Working with Data

Data Management and Analysis

python™

Adapted from Data Carpentry's material:
https://datacarpentry.org/python-ecology-lesson/02-starting-with-data.html

LIBRARIES
COLORADO STATE UNIVERSITY

About the Data

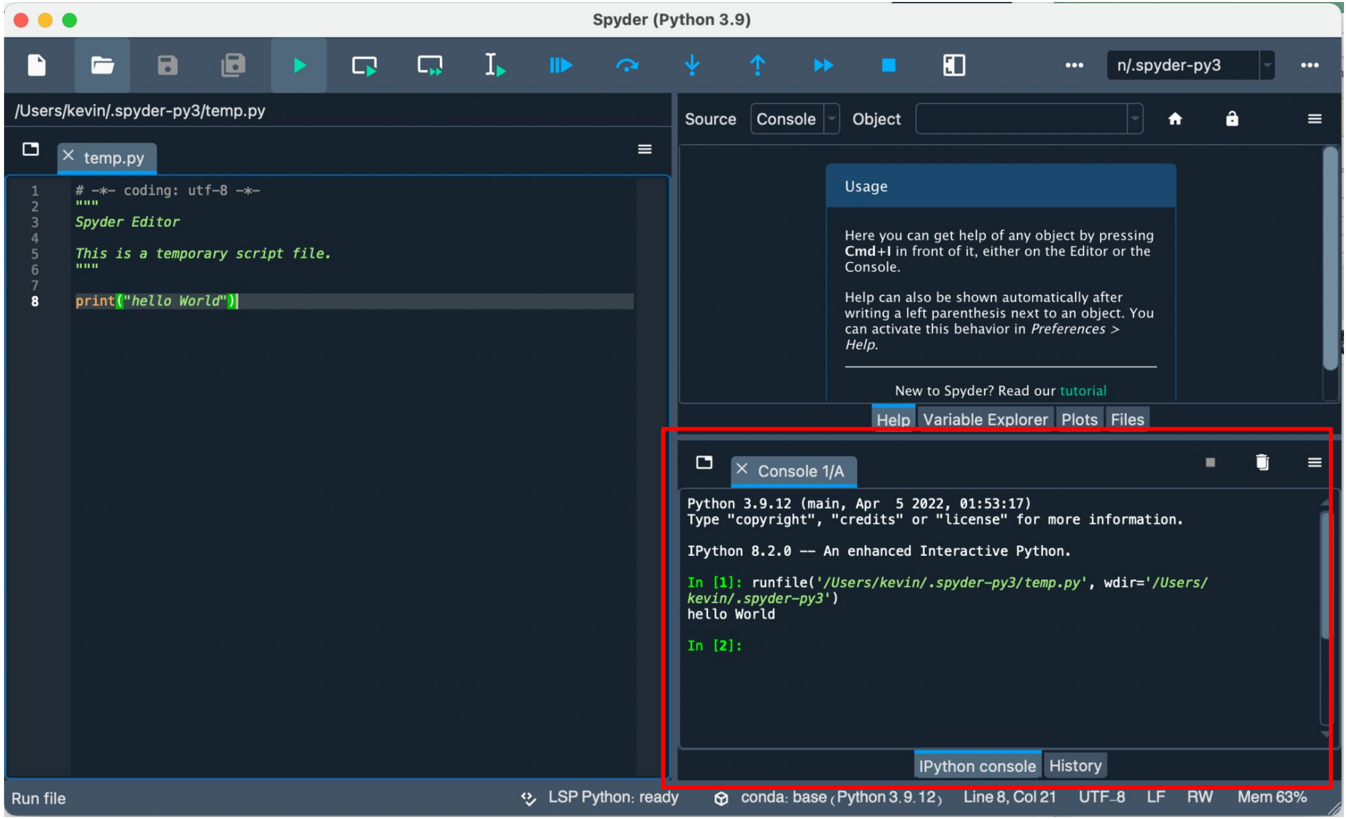
- The data used in the following lessons are observations of a small mammal community in southern Arizona.
- Running for almost 40 years, part of this project is looking at the effects of rodents and ants on the plant community.
- The rodents are sampled on a series of 24 plots, with different experimental manipulations controlling which rodents are allowed to access which plots.
- This is a real dataset that has been used in over 100 publications.

- The data is in 3 CSV files that can been linked together.
- We won't be covering the process of converted these data from a spreadsheet.
- It's important to note that in order to programmatically work with data, it should be organized and follow a predictable structure.

Exercise: Launch Spyder Check for Pandas library

Type the following into the console:

pip install pandas



```python
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

print("hello World")
```

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

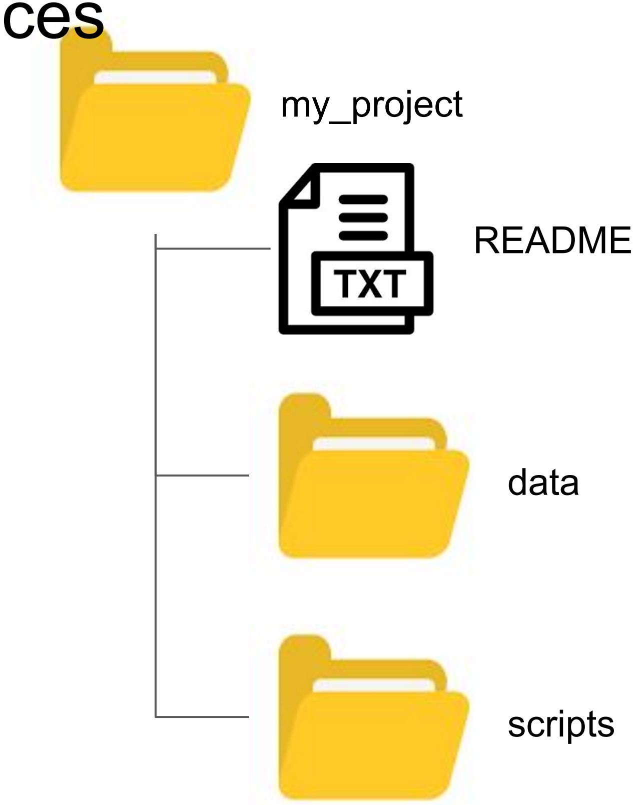
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
Python 3.9.12 (main, Apr  5 2022, 01:53:17)
Type "copyright", "credits" or "license" for more information.

IPython 8.2.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/kevin/.spyder-py3/temp.py', wdir='/Users/
kevin/.spyder-py3')
hello World

In [2]:
```

Recap: Research Project: Best Practices

- Create a project folder to work from
  - Add README to this folder
- Use folders to organize files in project folder
  - data/
    - use additional folders for raw and clean data
  - data_output/
    - to export processed results
  - documents/
    - outlines, drafts, other text
  - scripts/



my_project

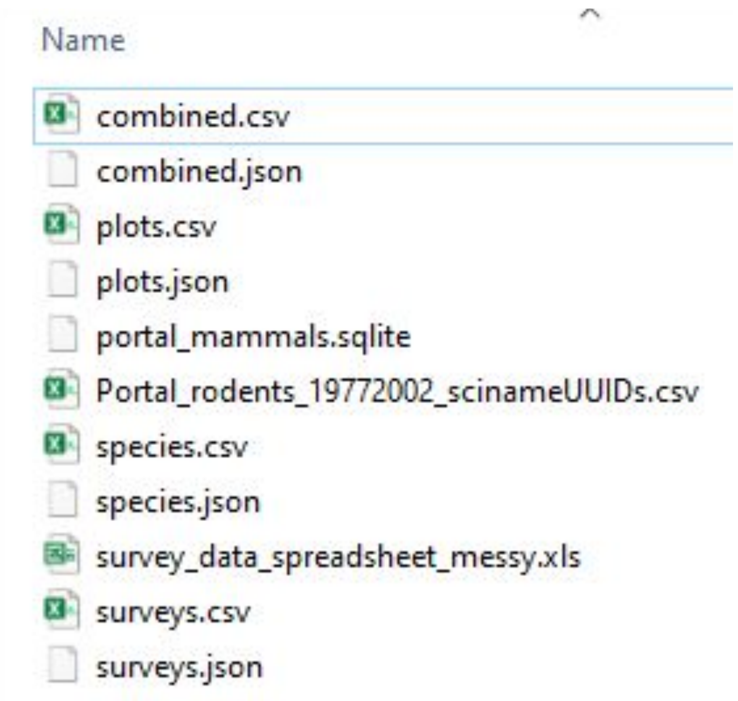
README

data

scripts

Exercise: Download the Data

- Go to: https://figshare.com/ndownloader/articles/1314459/versions/10
- Unzip the file
- Move all downloaded data into 'data' folder



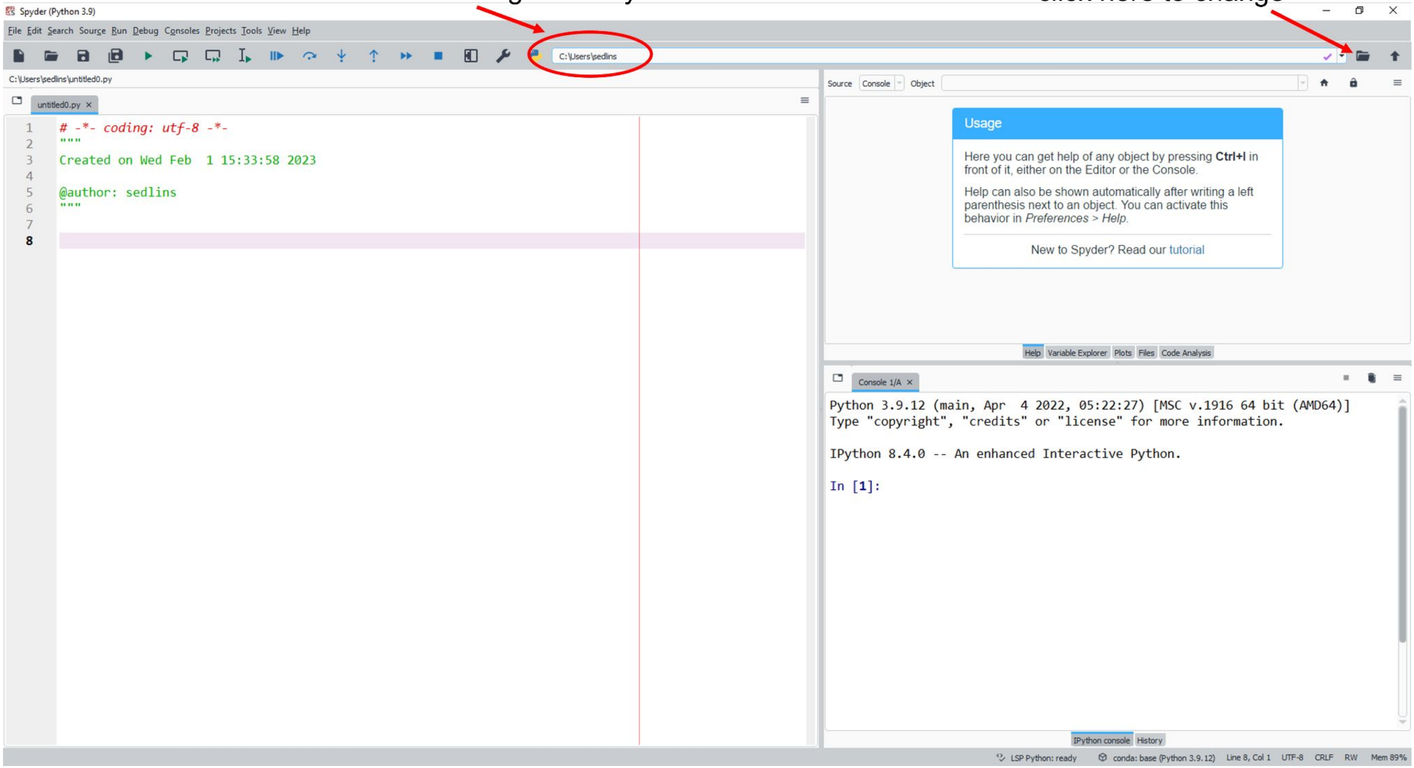
Name

combined.csv
combined.json
plots.csv
plots.json
portal_mammals.sqlite
Portal_rodents_19772002_scinameUUIDs.csv
species.csv
species.json
survey_data_spreadsheet_messy.xls
surveys.csv
surveys.json

Set Working Directory in Spyder

current working directory

click here to change



```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb  1 15:33:58 2023

@author: sedlins
"""

```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.4.0 -- An enhanced Interactive Python.

In [1]:
```

Loading Data

- Start with

```
import pandas as pd
```

Note: Python doesn’t load all of the libraries available to it by default.
Use syntax `import libraryName`
A nickname to shorten the command can be given using `as nickname`

- Then

```
# Note that pd.read_csv is used because we used import pandas as pd

pd.read_csv("../data/surveys.csv")
```

Note: The directory where your python code executes is important when accessing files.
The above example uses a relative path, meaning the script is looking from the current directory.
An absolute path would start with the drive letter (e.g C:/) on Windows or a '/' in Unix

Pandas and Python Data Types

Pandas Type	Python Type	Description
object	string	Will be assigned to your column if column has mixed types (numbers and strings).
int64	int	Numeric characters
float64	float	Numeric characters with decimals
datetime64, timedelta[ns]	N/A	Values meant to hold time data. Useful for time series experiments.

Exercise: Checking Pandas Data Type

- Check data type:
  ```
  import pandas as pd # Import package and name it as pd
  surveys_df = pd.read_csv("../data/surveys.csv") # load the dataframe
  type(surveys_df)
  surveys_df.dtypes
  surveys_df['sex'].dtype
  surveys_df['record_id'].dtype
  ```
- Convert data type
  ```
  surveys_df['record_id'] = surveys_df['record_id'].astype('float64')
  surveys_df['record_id'].dtype
  surveys_df['plot_id'].dtype
  surveys_df.plot_id.astype("float")
  surveys_df['plot_id'].dtype
  ```

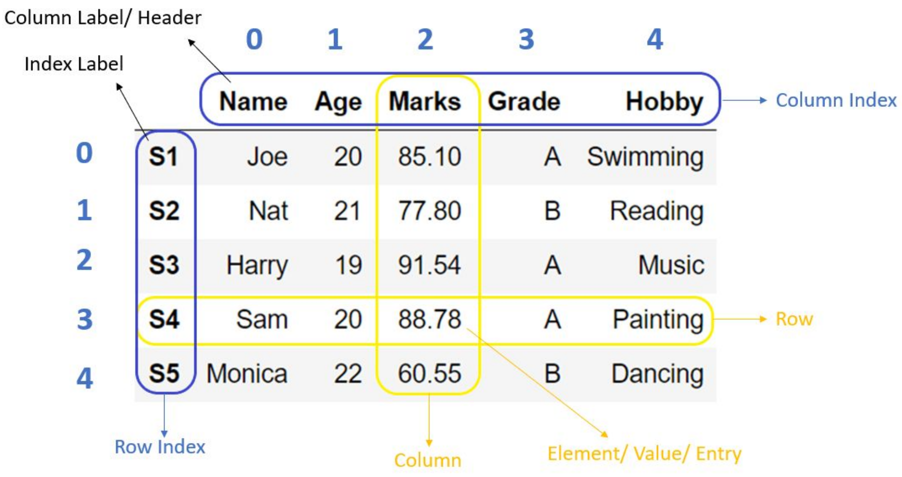
Exercise: Export to CSV

- Remove rows that contain missing data:
  - `surveys_df`
  - `df_na = surveys_df.dropna()`
  - `df_na`
  - `df_na.to_csv('../data/surveys_complete.csv', index= False)`

Exercise: Working with Data

Store data in variable `surveys_df`

```
import pandas as pd
surveys_df = pd.read_csv("data/surveys.csv")
```



		0	1	2	3	4
		Name	Age	Marks	Grade	Hobby
0	S1	Joe	20	85.10	A	Swimming
1	S2	Nat	21	77.80	B	Reading
2	S3	Harry	19	91.54	A	Music
3	S4	Sam	20	88.78	A	Painting
4	S5	Monica	22	60.55	B	Dancing

Determining object type and method responses

```
type(surveys_df)
surveys_df.head()  # The head() method displays the first several lines of a file.
surveys_df.columns  # Look at the column names
surveys_df.shape # Look at the number of rows and columns
```

Dataframes explained

- Rows = observations
- Cols = variables
  - All values in a column must be the same data type
- Data must be “rectangular” i.e. same number of rows/cols

Statistics From Data

```
surveys_df.columns # Look at the column names
pd.unique(surveys_df['species_id']) # get unique values from a column
```

Describe - to get all the stats

```
surveys_df['weight'].describe()
```

Or call each specifically

```
surveys_df['weight'].min() or ...max() or ...mean() or ...std() or ...count()
```

Groupby

- Summarize by one or more variables
- Creates a new dataframe

```
# Group data by sex
grouped_data = surveys_df.groupby('sex')
```

Exercise: Summary Data

1. How many recorded individuals are female `F` and how many are male `M`?
2. What happens when you group by two columns using the following syntax and then calculate mean values?
   - `grouped_data2 = surveys_df.groupby(['plot_id', 'sex'])`
   - `grouped_data2.mean()`
3. Summarize weight values for each site in your data.

   HINT: you can use the following syntax to only create summary statistics for one column in your data. `by_site['weight'].describe()`

Creating Summary Counts in Pandas

```
# Count the number of samples by species

species_counts = surveys_df.groupby('species_id')['record_id'].count()

species_counts

# also count just the rows that have the species “DO”

surveys_df.groupby('species_id')['record_id'].count()['DO']
```

Basic Plots with Pandas

```
#look at how many animals were captured in each site:
total_count =
surveys_df.groupby('plot_id')['record_id'].nunique()
# Let's plot it!
total_count.plot(kind='bar')
```

Exercise: Plotting with Pandas

1. Create a plot of average weight across all species per site.
2. Create a plot of total males versus total females for the entire dataset.

For more information on pandas plots, see pandas' documentation page on visualization.

Indexing, Slicing and Subsetting

Data Management and Analysis

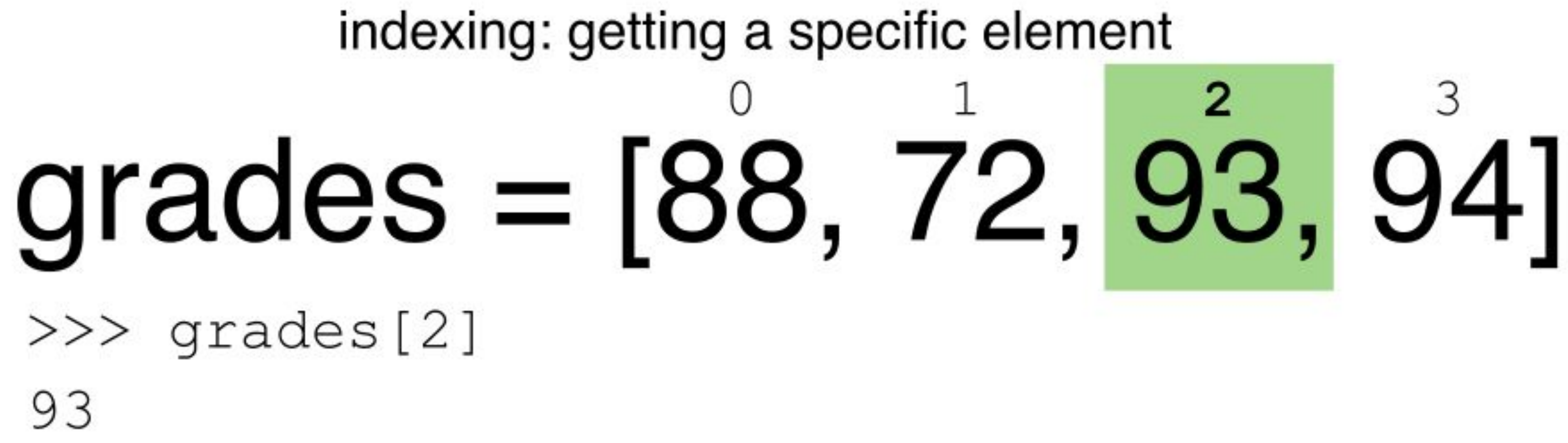
Adapted from Data Carpentry's material:
https://datacarpentry.org/python-ecology-lesson/03-index-slice-subset

LIBRARIES
COLORADO STATE UNIVERSITY

Data Selection

```
import pandas as pd
surveys_df = pd.read_csv("../data/surveys.csv")


# Method 1: select a 'subset' of the data using the column name
surveys_df['species_id']

# Method 2: use the column name as an 'attribute'; gives the same output
surveys_df.species_id

# Creates an object, surveys_species, that only contains the `species_id`
column
surveys_species = surveys_df['species_id']

# Select the species and plot columns from the DataFrame
surveys_df[['species_id', 'plot_id']]
```

Extracting Range based Subsets: Slicing



indexing: getting a specific element

```
grades = [88, 72, 93, 94]
>>> grades[2]
93
```

```
# Create a list of numbers:
a = [1, 2, 3, 4, 5]
```

Exercises: `a[0], a[5], a[len(a)]`

Slicing Subsets of Rows

'[ ]' operator selects a set of rows and/or columns from a DataFrame

- data[start:stop], start included, stops one step beyond end

```
# Select rows 0, 1, 2 (row 3 is not selected)
surveys_df[0:3]
```

```
# Select the first 5 rows (rows 0, 1, 2, 3, 4)
surveys_df[:5]
```

```
# Select the last element in the list

# (the slice starts at the last element, and ends at the end of the list)
surveys_df[-1:]
```

Copying Objects vs Referencing Objects

```
# Using the 'copy() method
true_copy_surveys_df = surveys_df.copy()

# Using the '=' operator
ref_surveys_df = surveys_df

# Assign the value `0` to the first three rows of data in the DataFrame
ref_surveys_df[0:3] = 0

# ref_surveys_df was created using the '=' operator
ref_surveys_df.head()

# surveys_df is the original dataframe
surveys_df.head()

# Reset surveys_df
surveys_df = pd.read_csv("../data/surveys.csv")
```

Slicing Subsets of Rows and Columns

- loc is primarily label based indexing
  - Integers may be used but they are interpreted as a label

```
#data.loc[[list (not range),[column ids] or ‘:’]
```

```
# Select all columns for rows of index values 0 and 10
surveys_df.loc[[0, 10], :]
```

- iloc is primarily integer based indexing

```
# data.iloc[row slicing, column slicing]
surveys_df.iloc[0:3, 1:4]
```

Exercise: Ranges Experimentation

What happens when you execute:

```
surveys_df[0:1]
surveys_df[:4]
surveys_df[:-1]
```

```
surveys_df.iloc[0:4, 1:4]
surveys_df.loc[0:4, 1:4]
```

Subsetting Data using Criteria

```
#select all rows that have a year value of 2002
surveys_df[surveys_df.year == 2002]
```

```
#select all rows that do not have a year value of 2002
surveys_df[surveys_df.year != 2002]
```

```
# using and '&'
surveys_df[(surveys_df.year >= 1980) & (surveys_df.year <= 1985)]
```

```
# use the isin command in Python to query a DataFrame based upon a list of values

surveys_df[surveys_df['species_id'].isin(['NL'])]
```

```
#'~' symbol in Python can be used to return the OPPOSITE of the selection that you specify
```

Exercise: Queries

1. Select a subset of rows in the `surveys_df` DataFrame that contain data from the year 1999 and that contain weight values less than or equal to 8. How many rows did you end up with?
2. Create a query that finds all rows with a weight value > or equal to 0.
3. Use the `isin` function to find all plots that contain 'NL' and 'DM' species in the “surveys” DataFrame. How many records contain these values?
4. Write a query that selects all rows with sex NOT equal to ‘M’ or ‘F’ in the “surveys” data.
   - Note: The `~` symbol in Python can be used to return the OPPOSITE of the selection that you specify in Python. It is equivalent to **is not in**.

Masks to identify a specific condition

Masks

- Used to locate a subset of values
- Can either exist or not
- For example, NaN, or “Not a Number” values
- Creates an output object with same shape as the original object,
  - but with a True or False value for each index location.

```
pd.isnull(surveys_df)
```

```
# To select just the rows with NaN values, we can use the 'any()' method
surveys_df[pd.isnull(surveys_df).any(axis=1)]
```

Exercise

1. What does this do?

   ```
   empty_weights = surveys_df[pd.isnull(surveys_df['weight'])]['weight']
   ```

   ```
   print(empty_weights)
   ```

2. Create a *new* DataFrame that only contains observations with sex values that are **not** female or male. Assign each sex value in the new DataFrame to the new value of ‘x’. Determine the number of null values in the subset.
3. Create a new DataFrame that contains only observations that are of sex male or female and where weight values are greater than 0.

   Bonus:
   Create a stacked bar plot of average weight by plot with male vs female values stacked for each plot

5 Minute Post Workshop Evaluation

https://forms.office.com/r/E1Yy7RNv3y