Codecademy Pro Intensive: Introduction to Data Analysis Capstone Project

Biodiversity for the National Parks

By Thomas Flanney

Purpose

- To perform data analysis on the conservation statuses of animals and plants in U.S. National Parks
- Use the skills learned in the Pro Intensive course to analyze datasets produced by the National Park Service, and present findings in a clear manner

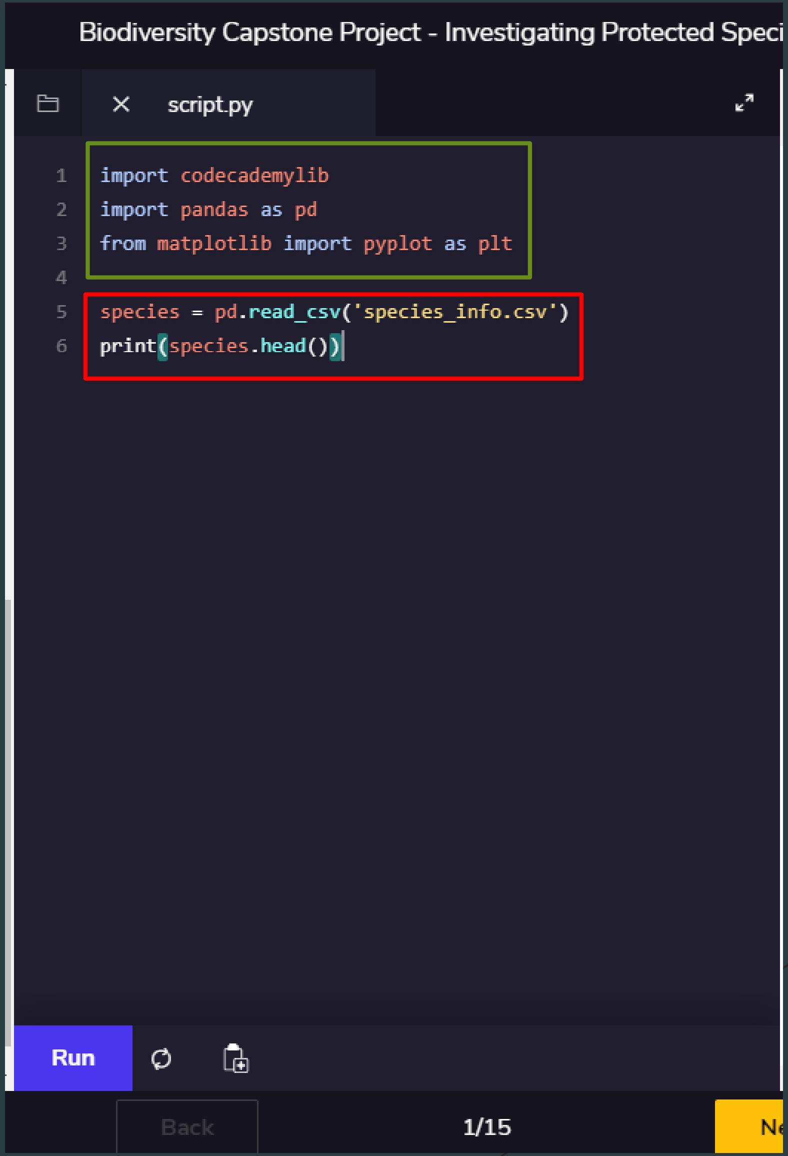
https://www.nps.gov/yell/learn/nature/bearreact.htm

Task 1

Investigate patterns found in endangered species in National Parks

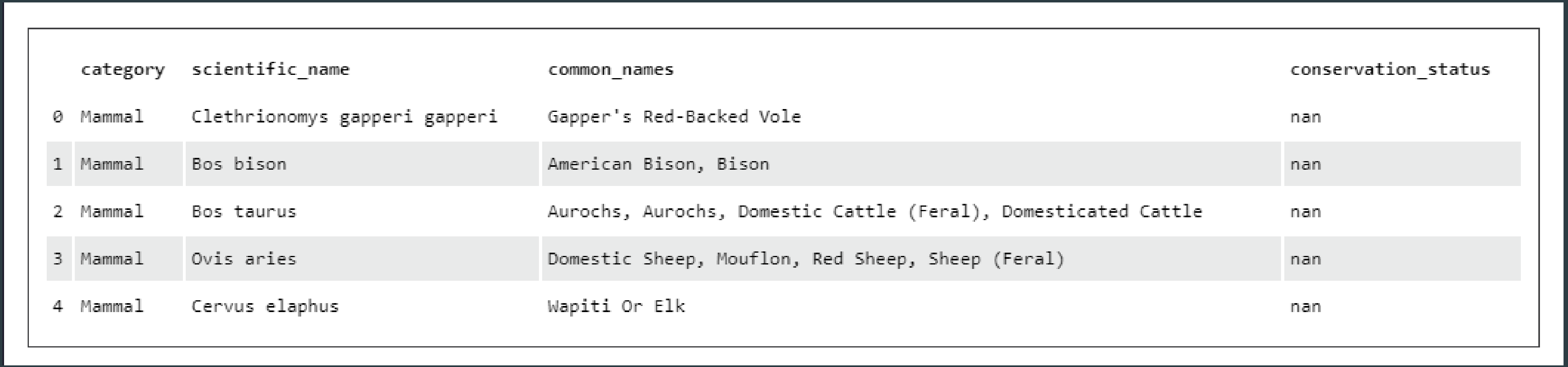
Step 1: See information contained in species CSV file

- First, import Pandas and Matplotlib into Python for future steps
- Next, investigate the first few rows of the DataFrame to see what information it contains
  - print(species.head()) to look at first five rows


Biodiversity Capstone Project - Investigating Protected Speci
script.py
Run
Back
1/15
Ne

```python
import codecademylib
import pandas as pd
from matplotlib import pyplot as plt

species = pd.read_csv('species_info.csv')
print(species.head())
```

Step 1 (continued): DataFrame's first five rows

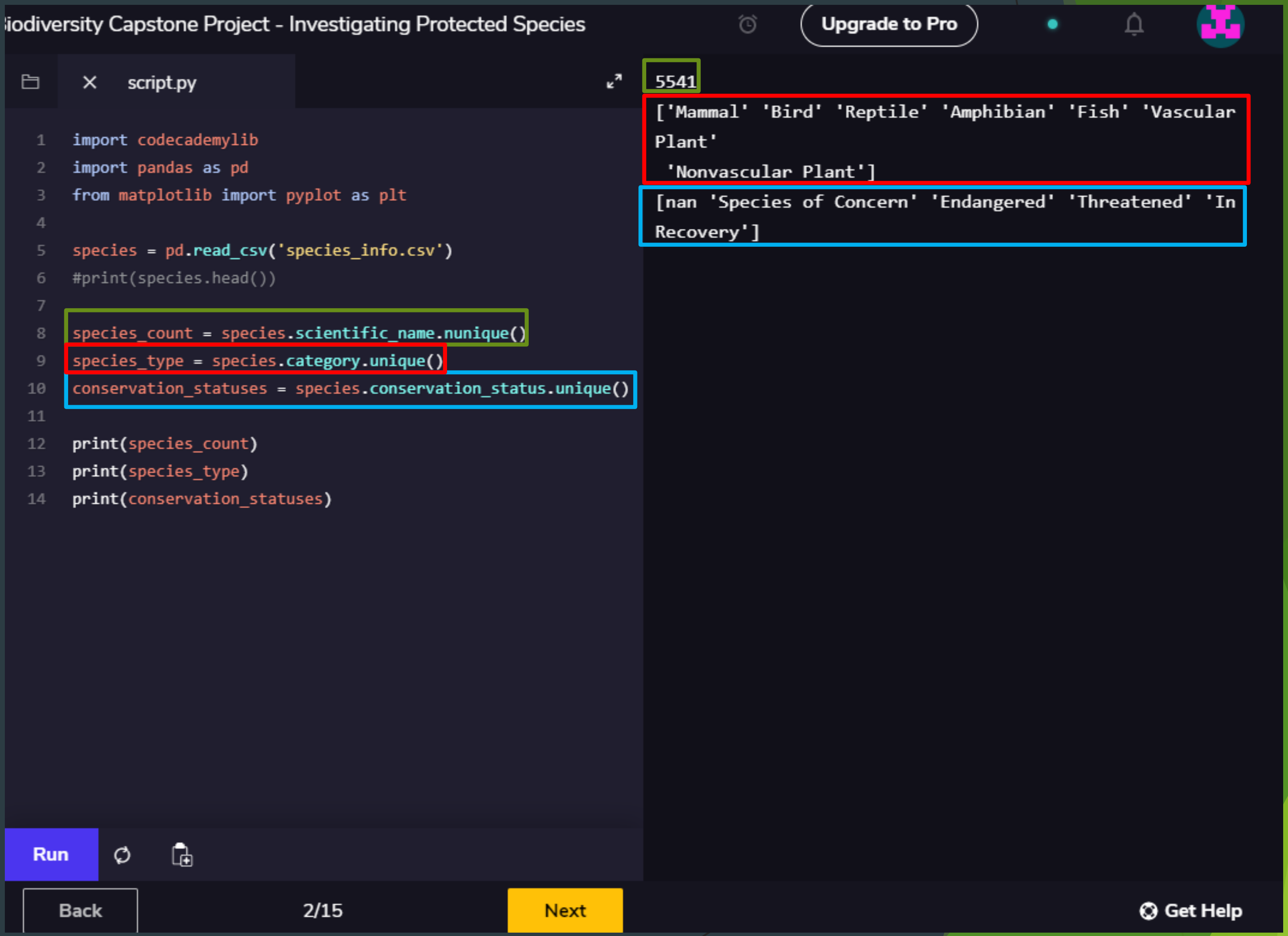


	category	scientific_name	common_names	conservation_status
0	Mammal	Clethrionomys gapperi gapperi	Gapper's Red-Backed Vole	nan
1	Mammal	Bos bison	American Bison, Bison	nan
2	Mammal	Bos taurus	Aurochs, Aurochs, Domestic Cattle (Feral), Domesticated Cattle	nan
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	nan
4	Mammal	Cervus elaphus	Wapiti Or Elk	nan

DataFrame columns included the category of the animal or plant, the scientific name, the common name, and the conservation status

Step 1 (continued): calculating DataFrame column stats

- Species_count: counts number of unique rows of DataFrame based on scientific name
- Species_type: counts list of unique names based on category
- Conservation_statuses: counts list of unique names based on conservation status



iodiversity Capstone Project - Investigating Protected Species
Upgrade to Pro
script.py
Run
Back
2/15
Next
Get Help

```
import codecademylib
import pandas as pd
from matplotlib import pyplot as plt

species = pd.read_csv('species_info.csv')
#print(species.head())

species_count = species.scientific_name.nunique()
species_type = species.category.unique()
conservation_statuses = species.conservation_status.unique()

print(species_count)
print(species_type)
print(conservation_statuses)
```

```
5541
['Mammal' 'Bird' 'Reptile' 'Amphibian' 'Fish' 'Vascular Plant'
 'Nonvascular Plant']
[nan 'Species of Concern' 'Endangered' 'Threatened' 'In Recovery']
```

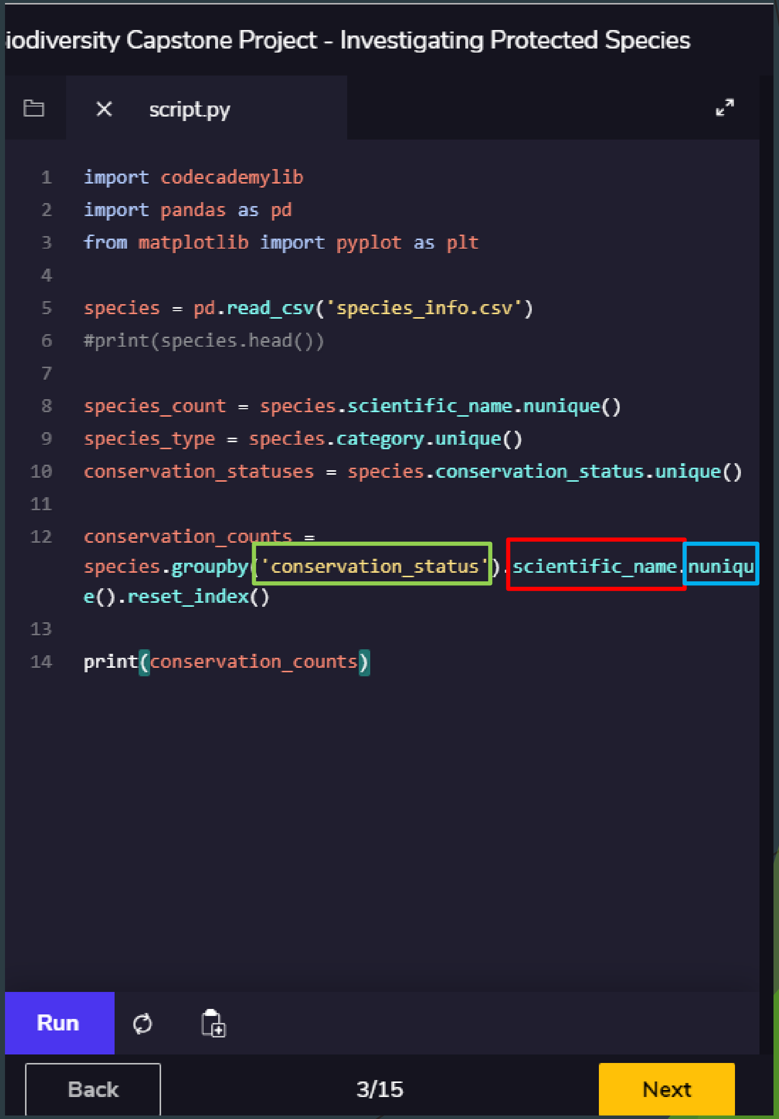
Conservation Status Meanings

- ‘Species of Concern’ = declining population or in need of conversation
- ‘Threatened’ = vulnerable to endangerment
- ‘Endangered’ = seriously at risk of extinction
- ‘In Recovery’ = formerly endangered but no longer in danger of extinction

Step 2: Obtain counts based on conservation statuses

Using aggregate functions

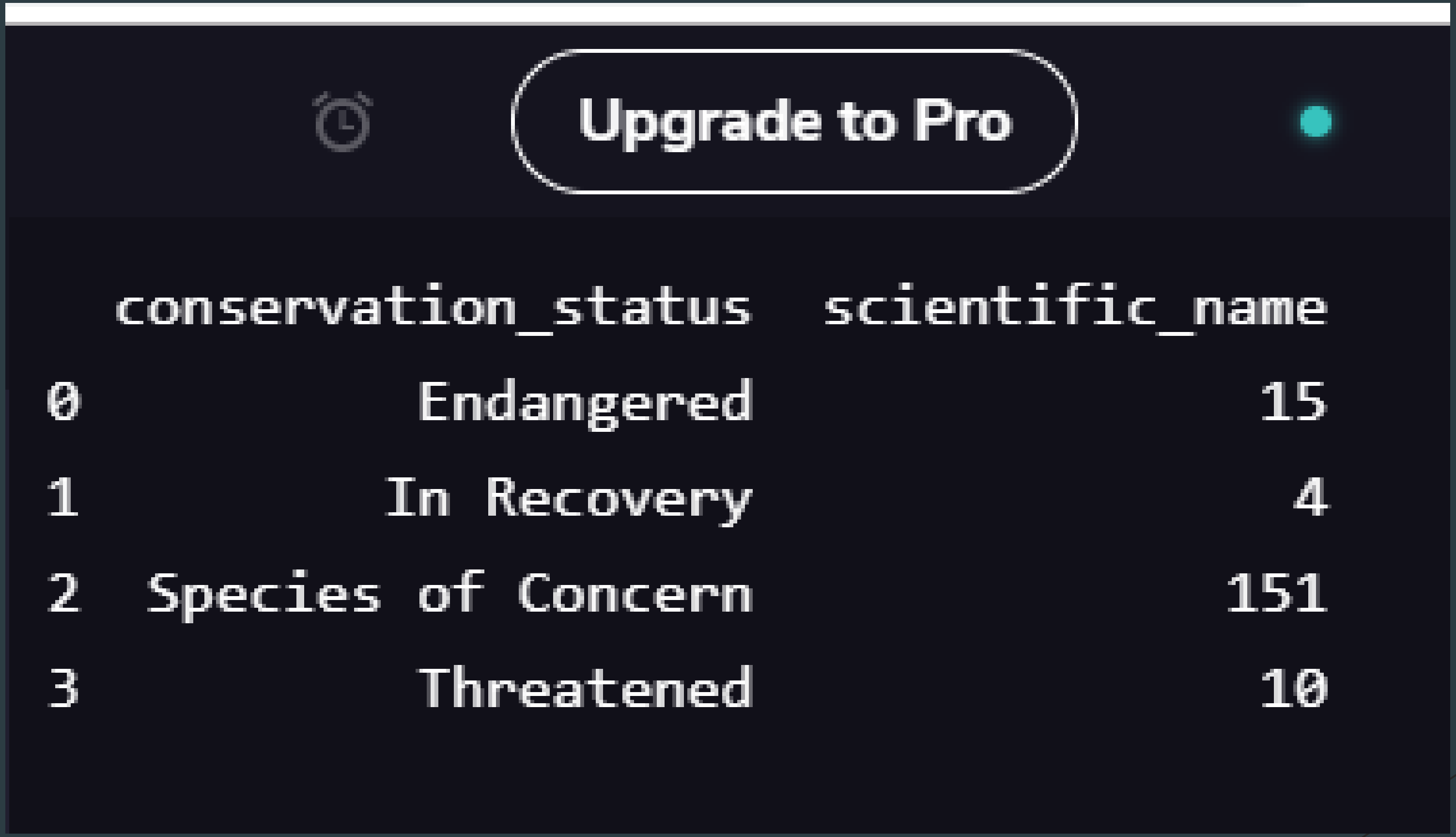
- Group by two columns
  - want to group by conservation status column
  - Want to perform measurement on scientific name (ensures no duplicates)
  - Perform nunique calculation (ensures getting a number for each scientific name)


iodiversity Capstone Project - Investigating Protected Species

script.py

```
import codecademylib
import pandas as pd
from matplotlib import pyplot as plt

species = pd.read_csv('species_info.csv')
#print(species.head())

species_count = species.scientific_name.nunique()
species_type = species.category.unique()
conservation_statuses = species.conservation_status.unique()

conservation_counts =
species.groupby('conservation_status').scientific_name.nuniqu
e().reset_index()

print(conservation_counts)
```

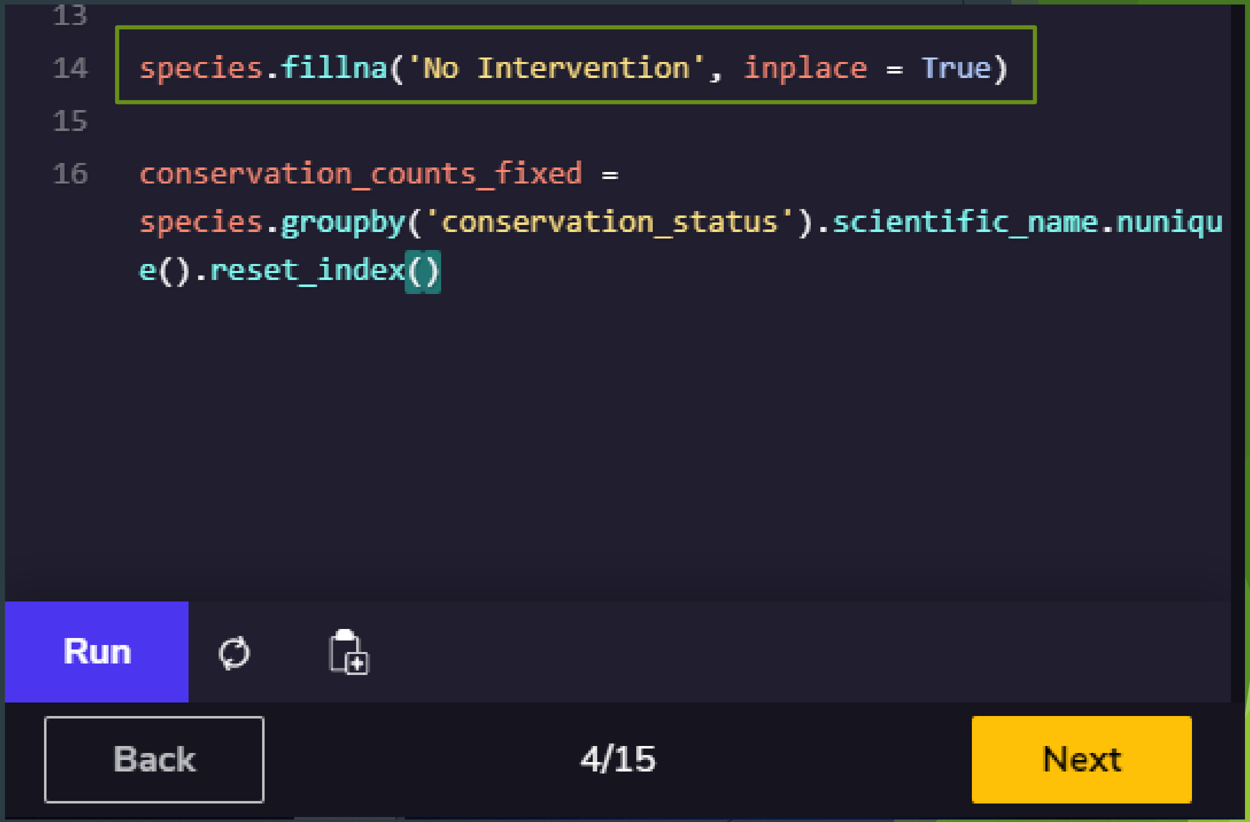
Step 2 (continued): Conservation Count Results



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

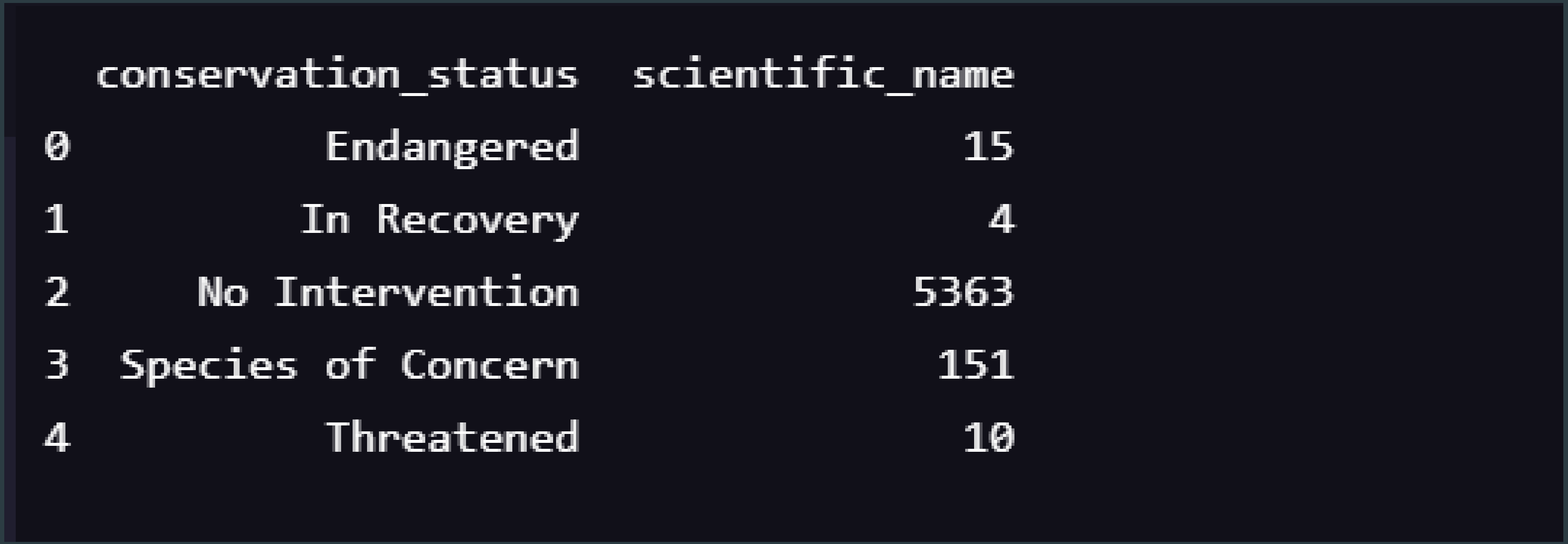
Step 2 (continued): Including ‘No Intervention’ to replace ‘nan’

- Species.fillna
  - Function that will replace all values labelled as ‘nan’ (Not A Number)
  - Two inputs
    - ‘No Intervention’ - what we want in place of nan
    - ‘inplace = True’ - allows the function to replace ‘nan’



```
species.fillna('No Intervention', inplace = True)

conservation_counts_fixed =
species.groupby('conservation_status').scientific_name.nunique().reset_index()
```

Step 2 (continued): Fixed Conservation Count Results

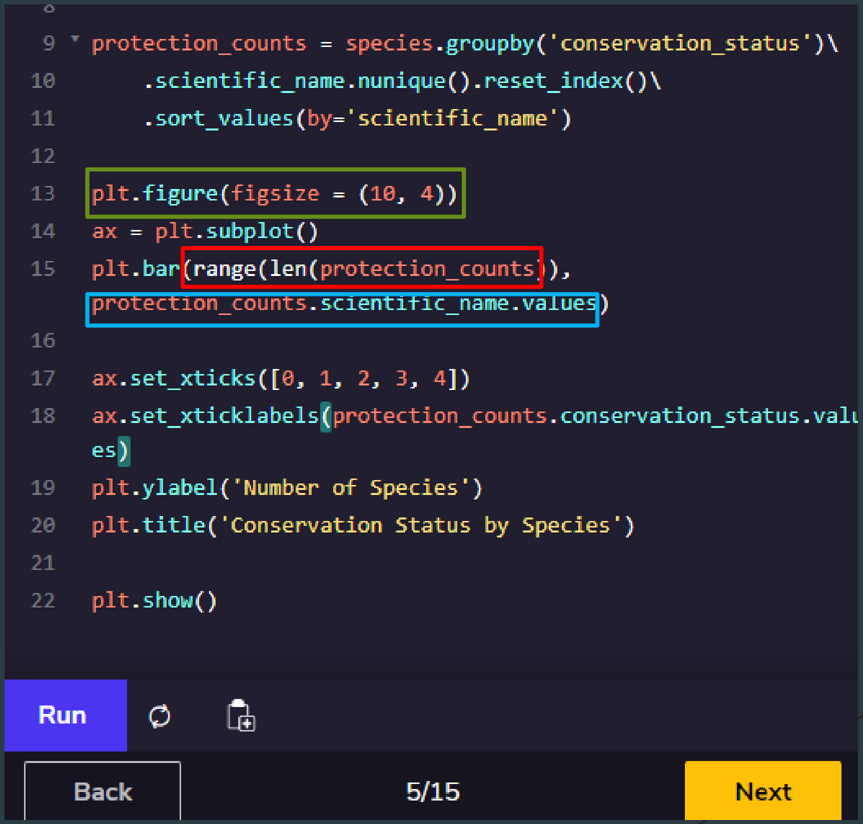


```
  conservation_status  scientific_name
0          Endangered               15
1         In Recovery                4
2     No Intervention             5363
3  Species of Concern              151
4          Threatened               10
```

Step 2 (continued): Bar Graph of Conservation Status by Species

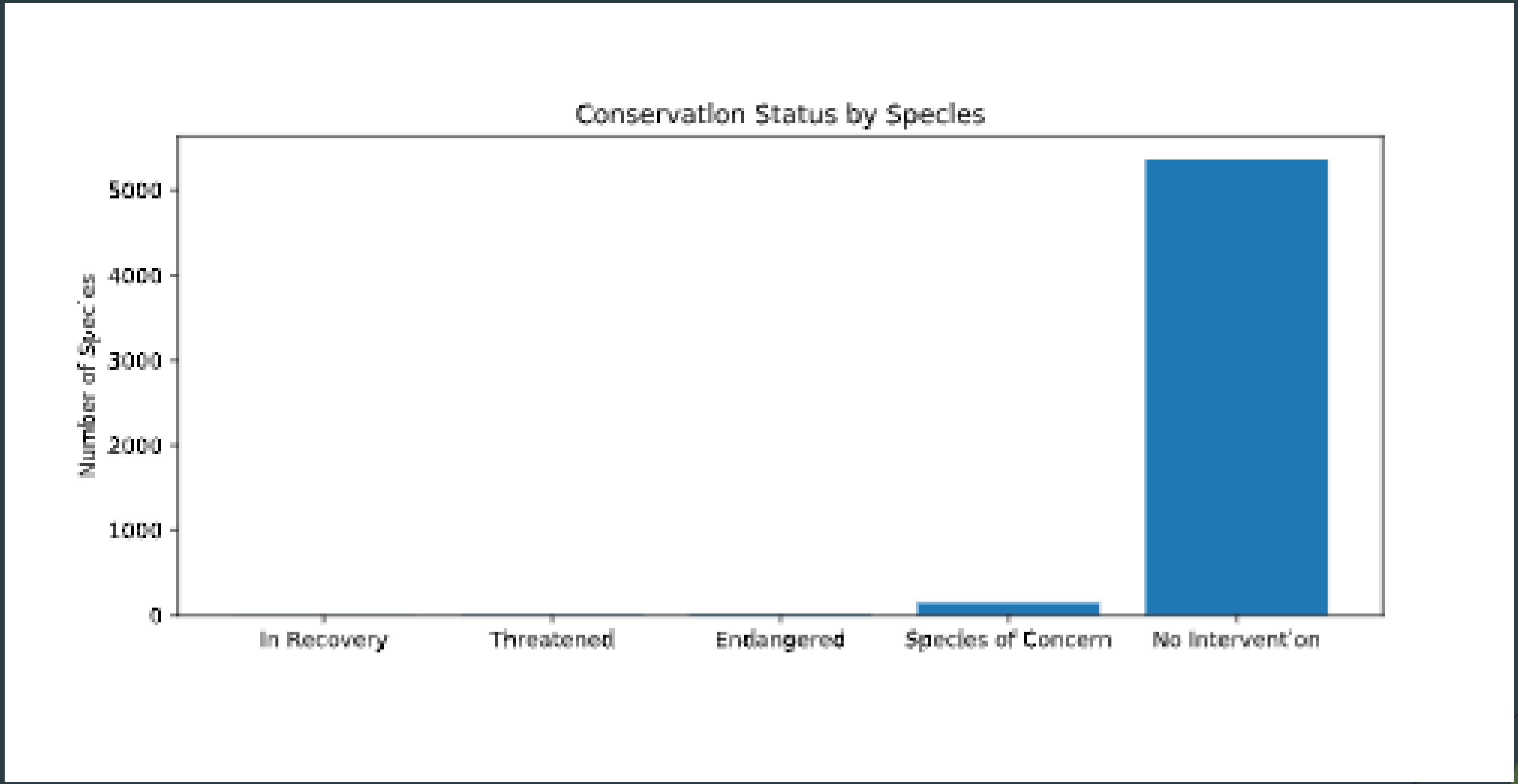
- Protection_counts DataFrame = further organization by scientific name

- Creating the bar graph
  - Figure size is 10 in. by 4 in.
  - X-axis values for graph is the range of length of protection counts (5)
  - Y-axis values for graph is values of protection counts by scientific name



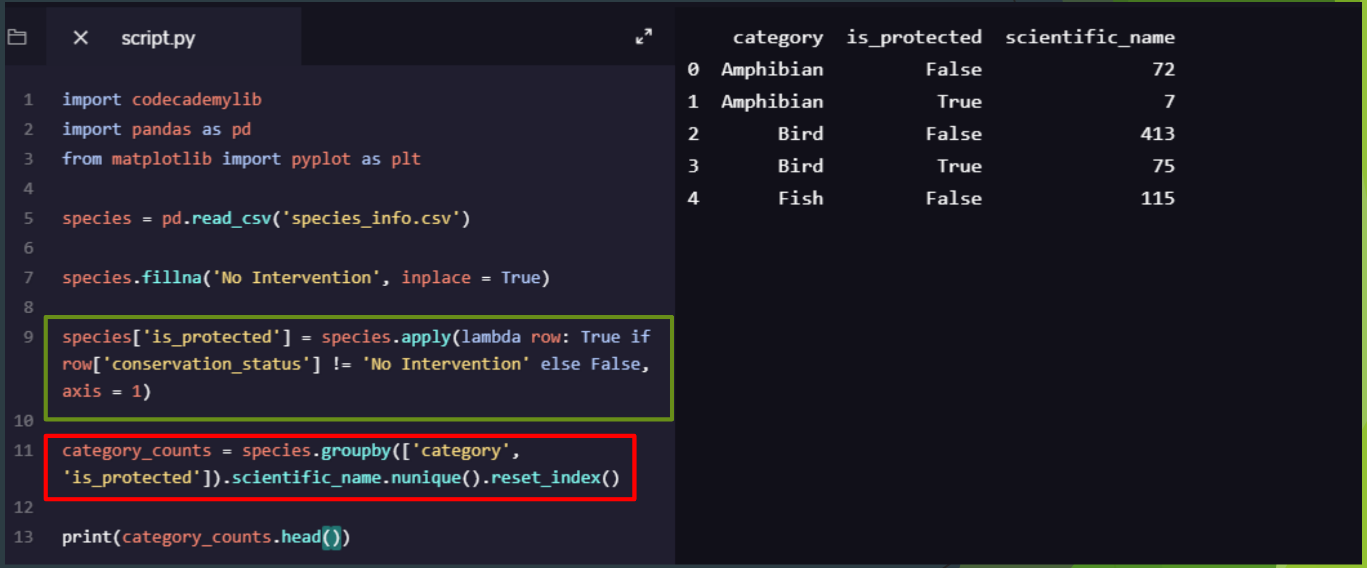
```
protection_counts = species.groupby('conservation_status')\
    .scientific_name.nunique().reset_index()\
    .sort_values(by='scientific_name')

plt.figure(figsize = (10, 4))
ax = plt.subplot()
plt.bar(range(len(protection_counts)),
protection_counts.scientific_name.values)

ax.set_xticks([0, 1, 2, 3, 4])
ax.set_xticklabels(protection_counts.conservation_status.valu
es)
plt.ylabel('Number of Species')
plt.title('Conservation Status by Species')

plt.show()
```

Step 2 (continued): Bar Graph Results



Step 3: Observe Whether Certain Species More Threatened Than Others

- Species['is_protected']
  - Creates new column called 'is_protected'
  - Includes lambda function that determines if conservation status of row is 'No Intervention'
- Category_counts
  - Groups by 'category', 'is_protected' and performs nunique measurement on 'scientific_name'

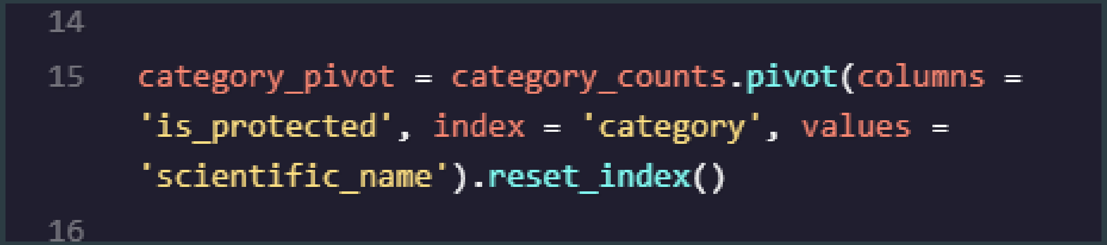


script.py

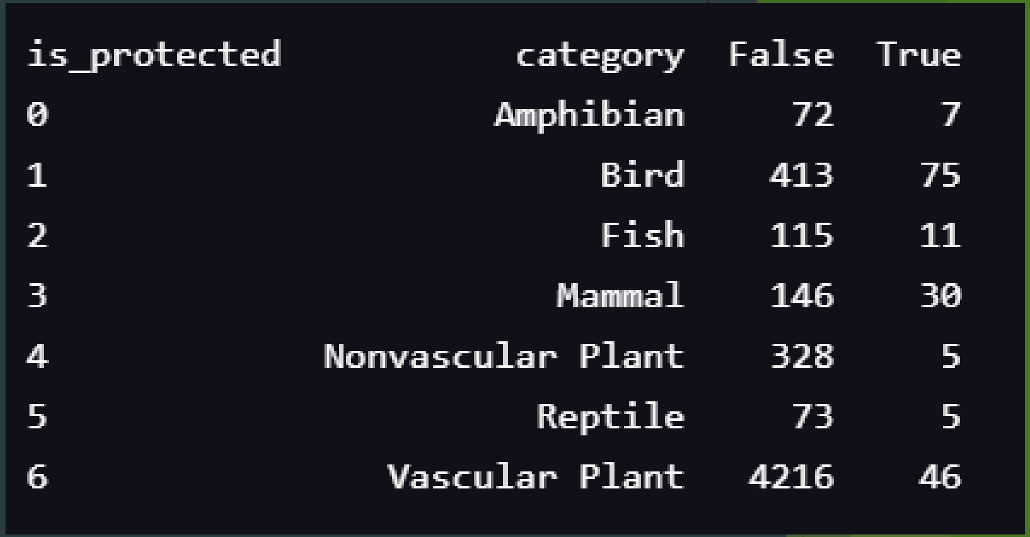
```python
import codecademylib
import pandas as pd
from matplotlib import pyplot as plt

species = pd.read_csv('species_info.csv')

species.fillna('No Intervention', inplace = True)

species['is_protected'] = species.apply(lambda row: True if
row['conservation_status'] != 'No Intervention' else False,
axis = 1)

category_counts = species.groupby(['category',
'is_protected']).scientific_name.nunique().reset_index()

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
```

Step 3 (continued): Pivot Table



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



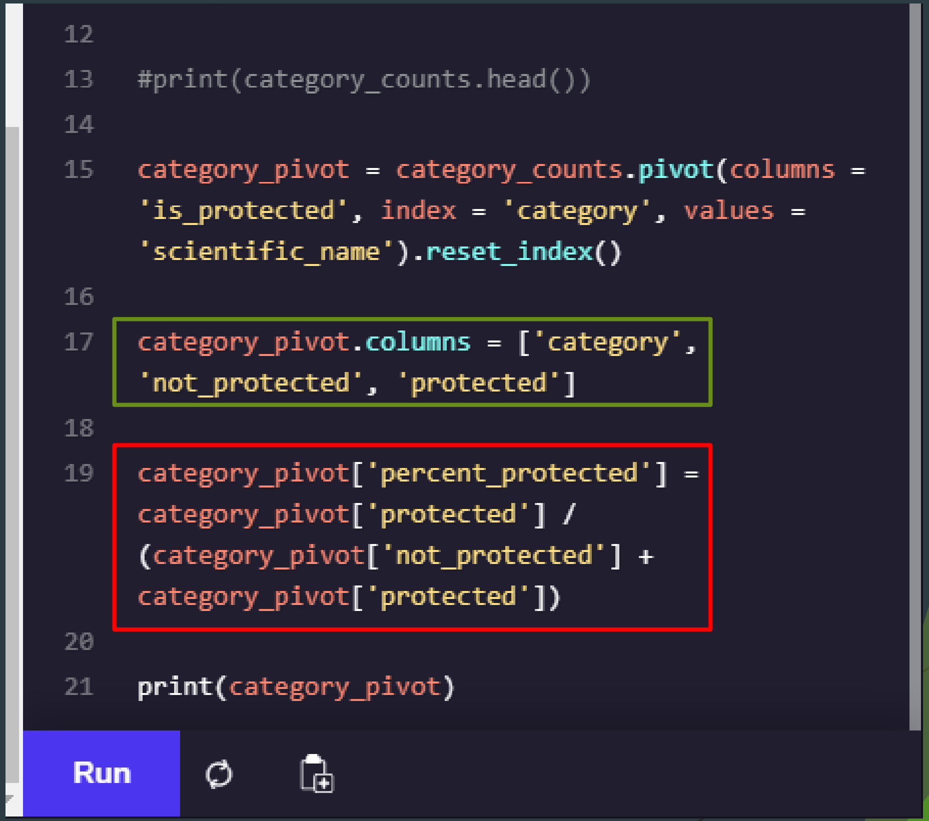
```
is_protected            category  False  True
0                      Amphibian     72     7
1                           Bird    413    75
2                           Fish    115    11
3                         Mammal    146    30
4              Nonvascular Plant    328     5
5                   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

Pivot Table makes data easier to read!

Birds and Vascular Plants: two most protected species
But what percentages are the highest?

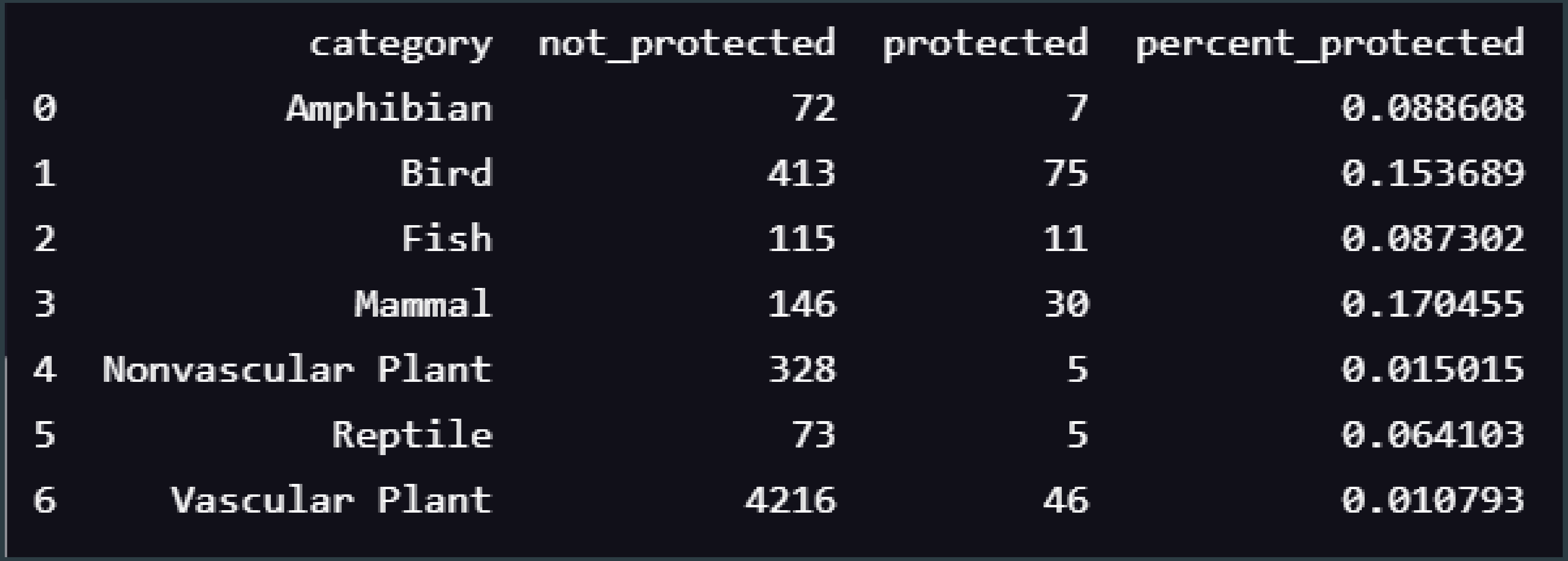
Step 3 (continued): Calculating Percentages of Protected Species

- Category_pivot.columns
  - Renames all columns with following inputs
- Category_pivot[‘percent_protected’]
  - Takes number of protected animals in each category
  - Divides it by total number of animals in each category



```
#print(category_counts.head())

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

category_pivot.columns = ['category',
'not_protected', 'protected']

category_pivot['percent_protected'] =
category_pivot['protected'] /
(category_pivot['not_protected'] +
category_pivot['protected'])

print(category_pivot)
```

Step 3 (continued): Percentages of Protected Species Results



```
            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
```

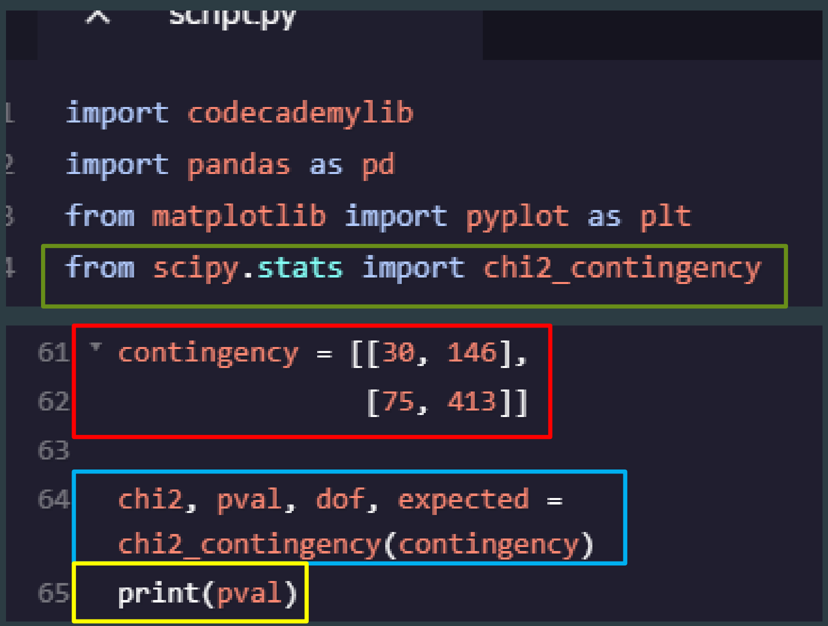
Mammals and Birds: two highest percentages of protected species

Step 4: Test Statistical Significance of Percentage Results

- Are mammals more likely to have species that need protection than birds or is the higher percentage due to chance?
- Need to perform statistical test
  - Data is **categorical** (although there are numbers associated, each animal is classified as either needing protection, or not needing protection)
  - Observing **two pieces of data** (comparing data collected from mammals to data collected from birds: two data sets)
- Perform Chi Square Test!
  - Statistical analysis when comparing two sets of categorical data

Step 4 (continued): Chi Square Test (Mammals vs Birds)

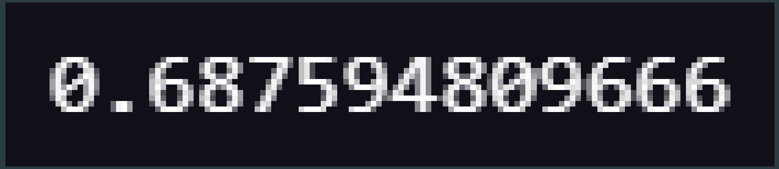
- Import chi square test from scipy into Python
- Create a table called contingency
  - Numbers in table are number of species either protected (column 1) or unprotected (column 2) for mammals (row 1) and birds (row 2)
- Perform contingency test
- Print p-value to evaluate significance
  - If p-value < 0.05, values have significance



```
import codecademylib
import pandas as pd
from matplotlib import pyplot as plt
from scipy.stats import chi2_contingency

contingency = [[30, 146],
               [75, 413]]

chi2, pval, dof, expected = chi2_contingency(contingency)
print(pval)
```

Step 4 (continued): Chi Square Test Results (Mammals vs Birds)

P-value =



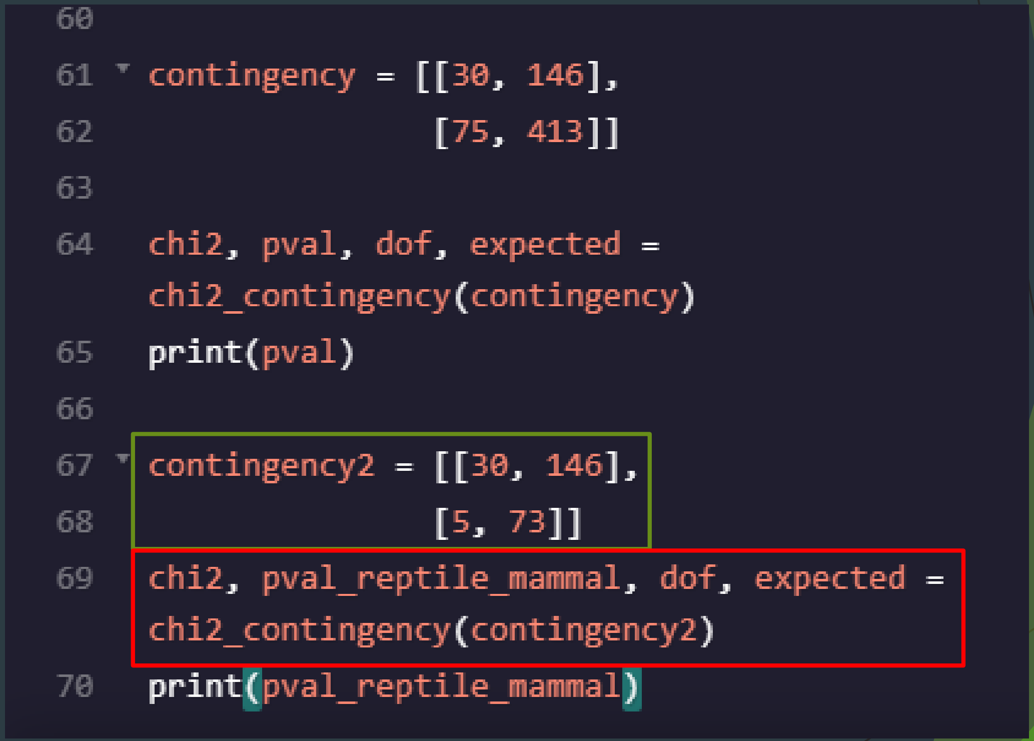
```
0.687594809666
```

Because the p-value is greater than 0.05, it is likely due to chance that mammals have a higher percentage of protected species than birds.

No significance!

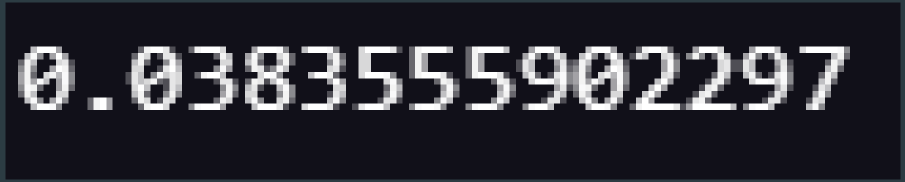
Step 4 (continued): Chi Square Test (Mammals vs Reptiles)

- What about if difference between mammals and reptile protection percentages are significant?
- Can run separate chi square test
  - Can only have two data sets per test
- Set up contingency table with new values
- Perform contingency test



```
contingency = [[30, 146],
               [75, 413]]

chi2, pval, dof, expected = chi2_contingency(contingency)
print(pval)

contingency2 = [[30, 146],
                [5, 73]]
chi2, pval_reptile_mammal, dof, expected = chi2_contingency(contingency2)
print(pval_reptile_mammal)
```

Step 4 (continued): Chi Square Test Results (Mammals vs Reptiles)

P-value = 

```
0.0383555902297
```

The p-value is less than 0.05.

There is significance between differences of protection percentages for mammals and reptiles!

What was learned in Task 1?

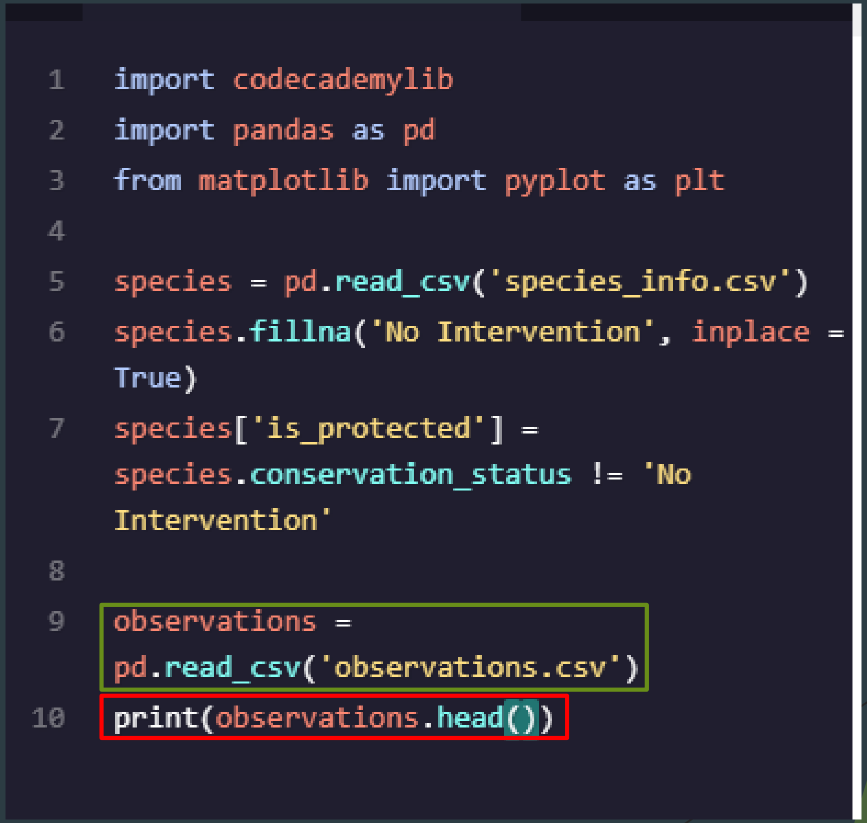
- How to open and read a DataFrame
- How to manipulate a DataFrame
  - Create new columns
  - Perform functions and counts on values in columns or rows
- Plot data into a bar graph
- Determine which statistical analysis to perform and determine if values have significance

Task 2

Analyze observations of species in National Parks and determine sample size needed for observable difference in disease in species

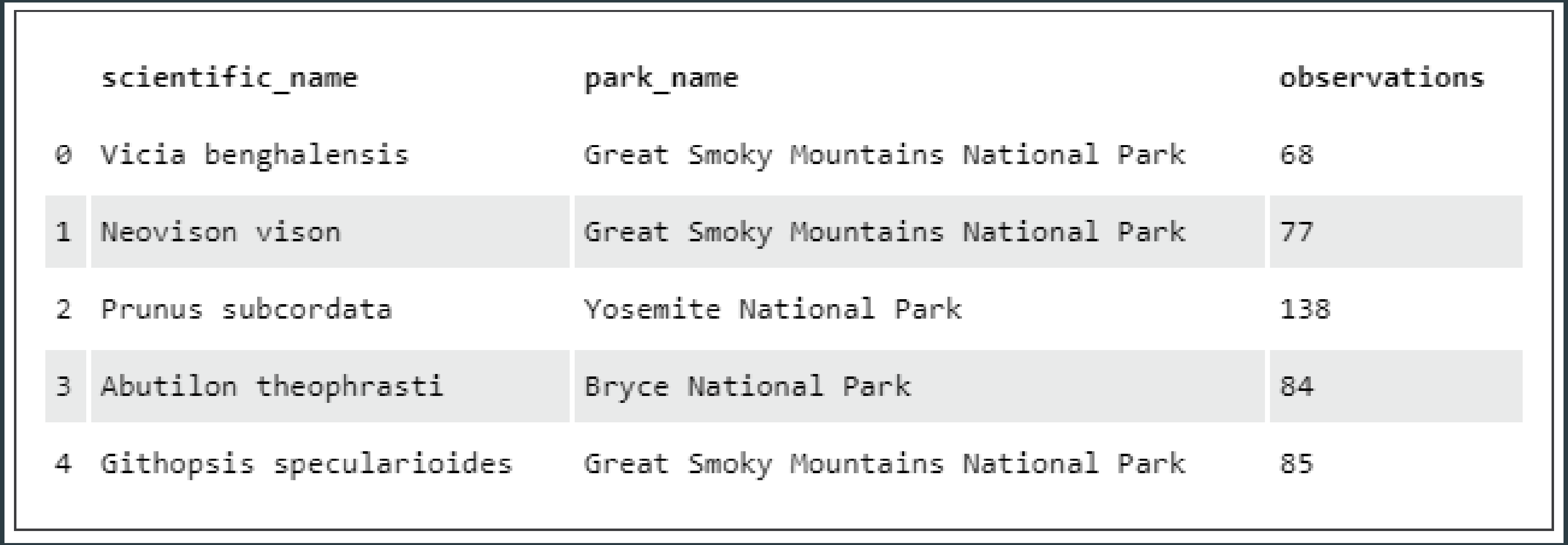
Step 1: Open and understand Observations CSV file

- Opens observations.csv file
- Prints only first 5 lines of .csv file



```python
import codecademylib
import pandas as pd
from matplotlib import pyplot as plt

species = pd.read_csv('species_info.csv')
species.fillna('No Intervention', inplace =
True)
species['is_protected'] =
species.conservation_status != 'No
Intervention'

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

Step 1 (continued): First five lines of observations.csv

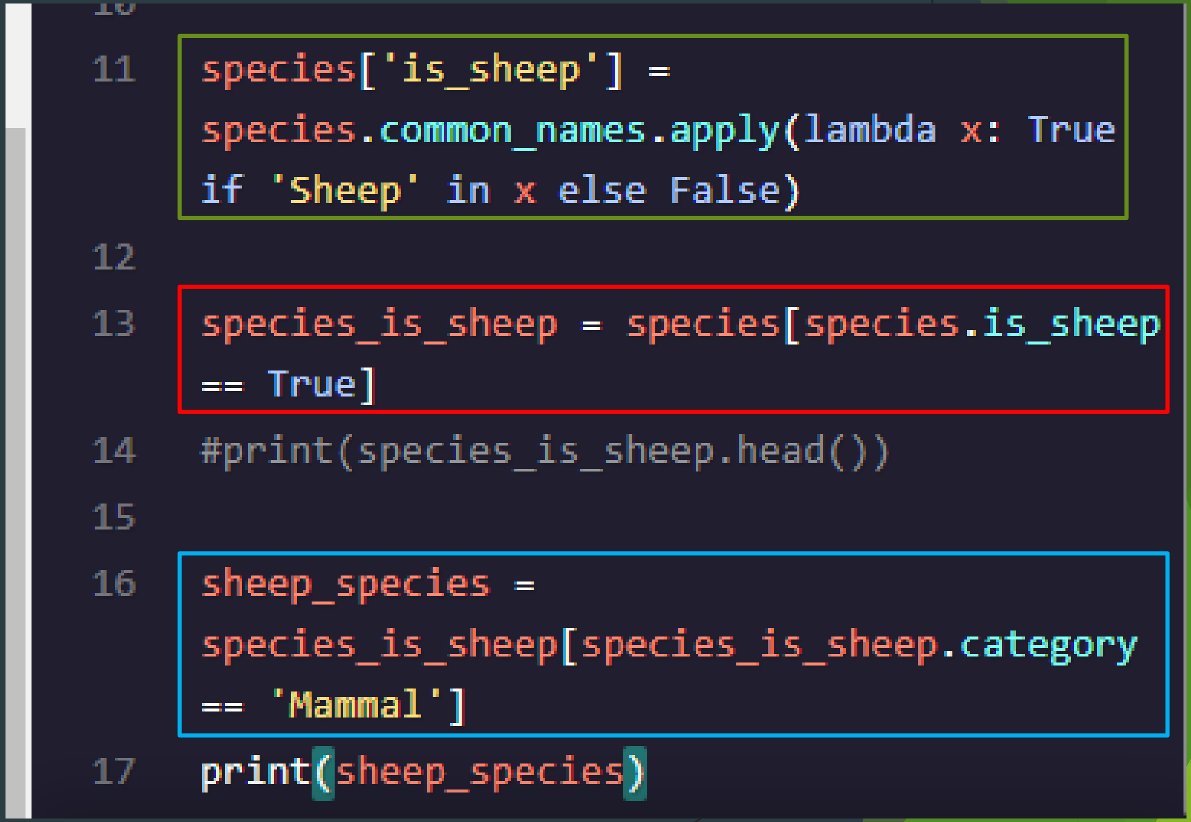


	scientific_name	park_name	observations
0	Vicia benghalensis	Great Smoky Mountains National Park	68
1	Neovison vison	Great Smoky Mountains National Park	77
2	Prunus subcordata	Yosemite National Park	138
3	Abutilon theophrasti	Bryce National Park	84
4	Githopsis specularioides	Great Smoky Mountains National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

3 columns: scientific name of animal, park animal was observed in, number of observations

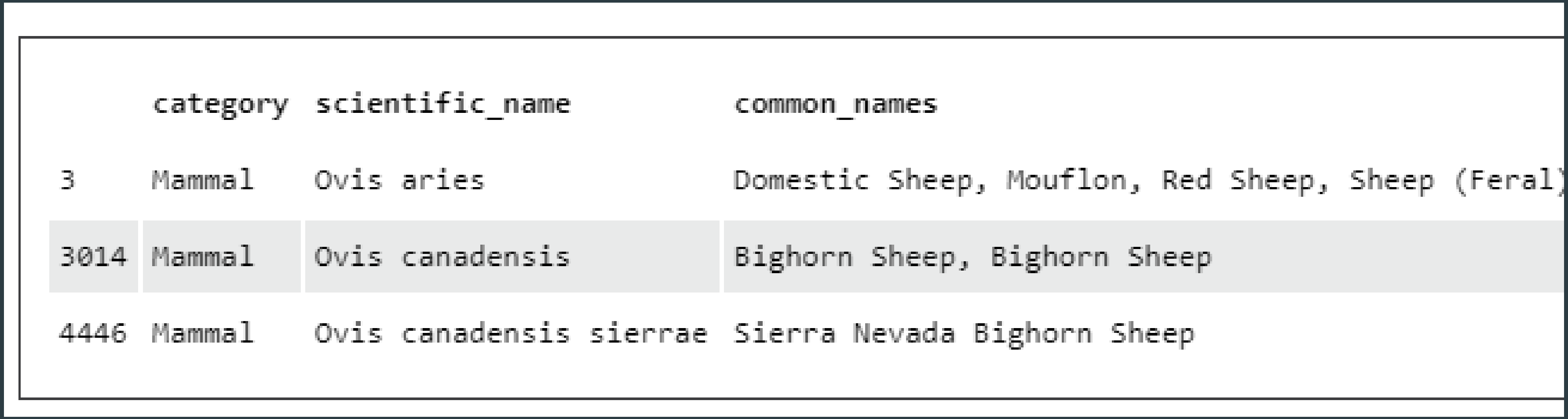
Step 2: Create New DataFrame that Counts Number of Sheep in Parks

- Creates new column to observations DataFrame titled ‘is_sheep’
  - Finds whether common name column has word ‘sheep’ in it
- Creates new DataFrame including only species that are sheep
- Creates new DataFrame including only species that are sheep AND are mammals



```
species['is_sheep'] =
species.common_names.apply(lambda x: True
if 'Sheep' in x else False)

species_is_sheep = species[species.is_sheep
== True]
#print(species_is_sheep.head())

sheep_species =
species_is_sheep[species_is_sheep.category
== 'Mammal']
print(sheep_species)
```

Step 2 (continued): Sheep DataFrame Results

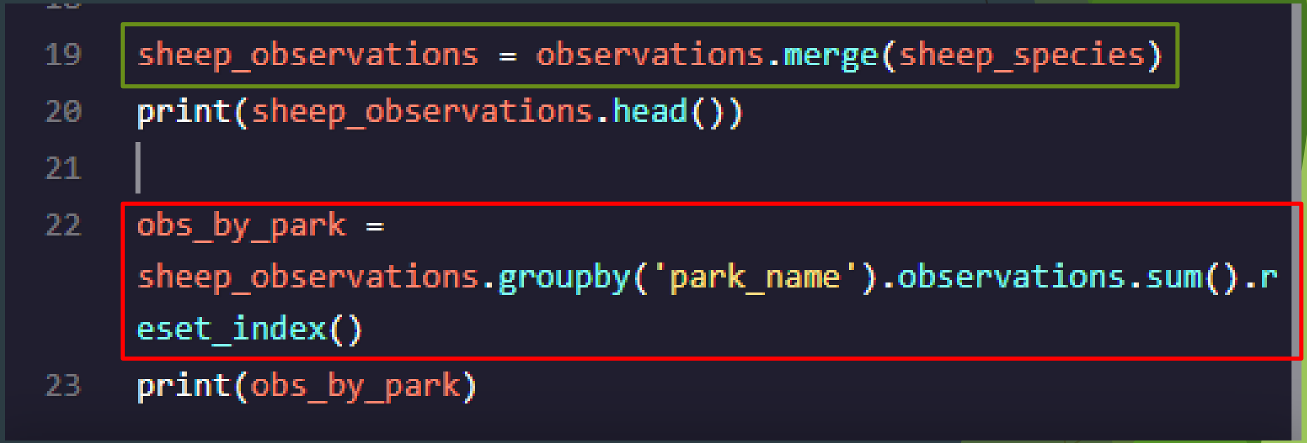


	category	scientific_name	common_names
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral
3014	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep
4446	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep

DataFrame contains three species of sheep

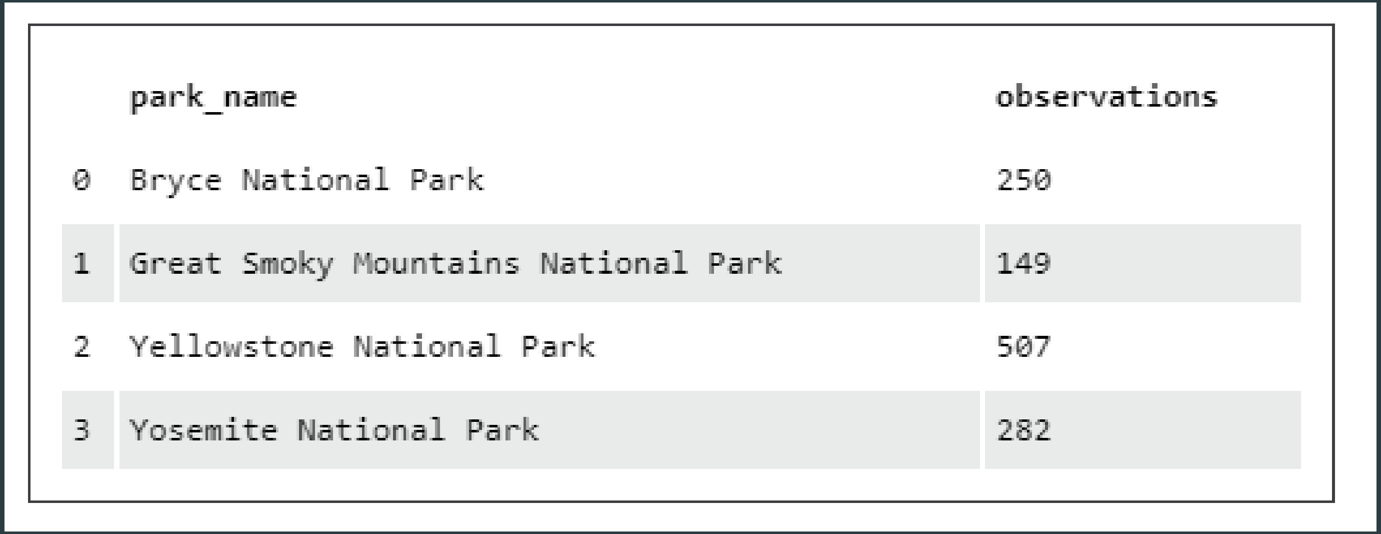
Step 3: Determine Where Sheep are Observed

- Need to determine which National Parks sheep are being observed from
- Merges observations DataFrame and sheep_species DataFrame
  - Done to perform eventual grouping
- Counts number of sheep observed in each National Park
  - 'park_name' = column want to group by
  - Observations = column want to do measurement on
  - Sum = measurement performed



```
sheep_observations = observations.merge(sheep_species)
print(sheep_observations.head())

obs_by_park =
sheep_observations.groupby('park_name').observations.sum().r
eset_index()
print(obs_by_park)
```

Step 3 (continued): Observations of Sheep by Park

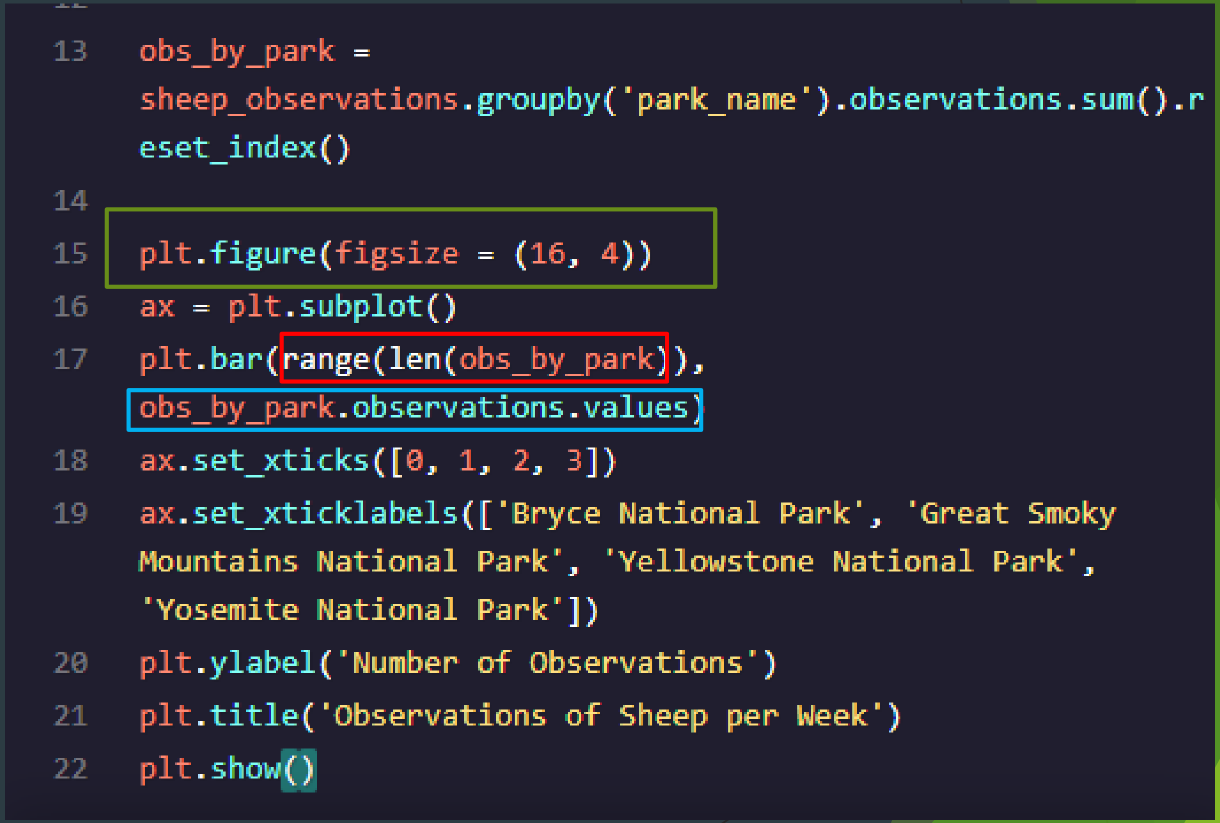


	park_name	observations
0	Bryce National Park	250
1	Great Smoky Mountains National Park	149
2	Yellowstone National Park	507
3	Yosemite National Park	282

All sheep observed in four National Parks

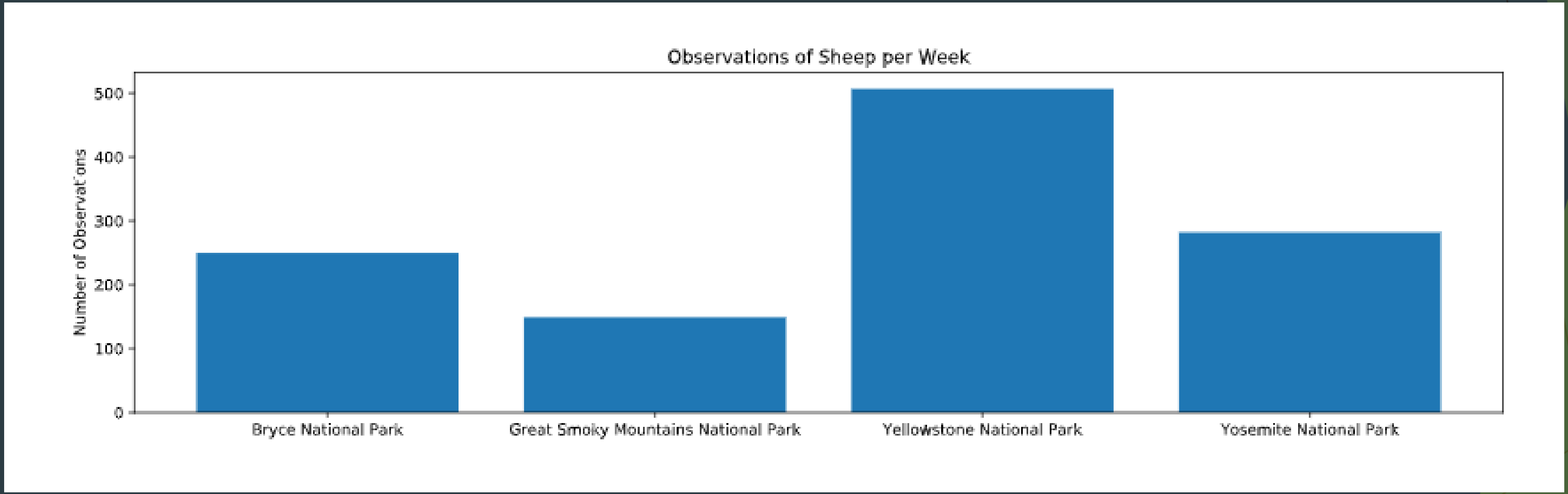
Step 4: Plot Observations of Sheep by Park into Bar Graph

- Figure size = 16 in. by 4 in.
- X-axis values = number of rows in obs_by_park DataFrame
- Y-axis values = values of the observations per park



```
obs_by_park = sheep_observations.groupby('park_name').observations.sum().reset_index()

plt.figure(figsize = (16, 4))
ax = plt.subplot()
plt.bar(range(len(obs_by_park)), obs_by_park.observations.values)
ax.set_xticks([0, 1, 2, 3])
ax.set_xticklabels(['Bryce National Park', 'Great Smoky Mountains National Park', 'Yellowstone National Park', 'Yosemite National Park'])
plt.ylabel('Number of Observations')
plt.title('Observations of Sheep per Week')
plt.show()
```

Step 4 (continued): Bar Graph of Sheep Observations by Park



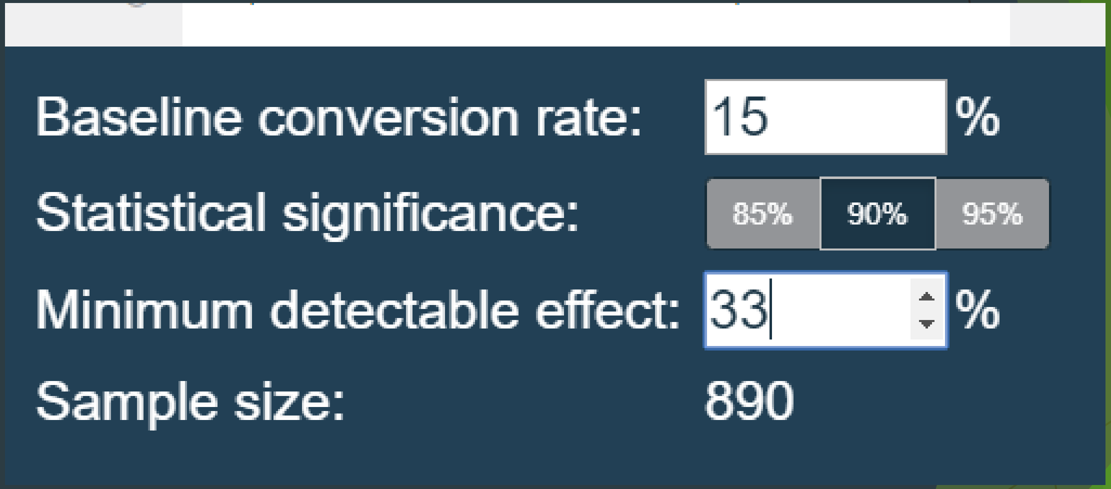
Step 5: Sample Size Determination for Foot and Mouth Disease Reduction

- Park Rangers want to help reduce amount of foot and mouth disease in sheep
  - Want to detect ±5% change to percentage of sheep that have disease
  - 15% of sheep in Bryce National Park last year had disease
  - Confidence level = 90%

- What is the sample size needed to get observable detection of ±5% and how long will it take to observe that many sheep?

Step 5 (continued): Determination of Sample Size

- Baseline = 15%
  - Because 15% of sheep observed at Bryce National Park last year had foot and mouth disease
- Statistical significance = 90%
  - Given
- Minimum detectable effect = 33%
  - Minimum detectable effect = 100 (.05 / 15)
  - 0.05 is 5% difference Park Rangers wish to detect and 15 is baseline
- Sample size needed = 890
  - From calculator


Baseline conversion rate: 15 %

Statistical significance: 85% [✓ 90%] 95%

Minimum detectable effect: 33 %

Sample size: 890

Step 5 (continued): Number of Weeks Needed

- If Yellowstone National Park has 507 sheep observations a week
  - Need to observe for about 2 weeks to get full sample size
  - 890 sample size / 507 observations = 1.755

- If Bryce National Park has 250 sheep observations a week
  - Need to observe for about 4 weeks to get full sample size
  - 890 sample size / 250 observations = 3.56

What was learned in Task 2?

- Merging and manipulating DataFrames based on information in other DataFrames
- Plotting bar graph with information from manipulated DataFrame
- Sample size determination

The End

Thank you for reading!