Comparison Operators

The result of a comparison expression is a `bool` value

Assignment statements:

- `x = 2`
- `y = 3`

Comparison expressions:

- `x > 1`
- `x > y`
- `y >= 3`
- `x == y`
- `x != 2`
- `2 < x < 5`
Combining Comparisons

Boolean operators can be applied to `bool` values

\[
\begin{align*}
a & = \text{True} & b & = \text{False} \\
\text{not } b & & a \lor b & & a \land \text{not } b \\
\text{a and b} & & \text{not } (a \lor b) & & b \land b
\end{align*}
\]

Evaluate to **True**

Evaluate to **False**

(Demo)
Aggregating Comparisons

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

\[
1 + 0 + 1 = 2 \\
\text{True} + \text{False} + \text{True} = 2 \\
\text{sum}([1, 0, 1]) = 2 \\
\text{sum}([\text{True}, \text{False}, \text{True}]) = 2 \\
\text{np.count_nonzero}([\text{True}, \text{False}, \text{True}]) = ? \\
\]

(Demo)
More Python Commands

- Printing
  - Use `print` to display the value of a variable

- Control Statements
  - The purpose of `if` is to define functions that choose different behavior based on their arguments
  - The purpose of `for` is to perform a computation for every element in a list or array

(Demo)
Defining Functions

User-defined functions give names to blocks of code.

```
def spread(values):
    return max(values) - min(values)
```
What does this function do? What kind of input does it take? What output will it give? What's a reasonable name?

```python
def f(s):
    return np.round(s / sum(s) * 100, 2)
```

(Demo)
Apply

The **apply** method creates an array by calling a function on every element in one or more input columns

- **First argument:** Function to apply
- **Other arguments:** The input column(s)

```
table_name.apply(one_arg_function, 'column_label')
```

```
table_name.apply(two_arg_function,
                'column_label_for_first_arg',
                'column_label_for_second_arg')
```

**apply** called with only a function applies it to each row

(Demo)
Applying functions to tables

- Go back to Lab 3, Questions 3 and 4
- Work in groups on the problems
Group

The **group** method aggregates all rows with the same value for a column into a single row in the result

- First argument: Which column to group by
- Second argument: (Optional) How to combine values
  - `len` — number of grouped values (default)
  - `sum` — total of all grouped values
  - `list` — list of all grouped values
Grouping By Two Columns

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

- First argument: A list of which columns to group by
- Second argument: (Optional) How to combine values
What’s next?

• Read Chapter 8-9 of *Computational and Inferential Thinking*

• Continue working on Project 1