In [1]:

```
import numpy as np
from datascience import *

# Configure notebook (happens automatically on data8.berkeley.edu)
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')

# Configure for presentation
np.set_printoptions(threshold=50, linewidth=50)
import matplotlib as mpl
mpl.rc('font', size=16)
```

Section 1 - Working with tables

In [2]:

```
## Run the following code to import a data set on home sales in 2010.

house_data = Table().read_table('/srv/data/DS_113_S23/Tests/house.csv')

house_data
```

Out[2]:

Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Utilit
1	526301100	20	RL	141	31770	Pave	nan	IR1	Lvl	All
2	526350040	20	RH	80	11622	Pave	nan	Reg	Lvl	All
3	526351010	20	RL	81	14267	Pave	nan	IR1	Lvl	All
4	526353030	20	RL	93	11160	Pave	nan	Reg	Lvl	All
5	527105010	60	RL	74	13830	Pave	nan	IR1	Lvl	All
6	527105030	60	RL	78	9978	Pave	nan	IR1	Lvl	All
7	527127150	120	RL	41	4920	Pave	nan	Reg	Lvl	All
8	527145080	120	RL	43	5005	Pave	nan	IR1	HLS	All
9	527146030	120	RL	39	5389	Pave	nan	IR1	Lvl	All
10	527162130	60	RL	60	7500	Pave	nan	Reg	Lvl	All

... (2920 rows omitted)

Question 1.1

In [12]:

```
## Please produce a new table that only has the columns "Neighborhood" and '
## Call it hood_values.
hood_values = Table().with_columns("Neighborhood", house_data["Neighborhood'
                                  "SalePrice", house_data["SalePrice"])
hood_values
```

Out[12]:

Neighborhood	SalePrice
NAmes	215000
NAmes	105000
NAmes	172000
NAmes	244000
Gilbert	189900
Gilbert	195500
StoneBr	213500
StoneBr	191500
StoneBr	236500
Gilbert	189000

... (2920 rows omitted)

Question 1.2

In [15]:

```
## Please make two tables, each with one row per neighborhood and two columr
## The first table should have one column showing the name of each neighborh
## and a second column reporting the average sale price for that neighborhoc
## The second table should report the number of houses sold in each neighbor
avg_salePrice = hood_values.group("Neighborhood", np.average)
avg_salePrice.show(5)

sold_count = hood_values.group("Neighborhood")
sold_count.show(5)
```

Neighborhood	SalePrice average
Blmngtn	196662
Blueste	143590
BrDale	105608
BrkSide	124756
ClearCr	208662

... (23 rows omitted)

Neighborhood	count
Blmngtn	28
Blueste	10
BrDale	30
BrkSide	108
ClearCr	44

... (23 rows omitted)

Question 1.3

In [37]:

```
## For the five most expensive neighborhoods, please make a table with three
## showing the name of the neighborhood, the average home value, and the num
## Please give the columns nice names.
## Full credit requires getting the code to do all the steps (i.e. no lookir

## join the two table and find the top 5 expensive neighborhoods
top5_salePrice = avg_salePrice.join("Neighborhood", sold_count).sort(
    "SalePrice average", descending=True).take(np.arange(5)).relabel(1, "Avg
top5_salePrice
```

Out[37]:

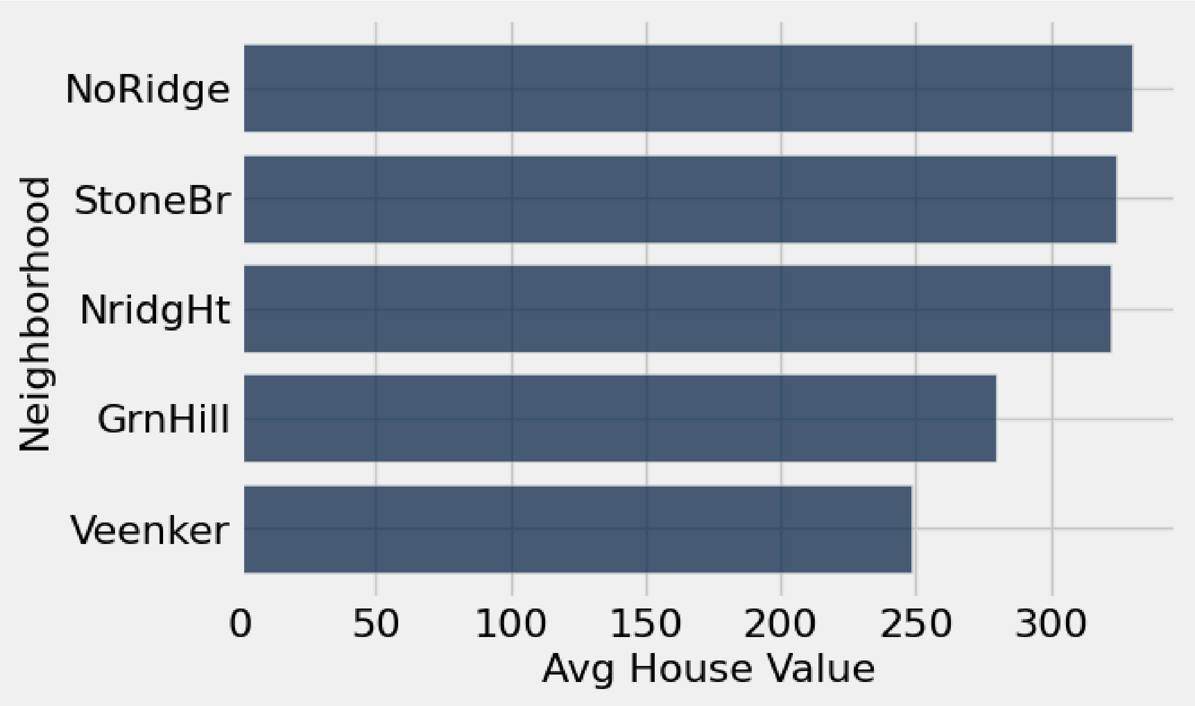
Neighborhood	Avg Home Value	# House sold
NoRidge	330319	71
StoneBr	324229	51
NridgHt	322018	166
GrnHill	280000	2
Veenker	248315	24

Question 1.4

In [38]:

```
## Change the units of Average Value from dollars to 1000's of dollars
## i.e. 353.890 instead of 352890.
## Then create a bar plot showing the five most expensive neighborhoods and
top5_salePrice = top5_salePrice.with_column("Avg House Value", top5_salePric
top5_salePrice.show()
top5_salePrice.barh("Neighborhood", "Avg House Value")
```

Neighborhood	# House sold	Avg House Value
NoRidge	71	330.319
StoneBr	51	324.229
NridgHt	166	322.018
GrnHill	2	280
Veenker	24	248.315



Question 1.5

In [45]:

```
## Are newer homes worth more?
## Go back to the original data table (house data) and calculate the average
## and after 1970.  Round each to the nearest dollar.

## before 1970
avg_before_1970 = round(np.average(house_data.where("Year Built", are.below(
print(avg_before_1970)
## after 1970
avg_after_1970 = round(np.average(house_data.where("Year Built", are.above_c
print(avg_after_1970)
```

```
137567
216681
```

Bonus Question (only if you have time)

In [ ]:

```
## Think of one potential confounding factor in your analysis above.
## Explore this confounding factor and see if it played a role in the differ
## (A confounding factor is a third variable that might be related to both c
```

Write your answer here.

Section 2 - Defining and using functions

Question 2.1

In [46]:

```
## Define a function that takes a number, squares it, and adds 7.
def square_and_add7 (number):
    '''take a number, square it and add 7'''
    return (number**2)+7
```

Out[46]:

```
11
```

Question 2.2

In [50]:

```
## Write a function that takes a string as an input and prints "Chipmunk"
## if the string is "Alvin", "Simon", or "Theodore".  Otherwise, it prints '

def chipmunk_or_witchdoctor (string):
    '''take a string and print chipmunk if the string is not "Alvin", "Simor
    if (string == "Alvin" or string == "Simon" or string == "Theodore"):
        print ("Darn! Go ask the witchdoctor!")
    else:
        print ("Chipmunk")
```

```
Chipmunk
Darn! Go ask the witchdoctor!
```

Question 2.3

In [51]:

```
## Here is code to make a table of ice cream flavors and a score reflecting

Kenneth_Ice_Cream = Table().with_columns(
                        "Flavor", make_array('Pistachio', 'Pineapple', 'Pepp
                                        "Score", make_array(10, 7, 5, 2, 8))

Kenneth_Ice_Cream
```

Out[51]:

Flavor	Score
Pistachio	10
Pineapple	7
Peppermint	5
Pop Rock	2
Pumpkin Spice	8

In [144…

```
## Write a function that takes an icecream flavor as an input and returns my
## Extra Credit: If a flavor is not in the table, print "Kenneth has no opir
def icecream_score (flavor):
    '''tale a icecream flavor and return my score'''
    for i in np.arange(5):
        if (flavor == Kenneth_Ice_Cream["Flavor"].item(i)):
            return Kenneth_Ice_Cream["Score"].item(i)
        else:
            no = "Kenneth has no opinion."
            return no

print (icecream_score("Pistachio"))
print (icecream_score("Choce"))
```

```
10
Kenneth has no opinion.
```

Section 3 - Running a simulation

In [61]:

```
## The following table lists the amount of weight an athlete can lift (in ki
## and the distance they achieved in the shot put competition (in meters).

shotput = Table().read_table('/srv/data/DS_113_S23/Tests/shotput.csv')
shotput
```

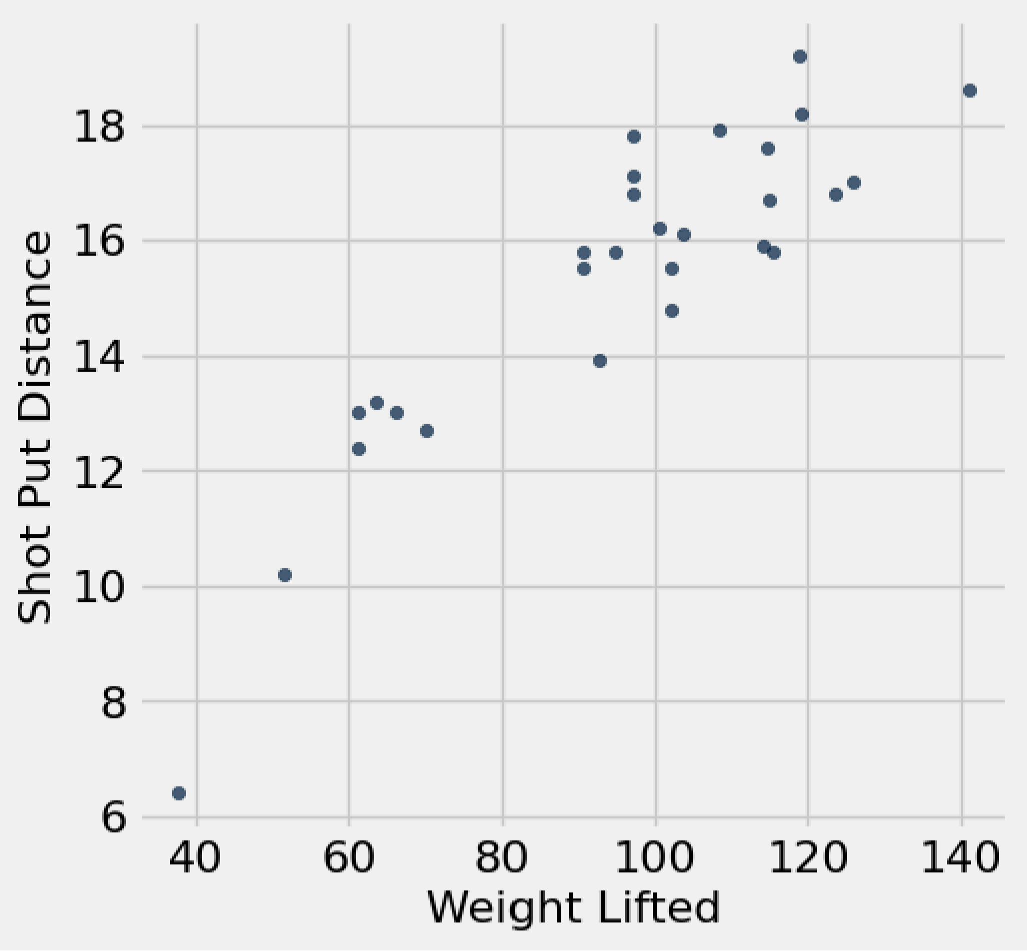
Out[61]:

Weight Lifted	Shot Put Distance
37.5	6.4
51.5	10.2
61.3	12.4
61.3	13
63.6	13.2
66.1	13
70	12.7
92.7	13.9
90.5	15.5
90.5	15.8

... (18 rows omitted)

Question 3.1

```
In [62]: ## Please make a scatter plot of the data with "Weight Lifted" on the x-axis
         ## Based on your graph, how far would you expect someone who could lift 80 k
         ## What about your graph suggests that you can make such a prediction with s
         shotput.scatter("Weight Lifted", "Shot Put Distance")
```



We can expect a person who could lift 80kg to throw a shot put about 14 or 15 meters away. Since we can see the positive relationship between the weight ligted and the shot put distance. And because the people who lifted about 60-70kg threw the shot put about 12 to 13 meters and the people who lifter about 90-100 kg threw the shot put about 14-16 meters. So, we can predict that a person who lifted 80 kg would throw a shot put 14 to 15 meters far.

Question 3.2

```
In [83]: ## Suppose the data are for my shot put team.
         ## On any given competition day, I do not know which five athletes might sho
         ## Please write code to find the total distance thrown by a random group of
         sum(shotput.sample(5, with_replacement=False).column("Shot Put Distance"))

Out[83]: 73.0
```

Question 3.3

```
In [130…  ## Write code to simulate the process in question 3.2 1000 times (i.e. take
          ## Store the total distance from each simulation in an array called total_di
          total_distances = make_array()
```

```
for i in np.arange(1000):
    distance_sum = sum(shotput.sample(5, with_replacement=False).column("Sho
    total_distances = np.append(total_distances,distance_sum)

total_distances
```

```
Out[130]: array([ 81.5,  77.5,  71.4, ...,  81.5,  75.2,
                  81.6])
```

Question 3.4

```
In [139…  ## Suppose my team will progress to nationals if we have a combined distance
          ## Make a histogram of your array of distances.  By looking at the histogram
          ## that my team will make nationals.
          total_distances.hist()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In [139], line 4
      1 ## Suppose my team will progress to nationals if we have a combined
distance of 75 meters.
      2 ## Make a histogram of your array of distances.  By looking at the
histogram, estimate the probability
      3 ## that my team will make nationals.
----> 4 total_distances.hist()

AttributeError: 'numpy.ndarray' object has no attribute 'hist'
```

Question 3.5 (harder)

```
In [145…  ## Improve your answer from above by changing the bins for the histogram in
          ## estimate the probability that our total distance is greater than 75.
          total_distances.hist(bins=10)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In [145], line 3
      1 ## Improve your answer from above by changing the bins for the hist
ogram in such a way as to improve your ability to
      2 ## estimate the probability that our total distance is greater than
75.
----> 3 total_distances.hist(bins=10)

AttributeError: 'numpy.ndarray' object has no attribute 'hist'
```

Extra Challenge

```
In [105…  ## Write a function to do the following using the house data:

          # Take as input a specified number of full bathrooms ("Full Bath") and bedro
          # and report the neighborhood with the highest average age (years since buil
          # homes that meet those conditions.

          def highest_avg_age (bathrooms, bedrooms):
              '''take a number of full bathrooms and bedrooms and report the neighborh
              bathroom = house_data.where("Full Bath", are.equal_to(bathrooms))
              bedrooms = bathroom.where("Bedroom AbvGr", are.equal_to(bedrooms))
              neighborhood_age = Table().with_columns("Neighborhood", bedrooms["Neighb
                                                      "Age", 2023-bedrooms["Year Built
```

```
    avg_age = neighborhood_age.group("Neighborhood",np.average).sort("Age av

    return avg_age["Neighborhood"].item(0)

highest_avg_age(1,2)
```

```
Out[105]: 'IDOTRR'
```