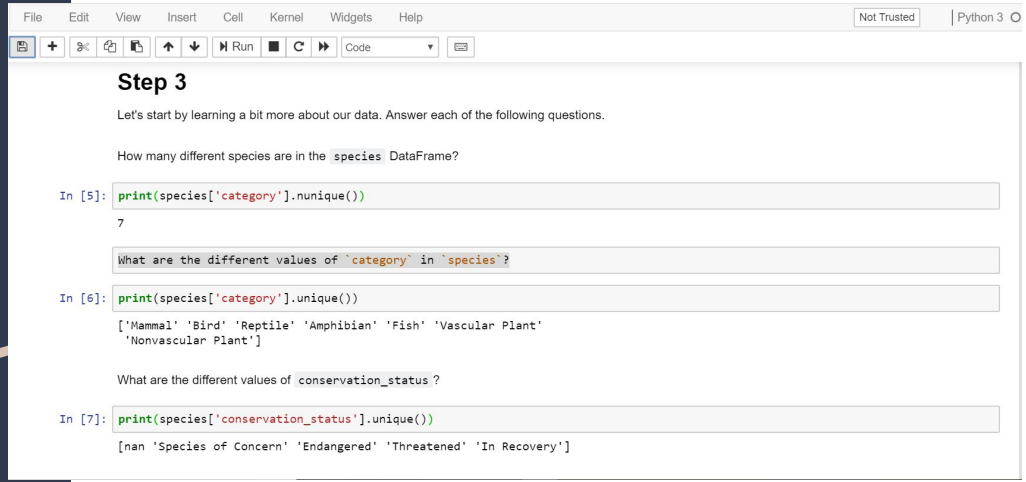
Capstone Option 2: Biodiversity for the National Parks

Adrian Nesta
Course 01.15.19

Importing and Analyzing species_info.csv

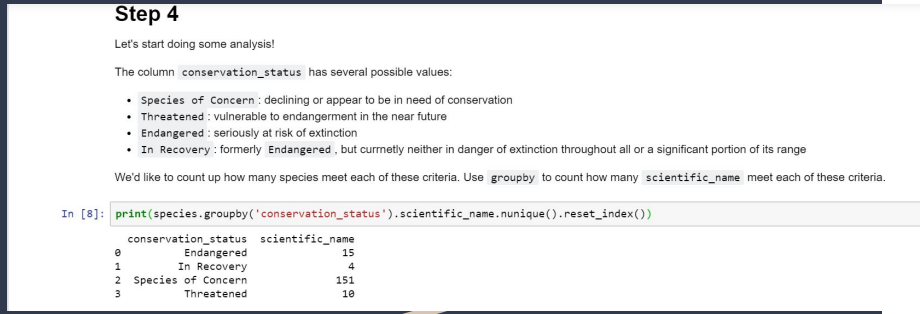
Using the pandas function pd.read_csv I imported species_info.csv to analyze the data and see what conclusions could be drawn about the different animal species from it.



I learned that there were 7 unique types of species: Mammal, Bird, Reptile, Amphibian, Fish, Vascular Plant, and Nonvascular Plant. I additionally discovered there were 4 types of Conservation status: Species of Concern, Endangered, Threatened and In Recovery

Analysis, cont.

I then created a dataframe using the .groupby method to display the amount of each species in each conservation category.



Step 4

Let's start doing some analysis!

The column `conservation_status` has several possible values:

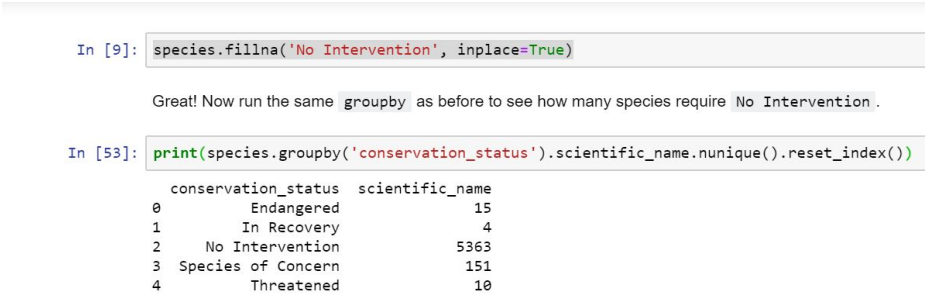
- `Species of Concern` : declining or appear to be in need of conservation
- `Threatened` : vulnerable to endangerment in the near future
- `Endangered` : seriously at risk of extinction
- `In Recovery` : formerly `Endangered` , but currnetly neither in danger of extinction throughout all or a significant portion of its range

We'd like to count up how many species meet each of these criteria. Use `groupby` to count how many `scientific_name` meet each of these criteria.

```
In [8]: print(species.groupby('conservation_status').scientific_name.nunique().reset_index())
```

	conservation_status	scientific_name
0	Endangered	15
1	In Recovery	4
2	Species of Concern	151
3	Threatened	10

Because the species that did not need protection were classified as NaN (null) in the conservation_status column. I used the .fillna method to create a row for them to provide a fuller picture of the data



```
In [9]: species.fillna('No Intervention', inplace=True)
```

Great! Now run the same `groupby` as before to see how many species require `No Intervention` .

```
In [53]: print(species.groupby('conservation_status').scientific_name.nunique().reset_index())
```

	conservation_status	scientific_name
0	Endangered	15
1	In Recovery	4
2	No Intervention	5363
3	Species of Concern	151
4	Threatened	10

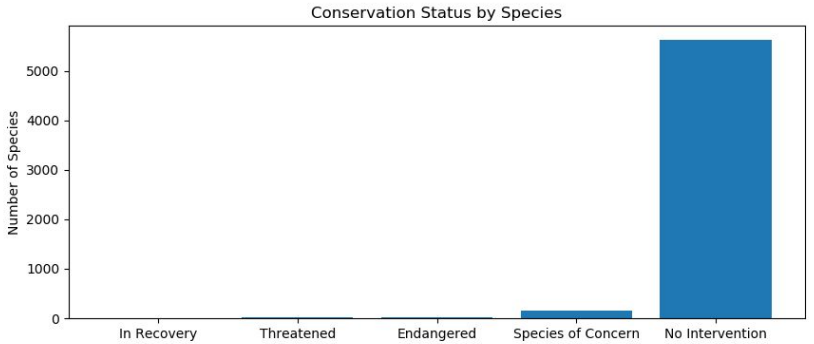
Bar Chart I

I used matplotlib to create a simple bar chart to visualize the conservation status of each species.

Now let's create a bar chart!

1. Start by creating a wide figure with `figsize=(10, 4)`
2. Start by creating an axes object called `ax` using `plt.subplot`.
3. Create a bar chart whose heights are equal to `scientific_name` column of `protection_counts`.
4. Create an x-tick for each of the bars.
5. Label each x-tick with the label from `conservation_status` in `protection_counts`
6. Label the y-axis `Number of Species`
7. Title the graph `Conservation Status by Species`
8. Plot the grap using `plt.show()`

In [12]:
```
plt.figure(figsize=(10,4))
ax = plt.subplot()
plt.bar(range(len(protection_counts)), protection_counts.scientific_name.values)
ax.set_xticks(range(len(protection_counts)))
ax.set_xticklabels(protection_counts.conservation_status.values)
plt.ylabel('Number of Species')
plt.title('Conservation Status by Species')
plt.show()
```



Are certain types of species in danger?

To view the data in the ‘is_protected’ column better. I created a pivot table in python using the .pivot method where data from ‘is_protected’ are the columns, the species categories are the rows, and the ‘scientific_name’ (amount) are the values.

It's going to be easier to view this data if we pivot it. Using `pivot`, rearange `category_counts` so that:

- `columns` is `is_protected`
- `index` is `category`
- `values` is `scientific_name`

Save your pivoted data to `category_pivot`. Remember to `reset_index()` at the end.

```
category_pivot = category_counts.pivot(columns='is_protected', index='category', values='scientific_name').reset_index()
```

Examine `category_pivot`.

```
print(category_pivot)
```

```
is_protected           category  False  True
0                     Amphibian     72     7
1                          Bird    413    75
2                          Fish    115    11
3                        Mammal    146    30
4             Nonvascular Plant    328     5
5                       Reptile     73     5
6                Vascular Plant   4216    46
```

To answer the question if one of the species observed in the data has statistically significant higher rate of endangerment, I first added two new columns to the data frame. The first was a ‘is_protected’ column.

In [14]:
```
species['is_protected'] = species['conservation_status'] != 'No Intervention'
```

```
Let's group by *both* `category` and `is_protected`.  Save your results to `category_counts`.
```

In [20]:
```
category_counts = species.groupby(['category', 'is_protected'])\
                         .scientific_name.nunique().reset_index()
```

Examine `category_counts` using `head()`.

In [22]:
```
print(category_counts.head())
```

```
    category  is_protected  scientific_name
0  Amphibian         False               72
1  Amphibian          True                7
2       Bird         False              413
3       Bird          True               75
4       Fish         False              115
```

Danger, cont.

After renaming the ‘false’ and ‘true’ columns to ‘not_protected’ and ‘protected’, respectively. I created a new column in the dataframe with the percent of each species that are currently under protection (i.e. in some kind of danger)

```
In [29]: category_pivot['percent_protected'] =
         category_pivot['protected'] / (category_pivot['not_protected'] + category_pivot['protected'])
```

Examine `category_pivot`.

```
In [30]: print(category_pivot)
```

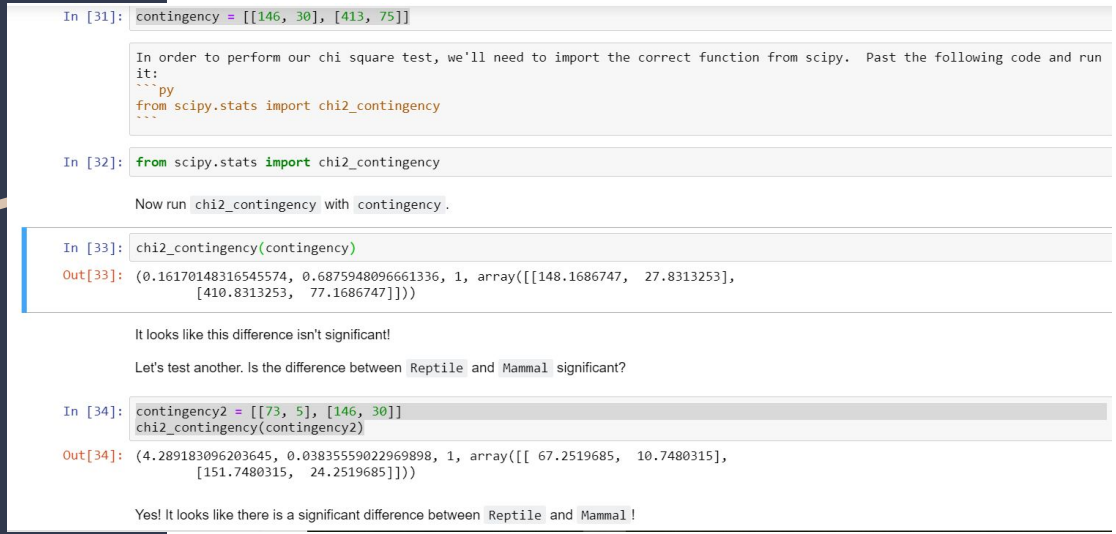
	category	not_protected	protected	percent_protected
0	Amphibian	72	7	0.088608
1	Bird	413	75	0.153689
2	Fish	115	11	0.087302
3	Mammal	146	30	0.170455
4	Nonvascular Plant	328	5	0.015015
5	Reptile	73	5	0.064103
6	Vascular Plant	4216	46	0.010793

Statistical Significance

The test revealed that difference between protection rates for Mammals and Birds was not statistically significant, but the difference between Mammals and Reptiles was!

I would recommend that the parks look into what might be causing the unusually high endagernment rate for mammals.

To compare the protection level between multiple species in the data to determine if any specific species has a statistically significant higher level of needed protection I imported the chi2 contingency test from scipy.stats.



In [31]:
```
contingency = [[146, 30], [413, 75]]
```

In order to perform our chi square test, we'll need to import the correct function from scipy. Past the following code and run it:

```py
from scipy.stats import chi2_contingency
```

In [32]:
```
from scipy.stats import chi2_contingency
```

Now run `chi2_contingency` with `contingency`.

In [33]:
```
chi2_contingency(contingency)
```

Out[33]:
```
(0.16170148316545574, 0.6875948096661336, 1, array([[148.1686747,  27.8313253],
       [410.8313253,  77.1686747]]))
```

It looks like this difference isn't significant!

Let's test another. Is the difference between `Reptile` and `Mammal` significant?

In [34]:
```
contingency2 = [[73, 5], [146, 30]]
chi2_contingency(contingency2)
```

Out[34]:
```
(4.289183096203645, 0.03835559022969898, 1, array([[ 67.2519685,  10.7480315],
       [151.7480315,  24.2519685]]))
```

Yes! It looks like there is a significant difference between `Reptile` and `Mammal`!

Sheep Observations

Indexing the old dataframe by the new column that filters out new sheep reveals that it erroneously picked up some extra data

```
In [36]: species['is_sheep'] = species.common_names.apply(lambda x: 'Sheep' in x)
         species.head()
```

Out[36]:

	category	scientific_name	common_names	conservation_status	is_protected	is_sheep
0	Mammal	Clethrionomys gapperi gapperi	Gapper's Red-Backed Vole	No Intervention	False	False
1	Mammal	Bos bison	American Bison, Bison	No Intervention	False	False
2	Mammal	Bos taurus	Aurochs, Aurochs, Domestic Cattle (Feral), Dom...	No Intervention	False	False
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True
4	Mammal	Cervus elaphus	Wapiti Or Elk	No Intervention	False	False

Select the rows of `species` where `is_sheep` is `True` and examine the results.

```
In [54]: species[species.is_sheep]
```

Out[54]:

	category	scientific_name	common_names	conservation_status	is_protected	is_sheep
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False	True
1139	Vascular Plant	Rumex acetosella	Sheep Sorrel, Sheep Sorrell	No Intervention	False	True
2233	Vascular Plant	Festuca filiformis	Fineleaf Sheep Fescue	No Intervention	False	True
3014	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True

After importing and examining the other file of data, observations.csv, I used a lambda function to add a new column to the dataframe that determined if the observation was a sheep

```
observations = pd.read_csv('observations.csv')
observations.head()
```

	scientific_name	park_name	observations
0	Vicia benghalensis	Great Smoky Mountains National Park	68
1	Neovison vison	Great Smoky Mountains National Park	77
2	Prunus subcordata	Yosemite National Park	138
3	Abutilon theophrasti	Bryce National Park	84
4	Githopsis specularioides	Great Smoky Mountains National Park	85

Further filtering and merging

To weed out the extra plant species data from the sheep dataframe I filtered on the species dataframe to include only ‘Mammals’. I then merged the ‘observations’ dataframe with the ‘sheep’ only dataframe to provide a more robust set of data.

In [44]:
```
sheep_species = species[(species.is_sheep) & (species.category == 'Mammal')]
```

```
Now merge `sheep_species` with `observations` to get a DataFrame with observations of sheep.  Save this DataFrame as
`sheep_observations`.
```

In [45]:
```
sheep_observations = observations.merge(sheep_species)
sheep_observations
```

Out[45]:

	scientific_name	park_name	observations	category	common_names	conservation_status	is_protected	is_sheep
0	Ovis canadensis	Yellowstone National Park	219	Mammal	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
1	Ovis canadensis	Bryce National Park	109	Mammal	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
2	Ovis canadensis	Yosemite National Park	117	Mammal	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
3	Ovis canadensis	Great Smoky Mountains National Park	48	Mammal	Bighorn Sheep, Bighorn Sheep	Species of Concern	True	True
4	Ovis canadensis sierrae	Yellowstone National Park	67	Mammal	Sierra Nevada Bighorn Sheep	Endangered	True	True
5	Ovis canadensis sierrae	Yosemite National Park	39	Mammal	Sierra Nevada Bighorn Sheep	Endangered	True	True
6	Ovis canadensis sierrae	Bryce National Park	22	Mammal	Sierra Nevada Bighorn Sheep	Endangered	True	True

Sheep by Park

Using pandas to create an ‘Observation by Park’ dataframe with the sum total of sheep from each park. I then visualized that pared down dataframe with another bar plot from matplotlib.

In [47]:
```
obs_by_park = sheep_observations.groupby('park_name').observations.sum().reset_index()
obs_by_park
```

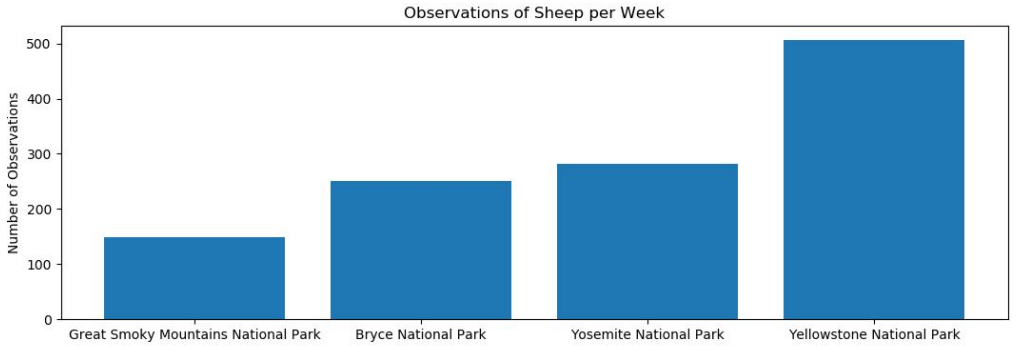
Out[47]:

	park_name	observations
0	Bryce National Park	250
1	Great Smoky Mountains National Park	149
2	Yellowstone National Park	507
3	Yosemite National Park	282

Create a bar chart showing the different number of observations per week at each park.

1. Start by creating a wide figure with `figsize=(16, 4)`
2. Start by creating an axes object called `ax` using `plt.subplot`.
3. Create a bar chart whose heights are equal to `observations` column of `obs_by_park`.
4. Create an x-tick for each of the bars.
5. Label each x-tick with the label from `park_name` in `obs_by_park`
6. Label the y-axis `Number of Observations`
7. Title the graph `Observations of Sheep per Week`
8. Plot the grap using `plt.show()`

In [49]:
```
plt.figure(figsize=(16,4))
ax = plt.subplot()
plt.bar(range(len(obs_by_park)), obs_by_park.observations.values)
ax.set_xticks(range(len(obs_by_park.park_name)))
ax.set_xticklabels(obs_by_park.park_name.values)
plt.ylabel('Number of Observations')
plt.title('Observations of Sheep per Week')
plt.show()
```



How many weeks?

Our scientists know that 15% of sheep at Bryce National Park have foot and mouth disease. Park rangers at Yellowstone National Park have been running a program to reduce the rate of foot and mouth disease at that park. The scientists want to test whether or not this program is working. They want to be able to detect reductions of at least 5 percentage points. For instance, if 10% of sheep in Yellowstone have foot and mouth disease, they'd like to be able to know thi with confidence.

Use Codecademy's sample size calculator to calculate the number of sheep that they would need to observe from each park. Use the default level of significance (90%).

Remember that "Minimum Detectable Effect" is a percent of the baseline.

```
minimum_detectable_effect = 100 * (0.05 / 0.15)
print(minimum_detectable_effect)
```

```
33.333333333333336
```

How many weeks would you need to observe sheep at Bryce National Park in order to observe enough sheep? How many weeks would you need to observe at Yellowstone National Park to observe enough sheep?

```
#Needs to see 870 sheep. One week of sheep observations at Bryce National Park yielded 250. Will need about 4 weeks
(870/250)
#One week of sheep observations at Yellowstone produced 507. Will need about 2 weeks
(870/507)
```

```
1.7159763313609468
```

To determine if the new program that Yellowstone implemented to reduce foot and mouth disease in the park's sheep will be effective, we need to know how many weeks of sheep observations will be needed to detect a 5% change from the 15% baseline. Using the Codecademy tool, calculating the minimum detectable effect (33.3%), and a default 90% confidence, I determined that it will take about 4 weeks in Bryce National Park to reach the necessary sheep observations and 2 weeks in Yellowstone.