Project 1 Lab: Functions, Table Methods

Data 8 Discussion Worksheet

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

1. Fun with Functions

a. After learning about them in Data 8, Noor wants to write a function that can calculate the hypotenuse of any right triangle. She wants to use her function to assign `C` to the hypotenuse of a triangle with side lengths, `A`, `B`, and `C`. However, she's made a few mistakes. Which ones can you identify? *HINT*: There are five unique errors. Assume that `numpy` has been imported as `np`.

```
def hypotenuse(a, b)
"""Returns the length of the hypotenuse of a right triangle,
   the square root 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: When we assign `sum` to a number we've lost the original behavior of the built-in `sum` function. We shouldn't re-assign variable names.
Error 4: The function will print the value of `squareroot` but won't return it, which means we won't have access to the value of `squareroot` anymore. That is, 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! To fix this, we'd have to use `C = hypotenuse(A, B)`

b. Write a function that takes in the following arguments: `tbl`: a table, `col`: a name of a column in `tbl`, and `n`: an integer. The function should return a table that contains the rows that have the `n` *largest* values for the specified column.

```
def top_n(tbl, col, n):
    sorted_tbl = tbl.sort(col, descending=True)
    top_n_rows = sorted_tbl.take(np.arange(n))
    return top_n_rows
```

2. Ellen and the Chocolate Factory

Ellen has opened up a chocolate factory where she sells small boxes of chocolates in groups of different sizes and colors. Her 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

Each row of the `chocolates` table represents **one box**. 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 she 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 evaluates to a new table which displays the total number of boxes for each color. *Hint*: *What is the default behavior of the group function?*

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

b. Write a line of code which evaluates to a new table with three columns: the color, 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).drop('Shape sum')

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

3. California Names

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. For this problem, assume that "female" refers to an "F" in the Sex column of our table, representing that the child was assigned female at birth.

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 evaluates to the most popular name overall.
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. Write a line of code that evaluates to the most popular female name in 1969.

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

c. Using your answer to part (b), Write a function `most_popular_female_name` that takes in `year` as an argument and returns the most popular female name in that year.

```
def most_popular_female_name(year):
    filtered_tbl = ____________________________________________
    names_array = _____________________________________________
    return ____________________________________________________
```

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