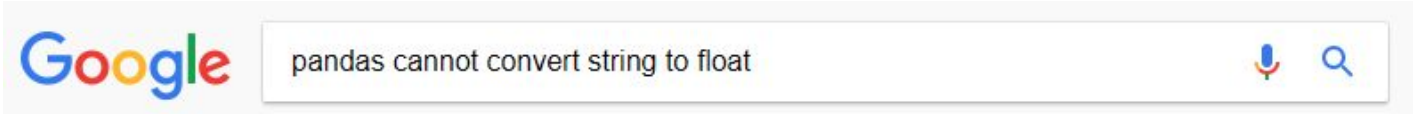
```
ValueError: could not convert string to float:
```

- If the error text is not entirely clear, it is oftentimes helpful to print out variables/columns to the console in order to uncover where the bug is. For example, printing out the "Cocoa Percent" series lets the programmer know that the `dtype` of this series is an object and not a float.

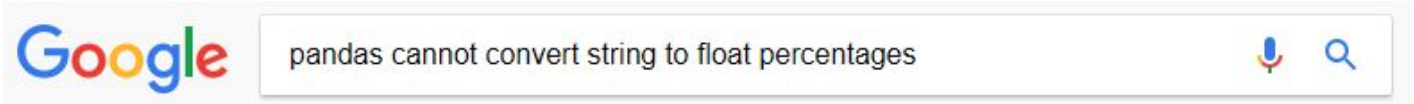
Bugfixing Bonanza

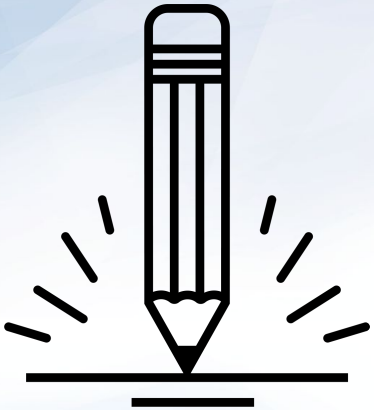
Step 2: Google up the error online and source for solutions

- The key part to this step is coming up with an accurate way to describe the bug. This may take multiple tries and is a skill that will develop over time.
- Google is the programmer's best friend as typing in a description of the bug being faced will often bring up links to some possible solutions. If not, simply change the search up a little bit until a solution is discovered.



- This particular problem requires the code to drop the percentages within the "Cocoa Percent" column, so the search should be a bit more specific.





Activity: Bugfixing Bonanza

In this activity, you will be provided with a Pandas project containing TONS of bugs inside of it. Your job will be to take the application and fix it up so that it works properly.

Suggested Time:
35 Minutes

Activity: Bugfixing Bonanza

- Dig through the Jupyter Notebook provided and attempt to fix as many bugs as possible. There are a lot of them and the bugs get harder to deal with as the code progresses.

- Once you have finished bugfixing, perform some additional analysis on the dataset provided. See what interesting trends are buried deep within these bug logs for the Eclipse IDE. So long as you challenge yourself, bugs will pop up and you will get even more bugfixing practice.

- **Hints:**
  - 
    After fixing the bugs in each block of code, be sure to run the cell below for an updated error.
  - There are a few new concepts being covered within this Jupyter Notebook. The most complex of these concepts is that of multi-indexing and it is very likely that this is where many will get held up. Do not worry though, multi-indexing is not in the homework and is not required outside of this activity. It is simply an interesting/powerful feature of Pandas.

Time’s Up! Let’s Review.

last
but
not least..


The End