# Project Activity: More Penguins!

This Discussion activity is a component of your group mini-project (https://philchodrow.github.io/PIC16A/project/). While the usual Discussion expectations apply with regards to your participation grade (i.e. if you work for the full 50 minutes, you will get full credit), it is recommended for the purposes of your final project that you coordinate with your group to eventually complete all parts of this assignment.

## Group Roles

The roles for this Discussion activity are slightly modified. The Driver and Proposer are the same as usual. Instead of a Reviewer, use an **Interpreter**. The job of the Interpreter is to think about the significance of each of the code outputs in the context of the long-term project goal (classifying penguin species). In parts of the Discussion where the problems ask you to explain or interpret your findings, the Interpreter should suggest responses to the Proposer and Driver. The **Interpreter** may also give code feedback when the group is writing functions.

### List Names and Group Roles Here:

- david (driver)
- Rashi (proposer)
- Lauren(reviewer)

## Part A

Run the following cell to load the penguin dataset as a `pandas DataFrame` called `penguins`. I've also supplied code to shorten the penguins species name for convenient exploration and plotting.

*If you experience `ConnectionRefused` errors when doing this, instead copy/paste the url into your browser. Save the data in the same directory as this notebook in a file called `penguins.csv`, and then replace `url` with `"penguins.csv"` in the block below.*

While working with this dataset, you may notice some blank or nonsensical values. Normally, for a project such as this, we would want to remove these values before we continue. However, in this worksheet you can just ignore them.

In [1]:
```
import pandas as pd
import numpy as np
import urllib
from matplotlib import pyplot as plt

url = "https://philchodrow.github.io/PIC16A/datasets/palmer_penguins.csv"
penguins = pd.read_csv(url)

# shorten the species name
penguins["Species"] = penguins["Species"].str.split().str.get(0)
```

In [5]:
```
# optional code here if you need to refresh your memory of the data set
penguins.head()
```

Out[5]:

| | studyName | Sample Number | Species | Region | Island | Stage | Individual ID | Clutch Completion | Date Egg | Culmen Length (mm) | Culmen Depth (mm) | Flipper Length (mm) | Body Mass (g) | Sex | Delta 15 N (o/oo) | De C ( |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | PAL0708 | 1 | Adelie | Anvers | Torgersen | Adult, 1 Egg Stage | N1A1 | Yes | 11/11/07 | 39.1 | 18.7 | 181.0 | 3750.0 | MALE | NaN | |
| **1** | PAL0708 | 2 | Adelie | Anvers | Torgersen | Adult, 1 Egg Stage | N1A2 | Yes | 11/11/07 | 39.5 | 17.4 | 186.0 | 3800.0 | FEMALE | 8.94956 | -24.( |
| **2** | PAL0708 | 3 | Adelie | Anvers | Torgersen | Adult, 1 Egg Stage | N2A1 | Yes | 11/16/07 | 40.3 | 18.0 | 195.0 | 3250.0 | FEMALE | 8.36821 | -25.3 |
| **3** | PAL0708 | 4 | Adelie | Anvers | Torgersen | Adult, 1 Egg Stage | N2A2 | Yes | 11/16/07 | NaN | NaN | NaN | NaN | NaN | NaN | |
| **4** | PAL0708 | 5 | Adelie | Anvers | Torgersen | Adult, 1 Egg Stage | N3A1 | Yes | 11/16/07 | 36.7 | 19.3 | 193.0 | 3450.0 | FEMALE | 8.76651 | -25.3 |

## Part B

Write a function called `penguin_summary_table` which accepts two arguments, `group_cols` and `value_cols`. This function should create a table in which the mean of each element of `value_cols` is shown, grouped according to the specified `group_cols`. For example, the call

```
penguin_summary_table(["Species"], ["Culmen Length (mm)", "Culmen Depth (mm)"])
```

should produce a summary table with the mean culmen length and depth per species.

For a more pleasant display, **round the numbers in your table to 2 decimal places**. This can be done using the code `my_data_frame.round(2)`.

This function can be implemented in just a few lines. Comments and docstrings are not necessary.

In [22]:
```
# your solution here
def penguin_summary_table(group_cols, value_cols):
    #summary = penguins.groupby(group_cols).aggregate(value_cols).apply(mean)
    summary = penguins.groupby(group_cols)[value_cols].mean()
    return summary.round(2)
```

In [27]:
```
penguin_summary_table(["Species", "Island", "Sex"], ["Culmen Length (mm)", "Culmen Depth (mm)"])
```

Out[27]:

| | | | Culmen Length (mm) | Culmen Depth (mm) |
|---|---|---|---|---|
| **Species** | **Island** | **Sex** | | |
| **Adelie** | **Biscoe** | **FEMALE** | 37.36 | 17.70 |
| | | **MALE** | 40.59 | 19.04 |
| | **Dream** | **FEMALE** | 36.91 | 17.62 |
| | | **MALE** | 40.07 | 18.84 |
| | **Torgersen** | **FEMALE** | 37.55 | 17.55 |
| | | **MALE** | 40.59 | 19.39 |
| **Chinstrap** | **Dream** | **FEMALE** | 46.57 | 17.59 |
| | | **MALE** | 51.09 | 19.25 |
| **Gentoo** | **Biscoe** | **.** | 44.50 | 15.70 |
| | | **FEMALE** | 45.56 | 14.24 |
| | | **MALE** | 49.47 | 15.72 |

In [29]:
```
penguin_summary_table(["Species", "Island"], ["Culmen Length (mm)", "Culmen Depth (mm)"])
```

Out[29]:

| | | Culmen Length (mm) | Culmen Depth (mm) |
|---|---|---|---|
| **Species** | **Island** | | |
| **Adelie** | **Biscoe** | 38.98 | 18.37 |
| | **Dream** | 38.50 | 18.25 |
| | **Torgersen** | 38.95 | 18.43 |
| **Chinstrap** | **Dream** | 48.83 | 18.42 |
| **Gentoo** | **Biscoe** | 47.50 | 14.98 |

## Part C

Use your function to explore the data a bit. Focus on the physiological variables:

- `Culmen Length (mm)`
- `Culmen Depth (mm)`
- `Flipper Length (mm)`
- `Body Mass (g)`
- `Delta 15 N (o/oo)`
- `Delta 13 C (o/oo)`

These last two variables are measures of nitrogen and carbon isotopes in the penguin's bloodstreams.

**Create at least three readable summary tables.** Then, work with your **Interpreter** to explain the significance of each table. Do observe any important differences between the penguin species?

Make sure that each table has a message, and that no information is shown that is not part of that message. Is there a part of the table that you have nothing to say about? Remove it!

- **Hint**: "This table suggests that there's not much of a difference between..." is a fine explanation of the table, as long as it's warranted.
- **Hint**: consider the sex of the penguins as well as the species.
  - There is a single penguin whose sex was not collected by the researchers and encoded as `.`. This should not cause major problems, but feel free to remove this row if you'd like to.

In [25]:
```
# Table 1
penguin_summary_table(['Species'],['Flipper Length (mm)', "Body Mass (g)"])
```

Out[25]:

| Species | Flipper Length (mm) | Body Mass (g) |
|---|---|---|
| **Adelie** | 189.95 | 3700.66 |
| **Chinstrap** | 195.82 | 3733.09 |
| **Gentoo** | 217.19 | 5076.02 |

**Discussion of Table 1**

This chart shows that the Gentoo is the heaviest and has the longest flipper length. The Adelie is the smallest and lightest and the Chinstrap is in the middle.

In [24]:
```
# Table 2
penguin_summary_table(['Sex'],['Body Mass (g)'])
```

Out[24]:

| Sex | Body Mass (g) |
|---|---|
| **.** | 4875.00 |
| **FEMALE** | 3862.27 |
| **MALE** | 4545.68 |

**Discussion of Table 2**

In this chart we can see that the male penguins are usually heavier than the female penguins.

In [23]:
```
# Table 3
penguin_summary_table(['Species', 'Sex'],['Delta 15 N (o/oo)', 'Delta 13 C (o/oo)' ])
```

Out[23]:

| Species | Sex | Delta 15 N (o/oo) | Delta 13 C (o/oo) |
|---|---|---|---|
| **Adelie** | **FEMALE** | 8.79 | -25.79 |
| | **MALE** | 8.93 | -25.83 |
| **Chinstrap** | **FEMALE** | 9.25 | -24.57 |
| | **MALE** | 9.46 | -24.53 |
| **Gentoo** | **.** | 8.04 | -26.18 |
| | **FEMALE** | 8.19 | -26.20 |
| | **MALE** | 8.30 | -26.17 |

**Discussion of Table 3**

In this chart we can see that the delta 15n and 13c levels will vary between different penguin species but within the the same speicies the difference is small but usually the male penguin will have higher levels.

## Part D

Based on your findings from these tables, propose a miniature decision tree to help distinguish between the penguin species. Your decision tree might have rules like the following:

1. First, check the island on which the penguin was found.
    A. If Torgersen, then check the body mass.
        a. If the body mass is over 4,000g, then guess Adelie.
        b. Otherwise, guess Chinstrap
    B. If Biscoe, then check the sex of the penguin.
        a. If female, guess Gentoo
        b. Otherwise, guess Chinstrap
    C. If Dream, then guess Adelie.

Your decision tree should operate using no more than three columns from the data.

Your decision tree should operate using no more than three columns from the data.

Below your decision tree, write an explanation of how you came up with it and how the tables that you created above informed your choices.

If you like, you may skip ahead to Part E and write your decision tree directly as a Python function. You should then explain your reasoning as a docstring in the function rather than typing it here.

## Part E

Write a function called `decision_tree` that implements your decision tree. It should accept as input single values of the relevant variables, and then return as output the guessed species of a penguin. Here's an example for the decision tree above:

```
def decision_tree(island, mass, sex):
    if island == "Torgersen":
        if mass > 4000:
            return "Adelie"
        else:
            return "Chinstrap"
    elif island == "Biscoe":
        if sex == "FEMALE":
            return "Gentoo"
        else:
            return "Chinstrap"
    else:
        return "Adelie"

decision_tree("Biscoe", 5000, "MALE")
```

```
'Chinstrap'
```

Comments and docstrings are not necessary in this case, unless you skipped Part D.

In [32]:
```
# your decision tree function here
def decision_tree(culmen_length, delta_n, island):
    if culmen_length > 45: #we know adelie has a small culmen length
        if delta_n < 8.5:
            return "Gentoo"
        else:
            if (island == "Dream"):
                return "Chinstrap"
            else:
                return "Adelie"
    else:
        return "Adelie"
        #we will finish after class
```

## Part F

The following code will generate a guess for each penguin using the `decision_tree` function shown above. Modify the line that defines the `guesser` function according to the variables required by your decision tree. Then, run the code to create a new column called `Guess` containing the species guess for each penguin.

In [ ]:
```
# modify the first line, then run
guesser = lambda r: decision_tree(r["Island"], r["Body Mass (g)"], r["Sex"])
penguins["Guess"] = penguins.apply(guesser, axis = 1)
```

## Part G

Compute the accuracy of your decision tree -- what percentage of the time does your decision tree give you the right answer?

**Hint**: this is a one-liner.

In [ ]:
```
# your solution here
```

Soon, we'll learn how to use Python to automatically generate good decision trees without us needing to eyeball the data.