Capstone 2: Biodiversity Project

Introduction

You are a biodiversity analyst working for the National Parks Service. You're going to help them analyze some data about species at various national parks.

Note: The data that you'll be working with for this project is *inspired* by real data, but is mostly fictional.

Step 1

Import the modules that you'll be using in this assignment:

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

```
from matplotlib import pyplot as plt
import matplotlib
import numpy as np
import pandas as pd
```

Step 2

You have been given two CSV files. `species_info.csv` with data about different species in our National Parks, including:

- The scientific name of each species
- The common names of each species
- The species conservation status

Load the dataset and inspect it:

- Load `species_info.csv` into a DataFrame called `species`

```
species = pd.read_csv("species_info.csv")
```

Inspect each DataFrame using `.head()`.

```
display(species.head())
```

	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), Dom...	NaN
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	NaN
4	Mammal	Cervus elaphus	Wapiti Or Elk	NaN

Step 3

Let's start by learning a bit more about our data. Answer each of the following questions.

How many different species are in the `species` DataFrame?

```
print(species.nunique())
```

```
category                 7
scientific_name       5541
common_names          5504
conservation_status       4
dtype: int64
```

What are the different values of `category` in `species` ?

```
print(species.category.unique())
```

```
['Mammal' 'Bird' 'Reptile' 'Amphibian' 'Fish' 'Vascular Plant'
 'Nonvascular Plant']
```

What are the different values of `conservation_status` ?

```
print(species.conservation_status.unique())
```

```
[nan 'Species of Concern' 'Endangered' 'Threatened' 'In Recovery']
```

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.

```
print(species.groupby("conservation_status").scientific_name.count())
```

```
conservation_status
Endangered             16
In Recovery             4
Species of Concern    161
Threatened             10
Name: scientific_name, dtype: int64
```

As we saw before, there are far more than 200 species in the `species` table. Clearly, only a small number of them are categorized as needing some sort of protection. The rest have `conservation_status` equal to `None` . Because `groupby` does not include `None` , we will need to fill in the null values. We can do this using `.fillna` . We pass in however we want to fill in our `None` values as an argument.

Paste the following code and run it to see replace `None` with `No Intervention` :

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

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

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

```
print(species.groupby("conservation_status").scientific_name.count())
```

```
conservation_status
Endangered              16
In Recovery              4
No Intervention       5633
Species of Concern     161
Threatened              10
Name: scientific_name, dtype: int64
```

Let's use `plt.bar` to create a bar chart. First, let's sort the columns by how many species are in each categories. We can do this using `.sort_values`. We use the the keyword `by` to indicate which column we want to sort by.

Paste the following code and run it to create a new DataFrame called `protection_counts`, which is sorted by `scientific_name`:

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

```
protection_counts = species.groupby(['category', 'conservation_status
'])\
    .scientific_name.nunique().reset_index()\
    .sort_values(['category','conservation_status']).reset_index(drop=Tr
ue)

protection_counts.rename(columns ={'scientific_name': 'count'}, inplace=
True)
to_add = {}

index = 1
for cat in protection_counts.category.unique():
    for status in protection_counts.conservation_status.unique():
        if protection_counts.loc[(protection_counts.category == cat)
                                 & (protection_counts.conservation_status ==
status),
                                 "count"].any() == False:
            to_add[index] = [cat, status,0]
            index += 1

df = pd.DataFrame(to_add.values(), columns=['category','conservation_sta
tus', 'count'])

protection_counts_new = pd.concat([protection_counts, df], axis= 0).sort
_values(['category','conservation_status']).reset_index(drop=True)
display(protection_counts_new)
```

	category	conservation_status	count
0	Amphibian	Endangered	1
1	Amphibian	In Recovery	0
2	Amphibian	No Intervention	72
3	Amphibian	Species of Concern	4
4	Amphibian	Threatened	2
5	Bird	Endangered	4
6	Bird	In Recovery	3
7	Bird	No Intervention	413
8	Bird	Species of Concern	68
9	Bird	Threatened	0
10	Fish	Endangered	3
11	Fish	In Recovery	0
12	Fish	No Intervention	115
13	Fish	Species of Concern	4
14	Fish	Threatened	4
15	Mammal	Endangered	6
16	Mammal	In Recovery	1

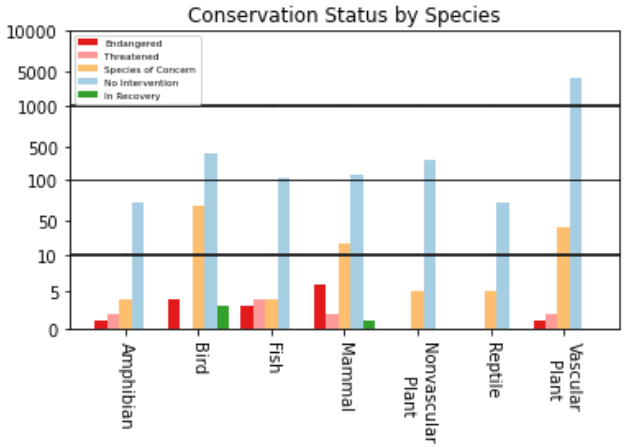
Now let's create a bar chart!

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 graph using `plt.show()`

	category	conservation_status	count
17	Mammal	No Intervention	146
18	Mammal	Species of Concern	22
19	Mammal	Threatened	2
20	Nonvascular Plant	Endangered	0
21	Nonvascular Plant	In Recovery	0
22	Nonvascular Plant	No Intervention	328
23	Nonvascular Plant	Species of Concern	5
24	Nonvascular Plant	Threatened	0
25	Reptile	Endangered	0
26	Reptile	In Recovery	0
27	Reptile	No Intervention	73
28	Reptile	Species of Concern	5
29	Reptile	Threatened	0
30	Vascular Plant	Endangered	1
31	Vascular Plant	In Recovery	0
32	Vascular Plant	No Intervention	4216
33	Vascular Plant	Species of Concern	43
34	Vascular Plant	Threatened	2

```
labels = protection_counts_new.category.unique()
endangered_means = protection_counts_new.loc[protection_counts_new.conse
rvation_status == 'Endangered', "count"].to_numpy()
recovery_means = protection_counts_new.loc[protection_counts_new.conserv
ation_status == 'In Recovery', "count"].to_numpy()
no_intervention_means = protection_counts_new.loc[protection_counts_new.
conservation_status == 'No Intervention', "count"].to_numpy()
concern_means = protection_counts_new.loc[protection_counts_new.conserva
tion_status == 'Species of Concern', "count"].to_numpy()
threatened_means = protection_counts_new.loc[protection_counts_new.conse
rvation_status == 'Threatened', "count"].to_numpy()


labels[4] = "Nonvascular\n Plant"
labels[6] = "Vascular\n Plant"

x = np.arange(len(labels)) * 1.5
width = 0.25

y_range = [ [0,10], [10,100], [100, 1000], [1000, 10000]]


fig, (ax1, ax2, ax3, ax4) = plt.subplots(nrows=4, ncols=1, sharex=True)


ax1.set_title('Conservation Status by Species')
ax1.set_ylim(y_range[3])
ax1.bar(x - width*2, endangered_means, width, label='Endangered', align=
'center', color="#e31a1c")
ax1.bar(x - width, threatened_means, width, label='Threatened', align='c
enter', color="#fb9a99")
ax1.bar(x, concern_means, width, label='Species of Concern', align='cent
er', color="#fdbf6f")
ax1.bar(x + width, no_intervention_means, width, label='No Intervention
', align='center', color="#a6cee3")
ax1.bar(x + width*2, recovery_means, width, label='In Recovery', align='
center', color="#33a02c")
ax1.legend(loc='upper left', ncol=1, fontsize='xx-small')

ax2.set_ylim(y_range[2])
ax2.bar(x - width*2, endangered_means, width, label='Endangered', align=
'center', color="#e31a1c")
ax2.bar(x - width, threatened_means, width, label='Threatened', align='c
enter', color="#fb9a99")
ax2.bar(x, concern_means, width, label='Species of Concern', align='cent
er', color="#fdbf6f")
ax2.bar(x + width, no_intervention_means, width, label='No Intervention
', align='center', color="#a6cee3")
ax2.bar(x + width*2, recovery_means, width, label='In Recovery', align='
center', color="#33a02c")


ax3.set_ylim(y_range[1])
```

```
ax3.bar(x - width*2, endangered_means, width, label='Endangered', align=
'center', color="#e31a1c")
ax3.bar(x - width, threatened_means, width, label='Threatened', align='c
enter', color="#fb9a99")
ax3.bar(x, concern_means, width, label='Species of Concern', align='cent
er', color="#fdbf6f")
ax3.bar(x + width, no_intervention_means, width, label='No Intervention
', align='center', color="#a6cee3")
ax3.bar(x + width*2, recovery_means, width, label='In Recovery', align='
center', color="#33a02c")


ax4.set_xticks(x)
ax4.set_xticklabels(labels, rotation=-90)
ax4.set_ylim(y_range[0])
ax4.bar(x - width*2, endangered_means, width, label='Endangered', align=
'center', color="#e31a1c")
ax4.bar(x - width, threatened_means, width, label='Threatened', align='c
enter', color="#fb9a99")
ax4.bar(x, concern_means, width, label='Species of Concern', align='cent
er', color="#fdbf6f")
ax4.bar(x + width, no_intervention_means, width, label='No Intervention
', align='center', color="#a6cee3")
ax4.bar(x + width*2, recovery_means, width, label='In Recovery', align='
center', color="#33a02c")

fig.subplots_adjust(hspace=0.02, bottom=0.25)

plt.savefig('Conservation_Status_vs_Species.jpeg', format='jpeg')
plt.show()
```



Step 4

Are certain types of species more likely to be endangered?

Let's create a new column in `species` called `is_protected`, which is `True` if `conservation_status` is not equal to `No Intervention`, and `False` otherwise.

```
species["is_protected"] = species.conservation_status.apply(lambda x: Tr
ue if x != 'No Intervention' else False)
display(species.head())
```

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

Let's group the `species` data frame by the `category` and `is_protected` columns and count the unique `scientific_name` s in each grouping.

Save your results to `category_counts` .

```
category_counts = pd.DataFrame(species.groupby(["category", 'is_protecte
d']).scientific_name.nunique()).reset_index()
display(category_counts)
```

	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
5	Fish	True	11
6	Mammal	False	146
7	Mammal	True	30
8	Nonvascular Plant	False	328
9	Nonvascular Plant	True	5
10	Reptile	False	73
11	Reptile	True	5
12	Vascular Plant	False	4216
13	Vascular Plant	True	46

Examine `category_counts` using `head()` .

```
display(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

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(index='category', columns='is_pro
tected', values='scientific_name')
```

Examine `category_pivot`.

```
display(category_pivot)
```

is_protected	False	True
category		
Amphibian	72	7
Bird	413	75
Fish	115	11
Mammal	146	30
Nonvascular Plant	328	5
Reptile	73	5
Vascular Plant	4216	46

Use the `.columns` property to rename the categories `True` and `False` to something more description:

- Leave `category` as `category`
- Rename `False` to `not_protected`
- Rename `True` to `protected`

```
category_pivot.rename(columns={False: 'not_protected', True: 'protected
'}, inplace=True)
```

Let's create a new column of `category_pivot` called `percent_protected`, which is equal to `protected` (the number of species that are protected) divided by `protected` plus `not_protected` (the total number of species).

```
category_pivot['precent_protected'] = category_pivot['protected'] /(cate
gory_pivot['not_protected'] + category_pivot['protected'])
```

Examine `category_pivot`.

```
display(category_pivot.sort_values(by='precent_protected', axis=0,ascend
ing=False))
```

is_protected category	not_protected	protected	precent_protected
Mammal	146	30	0.170455
Bird	413	75	0.153689
Amphibian	72	7	0.088608
Fish	115	11	0.087302
Reptile	73	5	0.064103
Nonvascular Plant	328	5	0.015015
Vascular Plant	4216	46	0.010793

It looks like species in category `Mammal` are more likely to be endangered than species in `Bird`. We're going to do a significance test to see if this statement is true. Before you do the significance test, consider the following questions:

- Is the data numerical or categorical?
- Categorical
- How many pieces of data are you comparing?
- 4 pieces of data

Based on those answers, you should choose to do a *chi squared test*. In order to run a chi squared test, we'll need to create a contingency table. Our contingency table should look like this:

	protected	not protected
Mammal	?	?
Bird	?	?

Create a table called `contingency` and fill it in with the correct numbers

```
contingency = [[category_pivot.loc["Mammal", 'protected'], category_pivo
t.loc["Mammal", 'not_protected']],
               [category_pivot.loc["Bird", 'protected'], category_piv
ot.loc["Bird", 'not_protected']]]
```

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:

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

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

Now run `chi2_contingency` with `contingency`.

```
Stat, PValue, DOF, ExpFreq = chi2_contingency(contingency)
print(PValue)
```

```
0.6875948096661336
```

It looks like this difference isn't significant!

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

```
contingency2 = [[category_pivot.loc["Mammal", 'protected'], category_piv
ot.loc["Mammal", 'not_protected']],
                [category_pivot.loc["Reptile", 'protected'], category_
pivot.loc["Reptile", 'not_protected']]]
Stat2, PValue2, DOF2, ExpFreq2 = chi2_contingency(contingency2)
print(PValue2)
```

```
0.03835559022969898
```

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

Step 5

Conservationists have been recording sightings of different species at several national parks for the past 7 days. They've saved sent you their observations in a file called `observations.csv`. Load `observations.csv` into a variable called `observations`, then use `head` to view the data.

```
observations = pd.read_csv("observations.csv")
display(observations.head(10))
```

	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
5	Elymus virginicus var. virginicus	Yosemite National Park	112
6	Spizella pusilla	Yellowstone National Park	228
7	Elymus multisetus	Great Smoky Mountains National Park	39
8	Lysimachia quadrifolia	Yosemite National Park	168
9	Diphyscium cumberlandianum	Yellowstone National Park	250

Some scientists are studying the number of sheep sightings at different national parks. There are several different scientific names for different types of sheep. We'd like to know which rows of `species` are referring to sheep. Notice that the following code will tell us whether or not a word occurs in a string:

```
# Does "Sheep" occur in this string?
str1 = 'This string contains Sheep'
'Sheep' in str1
```

```
True
```

```
# Does "Sheep" occur in this string?
str2 = 'This string contains Cows'
'Sheep' in str2
print(observations.columns)
display(species)
```

```
Index(['scientific_name', 'park_name', 'observations'], dtype='object')
```

	category	scientific_name	common_names	conservation_status	is_p
0	Mammal	Clethrionomys gapperi gapperi	Gapper's Red-Backed Vole	No Intervention	False
1	Mammal	Bos bison	American Bison, Bison	No Intervention	False
2	Mammal	Bos taurus	Aurochs, Aurochs, Domestic Cattle (Feral), Dom...	No Intervention	False
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False
4	Mammal	Cervus elaphus	Wapiti Or Elk	No Intervention	False
...	...	...	...	...	...
5819	Vascular Plant	Solanum parishii	Parish's Nightshade	No Intervention	False
5820	Vascular Plant	Solanum xanti	Chaparral Nightshade, Purple Nightshade	No Intervention	False
5821	Vascular Plant	Parthenocissus vitacea	Thicket Creeper, Virginia Creeper, Woodbine	No Intervention	False
5822	Vascular Plant	Vitis californica	California Grape, California Wild Grape	No Intervention	False
5823	Vascular Plant	Tribulus terrestris	Bullhead, Caltrop, Goathead, Mexican Sandbur, ...	No Intervention	False

5824 rows × 5 columns

Use `apply` and a `lambda` function to create a new column in `species` called `is_sheep` which is `True` if the `common_names` contains `'Sheep'`, and `False` otherwise.

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

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

```
display(species.loc[species.is_sheep == True,:])
```

	category	scientific_name	common_names	conservation_status	is_p
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False
1139	Vascular Plant	Rumex acetosella	Sheep Sorrel, Sheep Sorrell	No Intervention	False
2233	Vascular Plant	Festuca filiformis	Fineleaf Sheep Fescue	No Intervention	False
3014	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True
3758	Vascular Plant	Rumex acetosella	Common Sheep Sorrel, Field Sorrel, Red Sorrel,...	No Intervention	False
3761	Vascular Plant	Rumex paucifolius	Alpine Sheep Sorrel, Fewleaved Dock, Meadow Dock	No Intervention	False
4091	Vascular Plant	Carex illota	Sheep Sedge, Smallhead Sedge	No Intervention	False
4383	Vascular Plant	Potentilla ovina var. ovina	Sheep Cinquefoil	No Intervention	False
4446	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True

Many of the results are actually plants. Select the rows of `species` where `is_sheep` is `True` and `category` is `Mammal`. Save the results to the variable `sheep_species`.

```
sheep_species = species.loc[(species.category == 'Mammal') & (species.is
_sheep == True), :]
display(sheep_species)
```

	category	scientific_name	common_names	conservation_status	is_p
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False
3014	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True
4446	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True

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

```
sheep_observations = sheep_species.merge(observations, how='inner', on="
scientific_name")
display(sheep_observations.head(20))
```

	category	scientific_name	common_names	conservation_status	is_pro
0	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False
1	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False
2	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False
3	Mammal	Ovis aries	Domestic Sheep, Mouflon, Red Sheep, Sheep (Feral)	No Intervention	False
4	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True
5	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True
6	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True
7	Mammal	Ovis canadensis	Bighorn Sheep, Bighorn Sheep	Species of Concern	True
8	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True
9	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True
10	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True
11	Mammal	Ovis canadensis sierrae	Sierra Nevada Bighorn Sheep	Endangered	True

How many total sheep observations (across all three species) were made at each national park? Use `groupby` to get the `sum` of `observations` for each `park_name`. Save your answer to `obs_by_park`.

This is the total number of sheep observed in each park over the past 7 days.

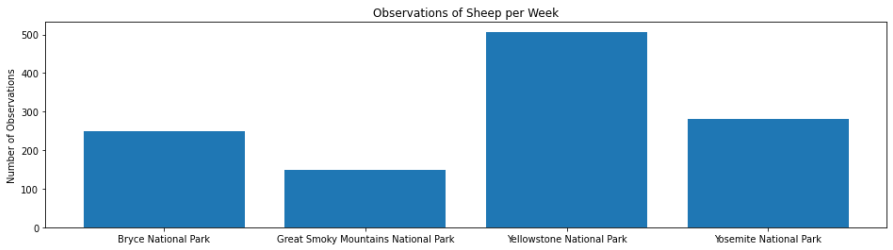
```
obs_by_park = sheep_observations.groupby("park_name").observations.sum
().reset_index()
display(obs_by_park)
```

	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

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()`

```
fig_1 = plt.figure(figsize=(16,4))
ax_1 = plt.subplot()
ax_1.bar(obs_by_park.index, obs_by_park.observations, tick_label=obs_by_
park.park_name)
ax_1.set_ylabel("Number of Observations")
ax_1.set_title("Observations of Sheep per Week")
plt.show()
```



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 this, with confidence.

Use Codecademy's sample size calculator (https://s3.amazonaws.com/codecademy-content/courses/learn-hypothesis-testing/a_b_sample_size/index.html) 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.

```
n_observations_req = 890
```

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?

```
n_of_weeks_4_sigf =  n_observations_req / obs_by_park.loc[obs_by_park.pa
rk_name == "Yellowstone National Park", "observations"]
print(n_of_weeks_4_sigf )
```

```
2    1.755424
Name: observations, dtype: float64
```