Data 8 Summer 2021
Discussion: Functions and Table Methods (Project 1 Discussion)

Welcome to Project 1 Discussion! This week we will be discussing functions and table methods like group. The group function allows us to aggregate the unique entries in one or more columns.

Question 1 Fun with Functions

a. The following code has a number of errors in it. Which ones can you identify?

```
def hypotenuse(a, b)
      ```Returns the length of the hypotenuse of a right triangle,
      the squareroot of a squared + b squared```
      squares = make_array(a, b)*2
      sum = sum(squares)
      squareroot = np.sqrt(sum)
      print(squareroot)
A = 5
B = 5
C = squareroot
```

Solution:
Error 1: the function is missing a colon ":" after the arguments list.
Error 2: squares should be squared with ** not *
Error 3: don't use the names of existing functions like sum as variable names, although this won't error it's still bad practice
Error 4: the function will print the value of C but won't return it, which means we won't have access to the value of C anymore-- ie we won't be able to assign it to any values or use it as the argument to any functions!
Error 5: We cannot access the `squareroot` name outside of the body of the function! We have to use `C = hypotenuse(A, B)`

b. Write a function that takes in one argument, a table tbl, another argument, a name of a column in that table col, and a boolean largest, and returns a table that contains the rows that have the ten largest or ten smallest values for the specified column, largest if the boolean largest is True, smallest if the boolean argument is False.

Solution:

```
def top_ten(tbl, col, largest):
        sorted_tbl = tbl.sort(col, descending=largest)
        ten_rows = sorted_tbl.take(np.arange(10))
        return ten_rows
```

c. Can a function take no arguments? When would you use a function with no arguments? How do you call a function without arguments? How does that compare to using a function as an argument?

Yes a function can take no arguments! To call a function with no arguments, we just write the
```

name of the function followed by empty parentheses i.e. name(). We might use a function with no arguments if the function has some element of randomness to it, ie every time we call it, it returns something different. Otherwise, if a function returns the exact same thing every time we might consider assigning that constant value to a name instead. Functions are names--when we pass one to group or apply for example it is the name of the function. To get the function to do something, you need to call it, it add the parentheses.

Question 2. Ian has opened up a chocolate store where he sells small boxes of chocolates in groups of different sizes and colors. His table `chocolates` is as follows:

Color	Shape	Amount	Price ($)
Dark	Round	4	1.30
Milk	Rectangular	6	1.20
White	Rectangular	12	2.00
Dark	Round	7	1.75
Milk	Rectangular	9	1.40
Milk	Round	2	1.00

Notice that the table contains multiple rows containing information about chocolates of the same color. We would like to figure out how many chocolates of each color he has for sale in total, and what the cost would be to purchase all chocolates of each unique color.

a. Write a line of code that will return a new table which displays the total number of boxes for each color.

```
chocolates.group('Color')
```

b. Write a line of code which will return a new table with the total number of chocolates and the total cost for each unique color. For example, the row for "Dark" should have a total of 4+7=11 chocolates, and a total cost of $1.30 + $1.75 = 3.05.

```
chocolates.group('Color', sum)

chocolates.drop('Shape').group('Color', sum)
```

Question 3. Some rows from the table `ca` are shown below. The table contains information about the most common baby names in California and the number of those occurrences in a particular year, from the years 1910-2019. (This dataset was submitted by a fellow Data 8 student!)

State	Sex	Year	Name	Occurrence
CA	F	1910	Mary	295
CA	F	1910	Helen	239

CA	F	1910	Dorothy	220
CA	F	1910	Margaret	163

a. Write a line of code that will return the most popular name over all the years.
Hint: Think about how to use the second argument in `.group`

```
ca.group('Name', sum).sort('Occurrence sum',
descending=True).column('Name').item(0)
```

b. Instead of the most popular name over all the years, write a line of code that will return the top 10 most popular names over all the years.

```
ca.group('Name', sum).sort('Occurrence sum',
descending=True).take(np.arange(10)).column('Name')
```

c. The top 10 names all appeared to be male names. Write a line of code that would return the most popular female names instead.

```
ca.where('Sex', 'F').group('Name', sum).sort('Occurrence sum',
descending=True).take(np.arange(10)).column('Name')
```

d. Write a line of code that will return the most popular female name in 1969

```
ca.where('Sex', 'F').where('Year', 1969).sort('Occurrence',
descending=True).column('Name').item(0)
```

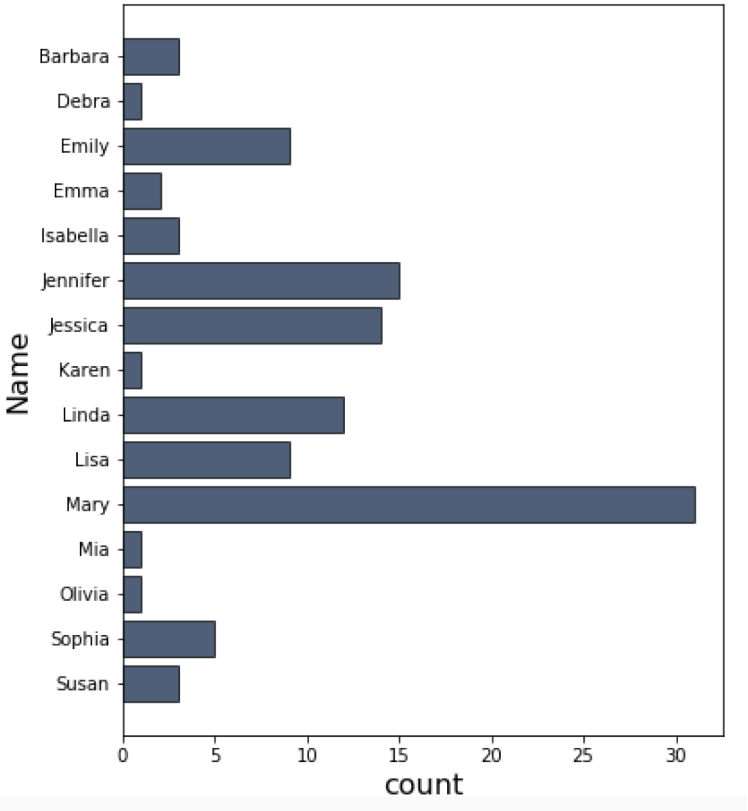
e. Write a function `most_popular_female_name` that takes in a year as an argument and returns the most popular female name in that year

```
def most_popular_female_name(year):
     return ca.where('Sex', 'F').where('Year', year)
     .sort('Occurrence', descending=True).column('Name').item(0)
```

f. The `ca` table is from 1910-2019. Define the years table with a column year and a row for each year from 1910-2019 (inclusive). Then create the table `popular_female_names` that has 2 columns, a year and a column for the female name that is most popular.

```
years = Table().with_column('Year', np.arange(1910,2020))
most_popular_female_names_array =
years.apply(most_popular_female_name, 'Year')
popular_female_names = years.with_column('Name',
most_popular_female_names_array)
```

g. Write a line of code that will generate the following bar chart:




```
popular_female_names.group('Name').barh('Name')
```

CAN'T DO THIS WEEK

Question 4. The table below, called `weights`, contains information about the weights of the chocolates that are sold. The weights of the chocolates differ depending on the shape, and round chocolates have two different sizes.

Shape	**Weight(g)**
Round	3.1
Round	4.25
Rectangular	3.6
Circle	2.9

The following line of code has been executed in a blank cell. Take a moment to discuss with your neighbors what the resulting table will look like. Then, write the number of columns and rows in the resulting table, and describe the information in the table in 1-2 sentences.

Hint: It may help to draw a sketch of the resulting table!

```
chocolates.join('Shape', weights)
```

Rows: 9, Columns: 5
The resulting table will look like this:

Shape	Color	Amount	Price ($)	Weight (g)
Rectangular	Milk	6	1.20	3.6
Rectangular	White	12	2	3.6
Rectangular	Milk	9	1.40	3.6
Round	Dark	4	1.30	3.1
Round	Dark	4	1.30	4.25
Round	Dark	7	1.75	3.1
Round	Dark	7	1.75	4.25
Round	Milk	2	1.00	3.1
Round	Milk	2	1.00	4.25

Question 5. We will continue with the same table as before, copied below for your convenience.

Color	Shape	Amount	Price ($)
Dark	Round	4	1.30
Milk	Rectangular	6	1.20
White	Rectangular	12	2.00
Dark	Round	7	1.75
Milk	Rectangular	9	1.40
Milk	Round	2	1.00

Write code to create a pivot table on the colors and shapes of chocolates, finding the average price for each color-shape combination. Then, fill in the blank table in the image of the resulting table.

```
chocolates.pivot('Shape', 'Color', values='Price ($)',
collect=np.average)
```

Hint: You can use the `np.average` function to find the average of an array of inputs. The average of no values is marked as zero.

Color	Rectangular	Round
White	2	0
Milk	1.3	1
Dark	0	1.525