In [1]:
```
# imports
import math
import pandas as pd
import numpy as np
import statistics as stats
import stemgraphic
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Useful functions I'll be utilizing later.
# Generates dataframes based on frequency data
def data_generator(value_vector, frequency_vector):
    data_vec = []
    n = 0
    for value in value_vector:
        data_vec.extend([value] * frequency_vector[n])
        n = n+1
    df = pd.DataFrame(data=data_vec)
    return df
```

Brady Lamson (blamson@msudenver.edu)

September 4, 2021

MTH 3210 Written Homework 2

1.1: 1-2, 4

Note: Section 1.1 will not be in complete sentences as I find a list of answers conveys information better here.

1. Give one possible sample of size 4 from each of the following populations:

- All daily newspapers published in the United States.
  - The Wall Street Journal
  - USA Today
  - The Independent
  - New York Post
- All companies listed on the New York Stock Exchange.
  - Autodesk Inc
  - Barnes Group Inc
  - Genpact Limited
  - Zillow Group Incorporated
- All students at your college or university
  - Any four students at the university.
- All grade point averages of students at your college or university
  - 3.5
  - 2.4
  - 4.0
  - 3.0

2. For each of the following hypothetical populations, give a plausible sample of size 4.

- All distances that might result when you throw a football.

    - 100 feet
    - 120 feet
    - 87 feet
    - 150 feet
- Page lengths of books published 5 years from now
    - 242 pages
    - 500 pages
    - 450 pages
    - 315 pages
- All possible earthquake-strength measurements (Richter scale) that might be recorded in California during the next year.
    - 4.3
    - 7.0
    - 5.1
    - 6.5
- All possible yields (in grams) from a certain chemical reaction carried out in a laboratory.
    - 3g
    - 12g
    - 2g
    - 15g

4a. Give three different examples of concrete populations and three different examples of hypothetical populations.

- Concrete Populations:
    - The number of people who registered for fall 2021 classes at MSU Denver.
    - The number of people who attended the Renaissance Festival on pirate weekend.
    - The number of eggs sold by Kroger tomorrow.
- Hypothetical populations:
    - All individuals with math degrees.
    - All registered Democrats.
    - The full population of Denver.

4b. For one each of your concrete and your hypothetical populations, give an example of a probability question and an example of an inferential statistics question.

- Concrete Population:
    - What is the likelihood of a class at MSU Denver being made up of primarily unvaccinated students?
    - How many students feel confident going into the fall semester?
- Hypothetical Population:
    - What is the probability of a citizen of Denver living within two miles of Union Station?
    - How many citizens of Denver utilize public transportation?

1.2: 12, 17-21, 24, 26

12.

The accompanying summary data on CeO_2 particle sizes (nm) under certain experimental conditions was read from a graph in the article **"Nanoceria - Energetics of Surfaces, Interfaces and Water Absorption" (*J. of the***

***Amer. Ceramic Soc.*, 2011: 3992-3999):**

Particle Size (nm)	Frequency (n=131)
[3.0, 3.5)	5
[3.5, 4.0)	15
[4.0, 4.5)	27
[4.5, 5.0)	34
[5.0, 5.5)	22
[5.5, 6.0)	14
[6.0, 6.5)	7
[6.5, 7.0)	2
[7.0, 7.5)	4
[7.5, 8.0)	1

a. What proportion of the observations are less than 5.

Here we add up our values in the first four rows and divide by n. $\frac{81}{131} \approx 0.618$, so that means approximately 62% of observations are less than 5.

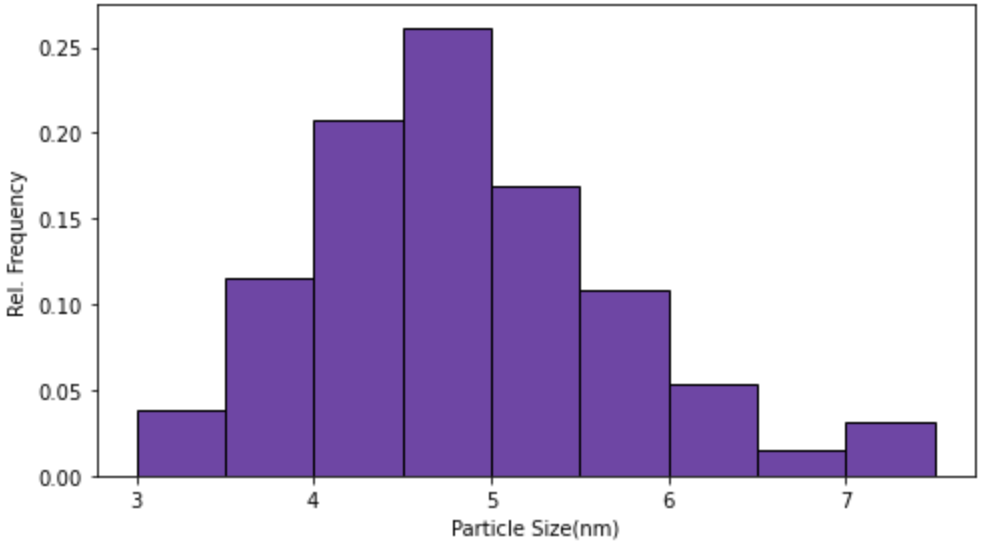
b.What proportion of the observations are at *least* 6.

Here we add the values in the bottom four rows and divide by n. $\frac{14}{131} \approx 0.107$, so we know that approximately 10.7% of observations are at least 6.

c. Construct a histogram with relative frequency on the vertical axis and comment on interesting features. In particular, does the distribution of particle sizes appear to be reasonably symmetric or somewhat skewed?

In [3]:

```
particle_size = [3.2, 3.7, 4.2, 4.7, 5.2, 5.7, 6.2, 6.7, 7.2, 7.7]
frequency = [5,15,27,34,22,14,7,2,4,1]
size_df = data_generator(particle_size, frequency)
size_df.rename(columns={ size_df.columns[0]: "Particle Size" }, inplace=True)

plt.rcParams["figure.figsize"] = [7.00, 4]
plt.rcParams["figure.autolayout"] = True
bins=np.arange(3,8,step=0.5)
hist, edges = np.histogram(size_df['Particle Size'], bins)
freq = hist/float(hist.sum())

plt.bar(bins[:-1], freq, width=0.5, align="edge", ec="k", color='#6E46A4')
plt.xlabel('Particle Size(nm)')
plt.ylabel('Rel. Frequency')
plt.show()
```



This distribution is mostly **bell shaped** and is slightly skewed to the **right**.

17.

The accompanying data came from a study of collusion in bidding within the construction industry. ("Detection of Collusive Behavior," J. *of Construction Engr. and Mgmnt,* 2012: 1251-1258).

# Bidders	# Contracts
2	7
3	20
4	26
5	16
6	11
7	9
8	6
9	8
10	3
11	2

a. What Proportion of the contracts involved at *most* five bidders? At *least* five bidders?

To calculate the first part we need to add up the number of contracts in the first four columns and then divide it by the total number of contracts.

$$7 + 20 + 26 + 16 = 69$$

$$69 + 11 + 9 + 6 + 8 + 3 + 2 = 108$$

$$\boxed{\frac{69}{108} \approx 0.639}$$

From this we can say that the number of contracts that involved at *most* five bidders is approximately 64% of the total sample.
As for how many had at *least* five we can take the total sample and subtract out all of the contracts below 5 bidders.

$$108 - 26 - 20 - 7 = 55$$

$$\boxed{\frac{55}{108} \approx 0.509}$$

So, approximately 50.9% of contracts had at least 5 bidders.

b. What proportion of the contracts involved *between* five and ten bidders, inclusive? Strictly between five and ten bidders (noninclusive)?
You know the drill from here. Since the first part is inclusive we subtract out the contracts that don't meet the criteria and divide.

$$108 - 2 - 26 - 20 - 7 = 53$$

$$\boxed{\frac{53}{108} \approx 0.491}$$

Approximately 49.1% of contracts are between 5 and 10 bidders. As for the noninclusive amount, subtract out the contracts at 5 and 10 bidders.

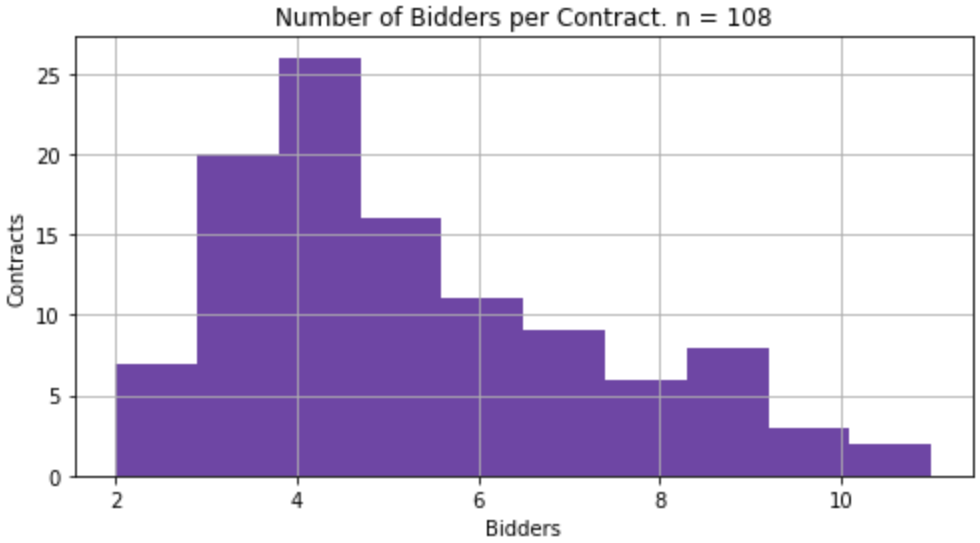
$$53 - 3 - 16 = 34$$

$$\boxed{\frac{34}{108} \approx 0.315}$$

When the range is noninclusive, approximately 31.5% of contracts are strictly between 5 and 10 bidders.
c. Construct a histogram and comment on interesting features.

In [4]:
```
num_bidders = [2,3,4,5,6,7,8,9,10,11]
num_contracts = [7,20,26,16,11,9,6,8,3,2]
bid_df = data_generator(num_bidders, num_contracts)

bid_df.plot.hist(grid=True, bins=10, color='#6E46A4', legend=False)
plt.title('Number of Bidders per Contract. n = 108')
plt.xlabel('Bidders')
plt.ylabel('Contracts')
```

Out[4]:
```
Text(0, 0.5, 'Contracts')
```



What's interesting about this plot is that while it seems **unimodal**, it's actually technically **bimodal**, though just barely. There's a slight little bumb at 9 bidders. This distribution is also right skewed.

18.

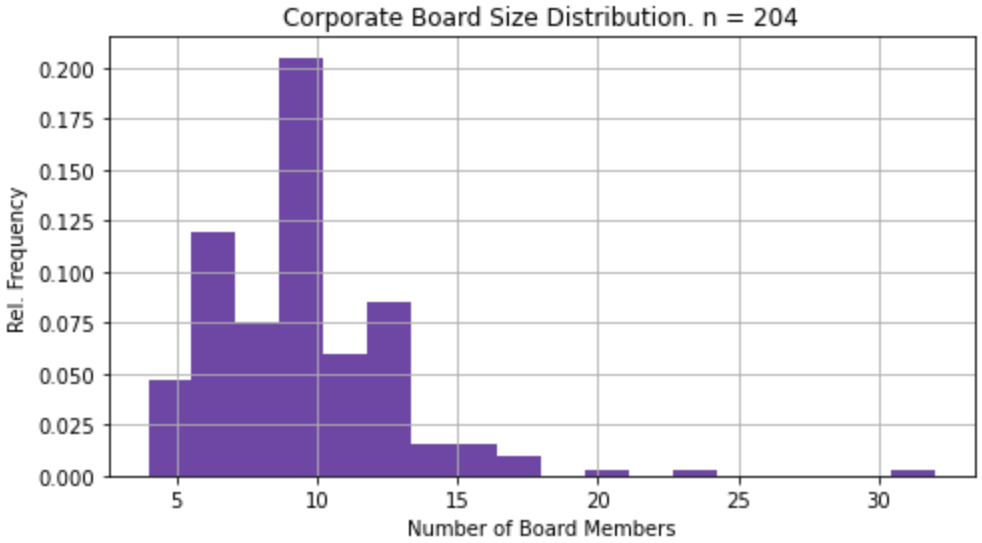
Every corporation has a governing board of directors. The number of individuals on a board varies from one corporation to another. One of the authors of "Does Optimal Corporate Board Size Exist? An Emperical Analysis" (*J of Applied Finance,* 2010: 57-69) provided the accompanying data on the number of directors on each board in a random sample of 204 corporations.

# Directors	Frequency
4	3
5	12
6	13
7	25
8	24
9	42
10	23
11	19
12	16
13	11
14	5
15	4
16	1
17	3
21	1
24	1
32	1

a. Construct a histogram of the data based on relative frequencies and comment on any interesting features.

In [5]:
```
director_count = [4,5,6,7,8,9,10,11,12,13,14,15,16,17,21,24,32]
frequency = [3,12,13,25,24,42,23,19,16,11,5,4,1,3,1,1,1]

director_df = data_generator(director_count, frequency)

director_df.plot.hist(grid=True, color='#6E46A4', bins=len(frequency)+1,
                      legend=False, density=True)
plt.title('Corporate Board Size Distribution. n = 204')
plt.xlabel('Number of Board Members')
plt.ylabel('Rel. Frequency')
```

Out[5]:
```
Text(0, 0.5, 'Rel. Frequency')
```



This would probably be a multimodal distribution. It is ever so slightly skewed to the right. What's surprising is the extreme frequency of 9 board members. It seems to be a fairly arbitrary amount yet it dwarfs everything else in the dataset.

b. Construct a frequency distribution in which the last row includes all boards with at least 18 directors. If this distribution had appeared in the cited article, would you be able to draw a histogram? Explain.

# Directors	Frequency
21	1
24	1
32	1

You can *technically* create a histogram out of anything. This one would have been solidly useless though. A total sample of 204 and all we can work with are 3 of them? That would be an incredibly inaccurate presentation of the data.

c. What proportion of these corporations have *at most* 10 directors?

$$\frac{3+12+13+25+24+42+23}{204} \approx 0.696$$

Approximately 70% of corporations have at most 10 directors.

d. What proportion of these corporations have *more* than 15 directors>

$$\frac{7}{204} \approx 0.0343$$

Approximately 3.43% of corporations have more than 15 directors.

19.

The number of contaminating particles on a silicon wafer prior to a certain rinsing process was determined for each wafer in a sample of size 100, resulting in the following frequencies.

# of Particles	Frequency (n = 100)
0	1
1	2
2	3
3	12
4	11
5	15
6	18
7	10
8	12
9	4
10	5
11	3
12	1
13	2
14	1

a. What proportion of the sampled wafers had at *least* one particle? At *least* five particles?

$$\boxed{\frac{99}{100} \approx 0.999}$$

Approximately 99% of the sampled wafers had at least one particle.

$$\boxed{\frac{100 - 11 - 12 - 3 - 2 - 1}{100} \approx 0.71}$$

Approximately 71% of the sampled wafers had at least five particles.

b. What proportion of the sampled wafers had *between* five and ten particles, inclusive? Stricly between five and ten particles?

$$\boxed{\frac{15 + 18 + 10 + 12 + 4 + 5}{100} \approx 0.64}$$

Approximately 64% of the sampled wafers had between five and ten particles, inclusive.

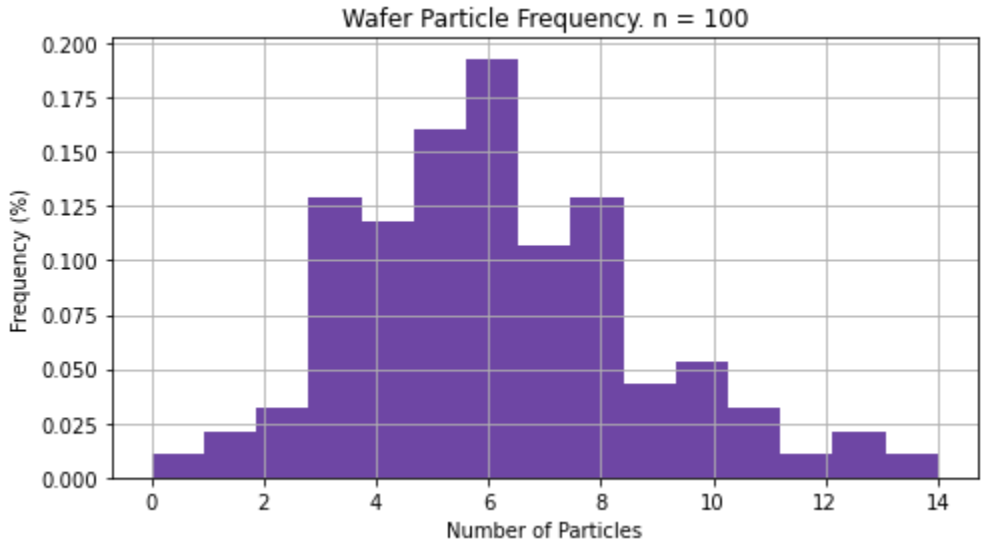
$$\boxed{\frac{64 - 15 - 5}{100} \approx 0.44}$$

Approximately 44% of the sampled wafers had strictly between five and ten particles.

c. Draw a histogram using relative frequency on the vertical axis. How would you describe the shape of the histogram?

In [6]:
```
particle_count = list(range(0,15))
frequency = [1,2,3,12,11,15,18,10,12,4,5,3,1,2,1]

wafer_df = data_generator(particle_count, frequency)
wafer_df.plot.hist(grid=True, color='#6E46A4', bins=len(frequency),
                   legend=False, density=True)
plt.title('Wafer Particle Frequency. n = 100')
plt.xlabel('Number of Particles')
plt.ylabel('Frequency (%)')
```

Out[6]:
```
Text(0, 0.5, 'Frequency (%)')
```



I would describe this histogram's distribution as multi-modal and bell-shaped.

20.

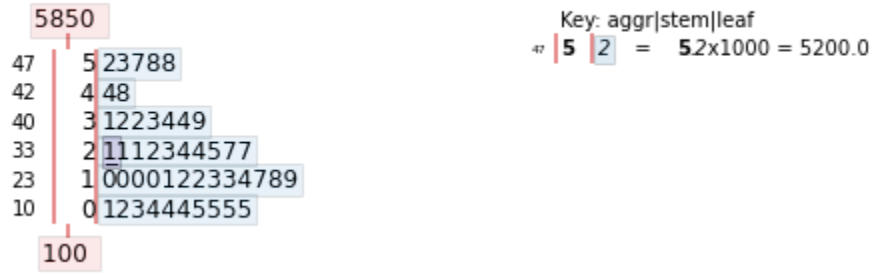
The article Determination of Most Representative Subdivision" (*J. of Energy Engr.*, 1993: 43-55) gave data on various characteristics of subdivisions that could be used in deciding whether to provide electrical power using overhead lines or underground lines. The values of the variable (x = total length of streets) will be listed below.

a. Construct a stem-and-leaf display using the thousands digit as the stem and the hundreds digit as the leaf, and comment on the various features of the display.

In [7]:
```
street_data = [1280, 5320, 4390, 2100, 1240, 3060, 4770, 1050, 360, 3330, 3380,
               340, 1000, 960, 1320, 530, 3350, 540, 3870, 1250, 2400, 960,
```

```
                1120, 2120, 450, 2250, 2320, 2400, 3150, 5700, 5220, 500, 1850,
                2460, 5850, 2700, 2730, 1670, 100, 5770, 3150, 1890, 510, 240,
                396, 1419, 2109]
stemgraphic.stem_graphic(street_data, scale = 1000)
```

Out[7]:
```
(<Figure size 540x162 with 1 Axes>,
 <matplotlib.axes._axes.Axes at 0x7f48a0758580>)
```

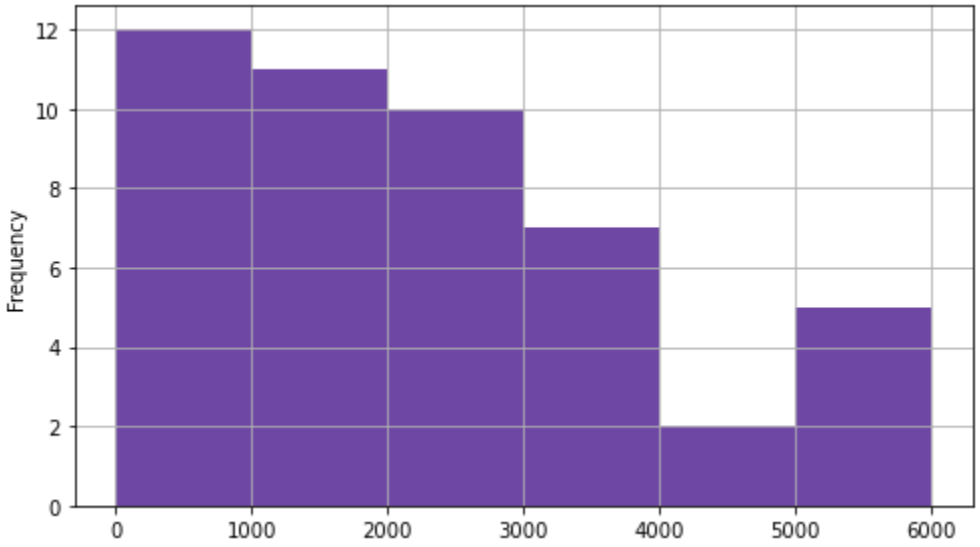


The thing that immediately stands out to me is the low frequency of streets with a length between 4 and 5 thousand. It also seems that most subdivisions tend to trend shorter in total length.

b. Construct a histogram using class boundaries 0, 1000, 2000, 3000, 4000, 5000, and 6000. What proportion of subdivisions have total length less than 2000? Between 2000 and 4000? How would you describe the shape of the histogram?

In [8]:
```
street_df = pd.DataFrame(data=street_data)
street_df.plot.hist(grid=True, color='#6E46A4',
                    bins=[0,1000,2000,3000,4000,5000,6000], legend=False)
```

Out[8]:
```
<AxesSubplot:ylabel='Frequency'>
```



What we can see from the histogram is that 23 subdivisions have a total length of less than 2000 which is approximately 48.9%.
What we can also see is that 17 subdivisions have a total length between 2000 and 4000. Thats approximately 36.2%

$$\boxed{\frac{23}{47} \approx 0.489} \quad \boxed{\frac{17}{47} \approx 0.362}$$

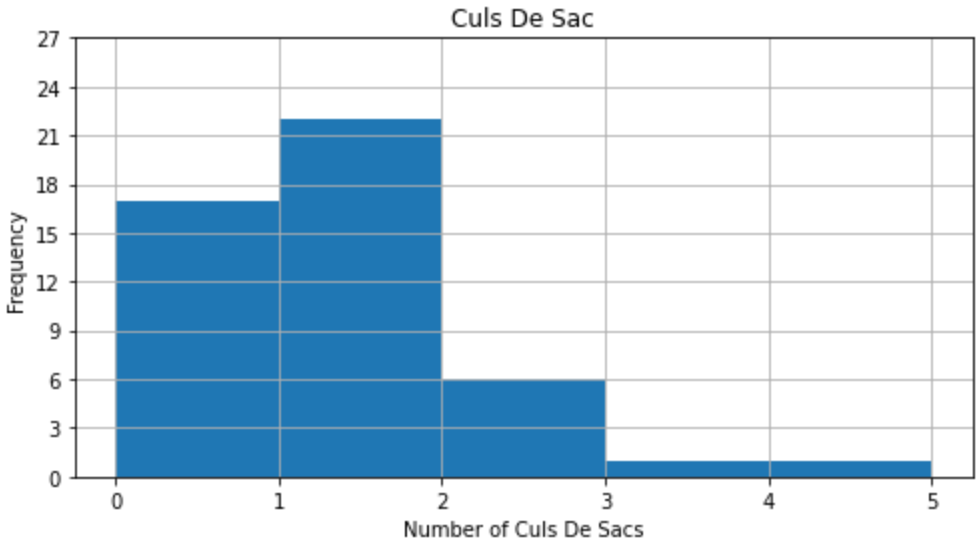
The shape of this histogram can roughly be described as a reverse J-shape.

21.

The article cited in exercise 20 also gave the following values of the variables y = number of culs-de-sac and z = number of intersections.

In [9]:
```
culs_de_sac = [1,0,1,0,0,2,0,1,1,1,2,1,0,0,1,1,0,1,1,1,1,0,
               0,0,1,1,2,0,1,2,2,1,1,0,2,1,1,0,1,5,0,3,0,1,1,0,0]
intersection_count = [1,8,6,1,1,5,3,0,0,4,4,0,0,1,2,1,4,0,
                      4,0,3,0,1,1,0,1,3,2,4,6,6,0,1,1,8,3,3,5,0,
                      5,2,3,1,0,0,0,3]

street_df_2 = pd.DataFrame()
street_df_2['Culs De Sac'] = culs_de_sac
street_df_2['Intersections'] = intersection_count

street_df_2.hist(column='Culs De Sac', bins=[0,1,2,3,4,5])
plt.yticks(np.arange(0, 30, step=3))
plt.xlabel('Number of Culs De Sacs')
plt.ylabel('Frequency')
```

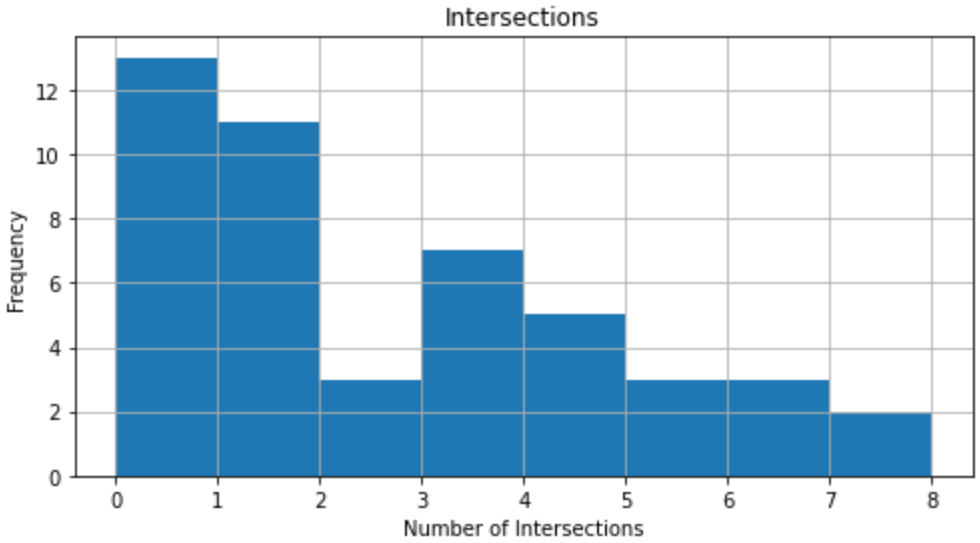
Out[9]:
```
Text(0, 0.5, 'Frequency')
```



a. What proprotion of these subdivisions had no culs-de-sacs? At least one cul-de-sac?
17 out of the 47 subdivisions had no culs-de-sacs. That's approximately 36.2%.
Meanwhile, 30 out of the 47 subdivisions had at least one of them. That's approximately 63.8%.

In [10]:
```
street_df_2.hist(column='Intersections', bins=[0,1,2,3,4,5,6,7,8])
plt.xlabel('Number of Intersections')
plt.ylabel('Frequency')
```

Out[10]:
```
Text(0, 0.5, 'Frequency')
```



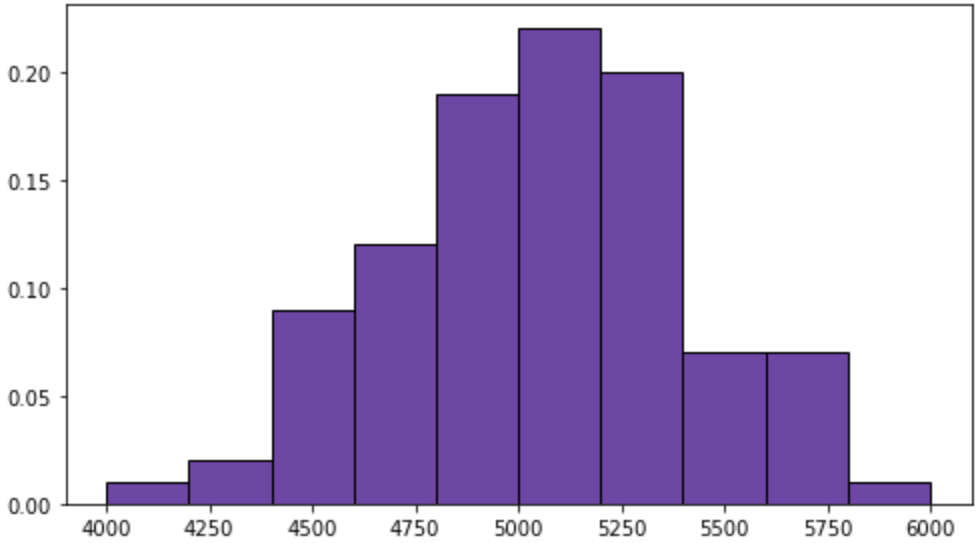
a. What proprotion of these subdivisions had at most five intersections? Fewer than 5 intersections?
5 subdivisions had more than 5 intersections, so that gives us 42 subdivisions that meet our criteria. Approximately 89.4% of subdivisions had at most 5 intersections.
As for ones with less than 5 intersections, that would be 39 of them. So approximately 83.0% meet that criteria.

24.

The accompanying data set consists of observations on shear strength(lb) of ultrasonic spot welds made on a certain type of alclad sheet. Construct a relative frequency histogram based on ten equal-width classes with boundaries 4000, 4200,... Comment on its Features.

In [11]:

```
my_set = [5434,5112,4820,5378,5027,4848,4755,5207,5049,4740,5248,5227,4931,5364,5189,
          4948,5015,5043,5260,5008,5089,4925,5621,4974,5173,5245,5555,4493,5640,4986,
          4521,4659,4886,5055,4609,5518,5001,4918,4592,4568,4723,5388,5309,5069,4570,
          4806,4599,5828,4772,5333,4803,5138,4173,5653,5275,5498,5582,5188,4990,4637,
          5288,5218,5133,5164,4951,4786,5296,5078,5419,4681,4308,5764,5702,5670,5299,
          4859,5095,5342,5679,4500,4965,4900,5205,5076,4823,5273,5241,4381,4848,4780,
          4618,5069,5256,5461,5170,4968,4452,4774,4417,5042]

strength_df = pd.DataFrame()
strength_df['Shear Strength'] = my_set

plt.rcParams["figure.figsize"] = [7.00, 4]
plt.rcParams["figure.autolayout"] = True
bins=np.arange(4000,6200,step=200)
hist, edges = np.histogram(my_set, bins)
freq = hist/float(hist.sum())

plt.bar(bins[:-1], freq, width=200, align="edge", ec="k", color='#6E46A4')
plt.show()
```



This histogram is actually a very neat unimodal, bell shaped distribution. Most of the shear strengths fall above 4800lbs.

26.

Automated electron backscattered diffraction is now being used in the study of fracture phenomena. The following information on misorientation angle (degrees) was extracted from the article "Observations on the Faceted Initiation Site in the Dwell-Fatigue Tested Ti-6242 Alloy: Crystallographic Orientation and Size Effects" (*Metallurgical and Materials Trans., 2006: 1507-1518*)

Class	Rel. Freq.
[0, 5)	0.177
[5, 10)	0.166
[10, 15)	0.175
[15, 20)	0.136
[20, 30)	0.194
[30, 40)	0.078
[40, 60)	0.044
[60, 90)	0.030

a. Is it true that more than 50% of the sampled angles are smaller than 15° , as asserted in the paper?
For this just add up the relative frequencies in the first three rows. These rows add up to 0.518, which does mean that 50% are infact smaller than 15°.

b. What proportion of the sampled angles are at least 30°?
Approximately 15.2% of the sampled angles are at least 30°.

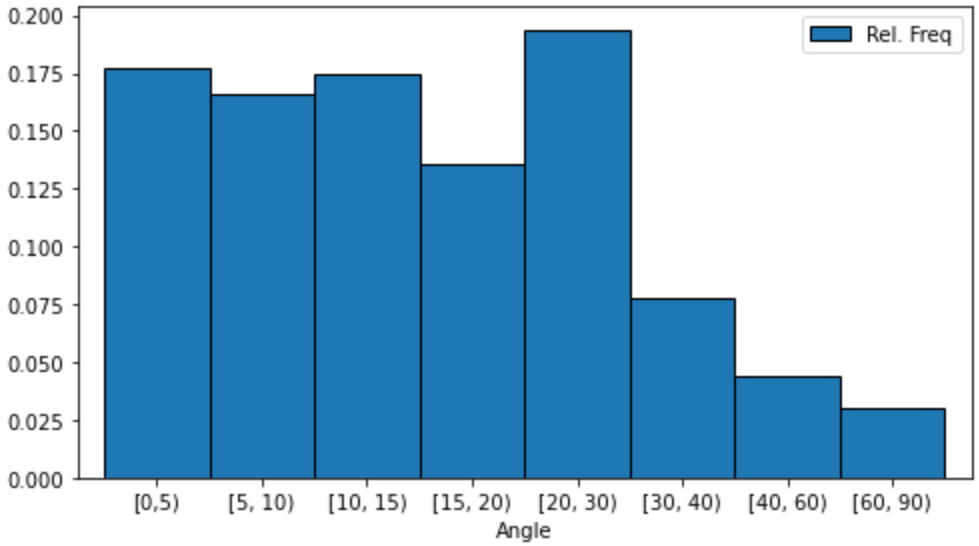
c. Roughly what proportion of angles are between 10° and 25°?
Slightly less than 50.5% as my calculation also included angles from 26 to 30.

d. Construct a histogram and comment on any interesting features.

In [12]:
```
angle_df = pd.DataFrame({
    'Angle': ['[0,5)', '[5, 10)', '[10, 15)', '[15, 20)', '[20, 30)',
```

```
                '[30, 40)', '[40, 60)', '[60, 90)']})

freq_df = pd.DataFrame({
    'Rel. Freq': [0.177, 0.166, 0.175, 0.136, 0.194, 0.078, 0.044, 0.030]})

misorientation_df = pd.concat([angle_df, freq_df], axis=1)

misorientation_df.plot.bar(x='Angle', y='Rel. Freq', rot=0,
                           width=1, edgecolor='black')
```

Out[12]: <AxesSubplot:xlabel='Angle'>



1.3: 34(a)(b), 35(a)(c), 36, 38, 39

34 (a)(b).

Exposure to microbial products, especially endotoxin, may have an impact on vulnerability to allergic diseases. The article "Dust Sample Methods for Endotoxin - An Essential, but Underestimated Issue" (*Indoor Air,* 2006: 20-27) considered various issues associated with determining endotoxin concentration. The following data on concentration (EU/mg) in settled dust for one sample of urban homes and another of farm homes was kindly supplied by the authors of the cited article.

Urban Sample	Farm Sample
6.0	4.0
5.0	14.0
11.0	11.0
33.0	9.0
4.0	9.0
5.0	8.0
80.0	4.0
18.0	20.0
35.0	5.0
17.0	8.9
23.0	21.0

Urban Sample	Farm Sample
	9.2
	3.0
	2.0
	0.3

a. Determine the sample mean for each sample. How do they compare?
First, it helps to define standard mean. We will let sample mean = $\overline{x}$.

$$\text{Urban Sample Mean} = \overline{x_u}$$

$$\text{Farm Sample Mean} = \overline{x_f}$$

Sample mean is calculated as follows:

$$\overline{x} = \frac{x_1 + x_2 + x_3 + \ldots + x_n}{n}$$

I'll handle the rest in the below python code.

In [13]:
```
farm_set = [4,14,11,9,9,8,4,20,5,8.9,21,9.2,3,2,0.3]
urban_set = [6,5,11,33,4,5,80,18,35,17,23]

# The traditional way to calculate this.
print('The Farm Sample Mean is ', (round(sum(farm_set) / len(farm_set), 3)))
print('The Urban Sample Mean is ', (round(sum(urban_set) / len(urban_set), 3)))

# Calculating the mean using numpy's mean method to show equivelance.
print('The Farm Sample Mean using numpy is ', round(np.mean(farm_set), 3))
print('The Urban Sample Mean using numpy is ', round(np.mean(urban_set), 3))
```

```
The Farm Sample Mean is  8.56
The Urban Sample Mean is  21.545
The Farm Sample Mean using numpy is  8.56
The Urban Sample Mean using numpy is  21.545
```

As we can see from the code, $\boxed{\overline{x_u} \approx 21.55}$ and $\boxed{\overline{x_f} \approx 8.56}$. The sample mean from the urban sample is far larger.

b. Determine the sample median for each sample. How do they compare? Why is the median for the urban sample so different from its mean?
The median value is simply the middle of the data set when ordered. I'll go ahead and use numpy to calculate that for both sets.

In [14]:
```
print('The Farm Sample Median is', np.median(farm_set))
print('The Urban Sample Median is', np.median(urban_set))
```

```
The Farm Sample Median is 8.9
The Urban Sample Median is 17.0
```

The sample median for the urban sample is still far larger. It's also a lot smaller than the urban mean. This is because of outliers, like 80, inflating the mean.

---

35. (a)(c)

Mercury is a persistent and dispersive environmental contaminant found in many ecosystems around the world. When released as an industrial by-product, it often finds its way into aquatic systems where it can have deleterious effects on various avian and aquatic species. The accompanying data on blood mercury concentration (μg/g) for adult females near contaminated rivers in Virginia was read from a graph in the article "Mercury Exposure Effects the Repreoductive Success of a Free-Living Terrestrial Songbird, the Carolina Wren" (*The Auk*, 2011: 759-769); this is a publication of the American Orinthologists' Union.

Samples part 1	Samples part 2
.20	1.42
.22	1.70
.25	1.83
.30	2.20
.34	2.25
.41	3.07
.55	3.25

a. Determine the values of the sample mean and sample median and explain why they are different. (Hint: $\sum x_1 = 18.55$)

In [15]:
```
mercury_set = [.2, .22, .25, .3, .34, .41, .55, 1.42, 1.7, 1.83, 2.20, 2.25,
               3.07, 3.25]
print('The sample mean is approximately', round(np.mean(mercury_set), 3))
print('The sample median is approximately', np.median(mercury_set))
```

```
The sample mean is approximately 1.285
The sample median is approximately 0.985
```

The reason these values differ is due to the amount of spread in the data. The mean is inflated ever so slightly by the last two values that are far larger than the rest. The median, meanwhile, is completely unaffected the largest values in the data set.

c. By how much could the observation .20 be increased without impacting the value of the sample median?
To answer this we need to know how we're getting the median in the first place. Through direct observation we can tell that the two values being averaged are 0.55 and 1.42. Therefore, we can increase our first observation all the way up to 0.55 without changing the median. We can see this in the python code below.

In [16]:
```
mercury_set[0]=0.55
print('The sample median is approximately', np.median(mercury_set))
mercury_set[0]=0.56
print('The new sample median is approximately', np.median(mercury_set))
```

```
The sample median is approximately 0.985
The new sample median is approximately 0.99
```

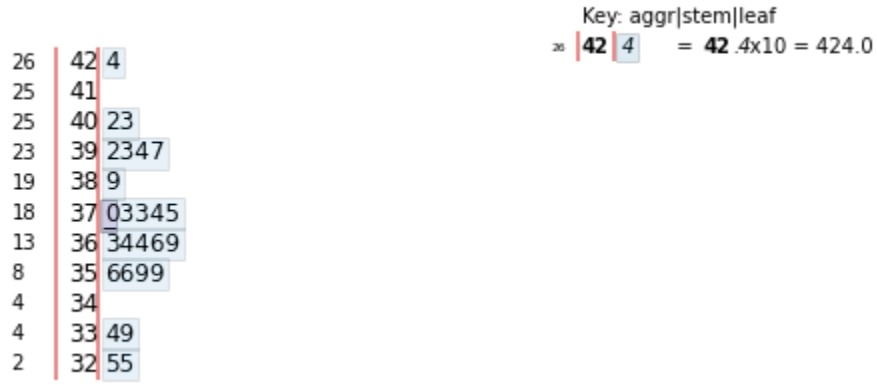
36.

A sample of 26 offshore oil workers took part in a simulated escape exercise, resulting in the accompanying data on time (seconds) to complete the escape. ("Oxygen Consumption and Ventilation During Escape from an Offshore Platform," *Ergonomics,* 1997: 281-292):

a. Construct a stem-and-leaf display of the below data. How does it suggest that the sample mean and median will compare?

In [17]:
```
escape_times = [389,356,359,363,375,424,325,394,402,373,373,370,364,366,364,325,339,393,39
                369,374,359,356,403,334,397]

stemgraphic.stem_graphic(escape_times)
```

Out[17]:
```
(<Figure size 540x252 with 1 Axes>,
 <matplotlib.axes._axes.Axes at 0x7f48a04840a0>)
```



I believe it suggests that they'll be pretty close. Based on visual inspection I would assume that the mean will be slightly lower than the median due to the abundance of values below the median.

b. Calculate the sample mean ($\bar{x}$) and the sample median ($\tilde{x}$)

In [18]:
```
print('The sample mean is approximately',
      round(np.mean(escape_times), 2), 'seconds')
print('The sample median is approximately',
      np.median(escape_times), 'seconds')
```

```
The sample mean is approximately 370.69 seconds
The sample median is approximately 369.5 seconds
```

c. By how much could the largest time, currently 424, be increased without affecting the value of the sample median? By how much could this value be decreased without affecting the value of the sample median?
Before we can answer this we need to know the two values being averaged for the median. In this set its 370 and 369. For the first part of this problem, the largest value can increase indefinitely without affecting the median. It would need to decrease to below 370 to affect the median.

d. What are the values of $\bar{x}$ and $\tilde{x}$ when the observations are reexpressed in minutes?
To do this let us divide our current list of values by 60 and then recalculate! We can compare this to

simply dividing the current mean and median by 60 and see if there's any difference in methodology.

In [19]:
```
escape_minutes = [round((escape / 60), 2) for escape in escape_times]
print(escape_minutes)

print('The sample mean is approximately',
      round(np.mean(escape_minutes), 2), 'minutes')
print('The sample median is approximately',
      np.median(escape_minutes), 'minutes')
```

```
[6.48, 5.93, 5.98, 6.05, 6.25, 7.07, 5.42, 6.57, 6.7, 6.22, 6.22, 6.17, 6.07, 6.1, 6.07,
5.42, 5.65, 6.55, 6.53, 6.15, 6.23, 5.98, 5.93, 6.72, 5.57, 6.62]
The sample mean is approximately 6.18 minutes
The sample median is approximately 6.16 minutes
```

The values we got are equivelant to if we simply divided our previous answers by 60.

$\boxed{\frac{370.69}{60} \approx 6.18}$ **and** $\boxed{\frac{369.5}{60} \approx 6.16}$**.**

This is unsurprising but also nice to confirm as a sanity check.

38.

Blood pressure values are often reported to the nearest 5 mmHg (100, 105, 110, etc.). Suppose the actual blood pressure values for nine randomly selected individuals are:

[118.6, 127.4, 138.4, 130.0, 113.7, 122.0, 108.3, 131.5, 133.2]

a. What is the median of the *reported* blood pressure values?

First we need to round our values to the nearest 5th integer. So for example 118.6 should become 120. After that we will calculate median as normal.

In [20]:
```
blood_pressure = [118.6, 127.4, 138.4, 130.0, 113.7, 122.0, 108.3, 131.5, 133.2]
base = 5 # This is the integer we want to base our rounding off of.

reported_blood_pressure = [(base * round(value / base))
                           for value in blood_pressure]
print(sorted(reported_blood_pressure))

print('The reported blood pressure median is',
      np.median(reported_blood_pressure))
```

```
[110, 115, 120, 120, 125, 130, 130, 135, 140]
The reported blood pressure median is 125.0
```

To explain the above code let me showcase it using our first value, 118.6. Remember to round the quotient before multiplying by the base.

$$\tilde{x} = 5 * \left(\frac{118.6}{5}\right)$$

$$\tilde{x} \approx: 5 * 24$$

$$\tilde{x} \approx: 120$$

So, we do this for the entire data set. And from there our sample median is 125.

b. Suppose the blood pressure of the second individual is 127.6 rather than 127.4 (a small change in a single value). How does this affect the median of the reported values? What does this say about the sensitivity of the median to rounding or grouping in the data?

We can easily test with a quick modification to our original data set and a rerun of our code.

In [21]:
```
blood_pressure[1] = 127.6
test_blood_vec = [(base * round(value / base)) for value in blood_pressure]
print(sorted(test_blood_vec))
print('The reported blood pressure median is', np.median(test_blood_vec))

print(base * round(127.4 / base))
```

```
[110, 115, 120, 120, 130, 130, 130, 135, 140]
The reported blood pressure median is 130.0
125
```

What we see is a jump by 5 for our median, which is quite large. Going back and working through the math by hand shows exactly what's going on.

$$\tilde{x} = 5 * \left(\frac{127.4}{5}\right)$$

$$\tilde{x} \approx: 5 * 25.48 \text{ (rounded down to 25)}$$

$$\tilde{x} \approx: 125$$

$$\tilde{x} = 5 * \left(\frac{127.6}{5}\right)$$

$$\tilde{x} \approx: 5 * 25.52 \text{ (rounded up to 26)}$$

$$\tilde{x} \approx: 130$$

So, a simple change in .04 to a value resulted in a median change of 5. I don't think this means that rounding in that nature is a bad idea inherently, but that kind of sensitivity is absolutely something to be aware of.

39.

The propagation of fatigue cracks in various aircraft parts has been the subject of extensive study in recent years. The accompanying data consists of propagation lives $\left(\frac{\text{flight hours}}{10^4}\right)$ to reach a given crack size in fastener holes intended for use in military aircraft (Statistical Crack Propagation in Fastener Holes Under Spectrum Loading," *J. Aircraft,* 1983: 1028-1032):

[.736, .863, .865, .913, .915, .937, .983, 1.007, 1.011, 1.064, 1.109, 1.132, 1.140, 1.153, 1.253, 1.394]

a. Compute and compare the values of the sample mean and median.

In [22]:
```
propagation_lives = [.736, .863, .865, .913, .915, .937, .983, 1.007, 1.011,
                     1.064, 1.109, 1.132, 1.140, 1.153, 1.253, 1.394]

print('The sample mean is approximately',
      round(np.mean(propagation_lives), 2))

print('The sample median is approximately',
      np.median(propagation_lives))
```

```
The sample mean is approximately 1.03
The sample median is approximately 1.009
```

These values are incredibly close, though the mean is influenced enough by the upper range of values that it ends up higher than the median.

b. By how much could the largest sample observation be decreased without affecting the value of the median?
The two values being averaged here are 1.007 and 1.011, so that means the largest sample could go as low as 1.011 without affecting the median.

1.4: 44(a)(b)(c), 46, 47(a)(b), 53, 54(a)(b)(c), 59(a)(c), 60

44.

Poly(3-hydroxybutyrate) (PHB), a semicrystalline polymer that is fully biodegradable and biocompatible is obtained from renewable resources. From a sustainability perspective, PHB offers many attractive properties though it is more expensive to produce than standard plastics. The accompanying data on melting point ($^\circ C$) for each of 12 specimens of the polymer using a differential scanning calorimeter appeared in the article "The Melting Behavior of Poly(3-Hydroxybutyrate) by DSC. Reproducibility Study" (*Polymer Testing,* 2013: 215-220).

[180.5, 181.7, 180.9, 181.6, 182.6, 181.6, 181.3, 182.1, 182.1, 180.3, 181.7, 180.5]

Compute the following:

a. The sample range.
First let us define range. It is simply the difference between the highest and lowest values in a dataset. Let us sort the data-set and then tackle the subtraction.

In [23]:
```
phb_set = sorted([180.5, 181.7, 180.9, 181.6, 182.6, 181.6, 181.3,
                  182.1, 182.1, 180.3, 181.7, 180.5])
print(phb_set)
```

```
print('The range of this set is approximately',
      round(max(phb_set) - min(phb_set), 2))
```

```
[180.3, 180.5, 180.5, 180.9, 181.3, 181.6, 181.6, 181.7, 181.7, 182.1, 182.1, 182.6]
The range of this set is approximately 2.3
```

b. The sample variance S^2 from the definition. (*Hint: First Subtract 180 from each observation.*)

To do this let us first define sample variance.

$$S^2 = \frac{\sum(x_i - \overline{x})^2}{n-1}$$

In [24]:

```
# I will be tackling two different solutions to this problem to
# show their equivelance.

# Numpy variance function ---------------------------------------------------
print('The sample variance using numpy is', round(np.var(phb_set, ddof=1),2))
    # note: ddof controls degrees of freedom

# Manual computation --------------------------------------------------------
#subtract 180 from each observation and round to address computational error.
modified_phb_set = [round((value - 180),2) for value in phb_set]

#Preparation
# value_minus_mean: Subtracts the mean from each value and squares the results.
value_minus_mean = [((value - np.mean(modified_phb_set))**2)
                    for value in modified_phb_set]
numerator = sum(value_minus_mean)
den = len(modified_phb_set) - 1

# Calculation
sample_variance = round((numerator / den), 2)
print('The sample variance using the traditional method is', sample_variance)
```

```
The sample variance using numpy is 0.52
The sample variance using the traditional method is 0.52
```

As we can see here, I will be using a lot of numpy as it is an incredibly powerful tool.

c. The sample standard deviation.

First, as always, let us define standard deviation.

$$S = \sqrt{\frac{\sum(x_i - \overline{x})^2}{n-1}}$$

It is simply the square root of the variance. So, for the sake of it let us compare tools again.

In [25]:

```
print('The standard deviation using numpy is', round(np.std(phb_set, ddof=1),2))

phb_stand_dev = round(math.sqrt(sample_variance),2)
print('The standard deviation is', phb_stand_dev)
```

```
The standard deviation using numpy is 0.72
The standard deviation is 0.72
```

The article "Effects of Short-Term Warming on Low and High Latitude Forest Ant Communities" (*Ecoshpere*, May 2011, Article 62) described an experiment in which observations on various characteristics were made using minichambers of three different types: (1) cooler (PVC frames covered with shade cloth), (2) control (PVC frames only), and (3) warmer (PVC frames covered with plastic). One of the article's authors kindly supplied the accompanying data on the difference between air and soil temperatures (°C).

Cooler	Control	Warmer
1.59	1.92	2.57
1.43	2.00	2.60
1.88	2.19	1.93
1.26	1.12	1.58
1.91	1.78	2.30
1.86	1.84	0.84
1.90	2.45	2.65
1.57	2.03	0.12
1.79	1.52	2.74
1.72	0.53	2.53
2.41	1.90	2.13
2.34		2.86
0.83		2.31
1.34		1.91
1.76		

a. Compare measures of center for three different samples.
The three measures of center I'll be using are mean and median. I'll utilize numpy for each of these sets.

In [26]:
```
# Setup -----------------------------
cooler_df = pd.DataFrame({
    'Cooler':[1.59,1.43,1.88,1.26,1.91,1.86,1.90,1.57,1.79,1.72,2.41,2.34,0.83,
              1.34,1.76]})
control_df = pd.DataFrame({
    'Control':[1.92, 2.00, 2.19, 1.12, 1.78, 1.84, 2.45, 2.03, 1.52,
               0.53, 1.90]})
warmer_df = pd.DataFrame({
    'Warmer':[2.57, 2.60, 1.93, 1.58, 2.30, 0.84, 2.65, 0.12, 2.74, 2.53, 2.13,
              2.86, 2.31, 1.91]})

# Merges dataframes together and adds NaNs to empty indices
temp_dif_df = pd.concat([cooler_df, control_df, warmer_df], axis=1)
```

In [27]:
```
# Computations ------------------------
print('The mean of the cooler set is',
      round(np.mean(temp_dif_df['Cooler']),2), '\n'
     'The mean of the control set is',
      round(np.mean(temp_dif_df['Control']),2), '\n'
     'The mean of the warmer set is',
      round(np.mean(temp_dif_df['Warmer']),2), '\n')

print('The median of the cooler set is',
      round(np.median(temp_dif_df['Cooler']),2), '\n'
```

```
      'The median of the control set is',
       round(np.nanmedian(temp_dif_df['Control']),2), '\n'
      'The median of the warmer set is',
       round(np.nanmedian(temp_dif_df['Warmer']),2), '\n')
```

```
The mean of the cooler set is 1.71
The mean of the control set is 1.75
The mean of the warmer set is 2.08

The median of the cooler set is 1.76
The median of the control set is 1.9
The median of the warmer set is 2.3
```

What we can see here is that the ranking of the middle measurements are consistent between mean and median. The median is also consistently larger than the mean here, which is unsurprising when you see how small the lowest measurements are. That is absolutely skewing the mean down slightly.

b. Calculate, interpret, and compare the standard deviations for the three different samples.

In [28]:

```
print('The standard deviation of the cooler set is',
       round(np.std(temp_dif_df['Cooler']),2), '\n'
      'The standard deviation of the control set is',
       round(np.std(temp_dif_df['Control']),2), '\n'
      'The mean standard deviation the warmer set is',
       round(np.std(temp_dif_df['Warmer']),2), '\n')
```

```
The standard deviation of the cooler set is 0.39
The standard deviation of the control set is 0.51
The mean standard deviation the warmer set is 0.75
```

The standard deviation follows the same ranking at the measures of center. This is largely unsurprising. What is surprising is the degree though. The warmer sets standard deviation is twice that of the cooler set.

c. Do the fourth spreads for the three samples convey the same message as do the standard deviations about relative variability?
Note: Fourth spread is a fancy way of saying interquartile range. Let us compare the IQR of each set.

In [29]:

```
# Setup ------------------------------
cooler_q75, cooler_q25 =  np.percentile(temp_dif_df['Cooler'], [75 ,25])
control_q75, control_q25 =  np.nanpercentile(temp_dif_df['Control'], [75 ,25])
warmer_q75, warmer_q25 =  np.nanpercentile(temp_dif_df['Warmer'], [75 ,25])

# Results ------------------------------
print('The interquartile range of the cooler set is',
       round((cooler_q75 - cooler_q25) ,2), '\n'
      'The interquartile range of the control set is',
       round((control_q75 - control_q25),2), '\n'
      'The interquartile range of the warmer set is',
       round((warmer_q75 - warmer_q25),2), '\n')
```

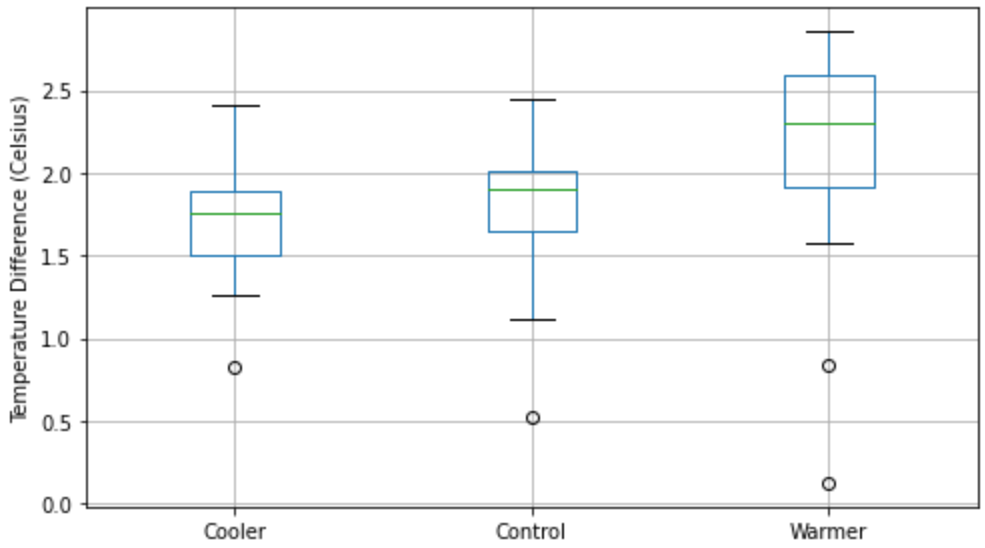
```
The interquartile range of the cooler set is 0.39
The interquartile range of the control set is 0.36
The interquartile range of the warmer set is 0.68
```

This method conveys a similar degree of spread for all but the control set. That one was quite different from the standard deviation calculation.

d. Construct a comparitive boxplot and comment on any interesting features.

In [30]:
```
temp_dif_df.boxplot(return_type=None)
plt.ylabel('Temperature Difference (Celsius)')
```

Out[30]:
```
Text(0, 0.5, 'Temperature Difference (Celsius)')
```



What we see here is that as the temperature gets warmer the data values start to deviate more and more. The warmer data set even has two points that would qualify as outliers which explains the far higher mean it had.

47.

Zinfandel is a popular red wine varietal produced almost exclusively in California. It is rather controversial among wine connoisseurs because its alcohol content varies quite substantially from one producer to another. In May 2013, the author went to the website klwines.com, randomly selected 10 zinfandels from among the 325 available, and obtained the following values of alcohol content (%):

`[14.8, 14.5, 16.1, 14.2, 15.9, 13.7, 16.2, 14.6, 13.8, 15.0]`

a. Calculate and interpret several measures of center.

In [31]:
```
wine_set = [14.8, 14.5, 16.1, 14.2, 15.9, 13.7, 16.2, 14.6, 13.8, 15.0]

print('The mean of the alcohol content is', np.mean(wine_set), '%')
print('The median of the alcohol content is', np.median(wine_set), '%')
```

```
The mean of the alcohol content is 14.88 %
The median of the alcohol content is 14.7 %
```

b. Calculate the sample variance using the defining formula.

In [32]:
```
print('The variance of the alcohol content set is', round(np.var(wine_set),2), '%')
```

```
The variance of the alcohol content set is 0.75 %
```

53.

A mutual fund is a professionally managed investment scheme that pools money from many investors and invests in a variety of securities. Growth funds focus primarily on increasing the value of investments, whereas blended funds seek a balance between current income and growth. Here is data on the expense ratio (expenses as a % of assets, from www.morningstar.com) for samples of 20 large-cap balanced funds and 20 large-cap growth funds ("largecap" refers to the sizes of companies in which the funds invest; the population sizes are 825 and 762, respectively):

BL: [1.03, 1.23, 1.10, 1.64, 1.30, 1.27, 1.25, 0.78, 1.05, 0.64, 0.94, 2.86, 1.05, 0.75, 0.09, 0.79, 1.61, 1.26, 0.93, 0.84]

GR: [0.52, 1.06, 1.26, 2.17, 1.55, 0.99, 1.10, 1.07, 1.81, 2.05, 0.91, 0.79, 1.39, 0.62, 1.52, 1.02, 1.10, 1.78, 1.01, 1.15]

a. Calculate and compare the values of $\overline{x}$, $\tilde{x}$, and s for the two types of funds.

In [33]:
```
# Setup --------------------------------
bl_df = pd.DataFrame({
    'Bl':[1.03,1.23,1.10,1.64,1.30,1.27,1.25,0.78,1.05,0.64,0.94,2.86,
          1.05,0.75,0.09,0.79,1.61,1.26,0.93,0.84]})
gr_df = pd.DataFrame({
    'Gr':[ 0.52,1.06,1.26,2.17,1.55,0.99,1.10,1.07,1.81,2.05,0.91,0.79,
          1.39,0.62,1.52,1.02,1.10,1.78,1.01,1.15]})

expense_df = pd.concat([bl_df, gr_df], axis=1)
```

In [34]:
```
# Calculations -----------------------------
# mean
print('The mean of the Bl set is',
      round(np.mean(expense_df['Bl']),2), '\n'
     'The mean of the Gr set is',
      round(np.mean(expense_df['Gr']),2), '\n')

# median
print('The median of the Bl set is',
      round(np.median(expense_df['Bl']),2), '\n'
     'The median of the Gr set is',
      round(np.nanmedian(expense_df['Gr']),2), '\n')

# standard deviation
print('The standard deviation of the Bl set is',
      round(np.std(expense_df['Bl']),2), '\n'
     'The standard deviation of the Gr set is',
      round(np.std(expense_df['Gr']),2), '\n')
```

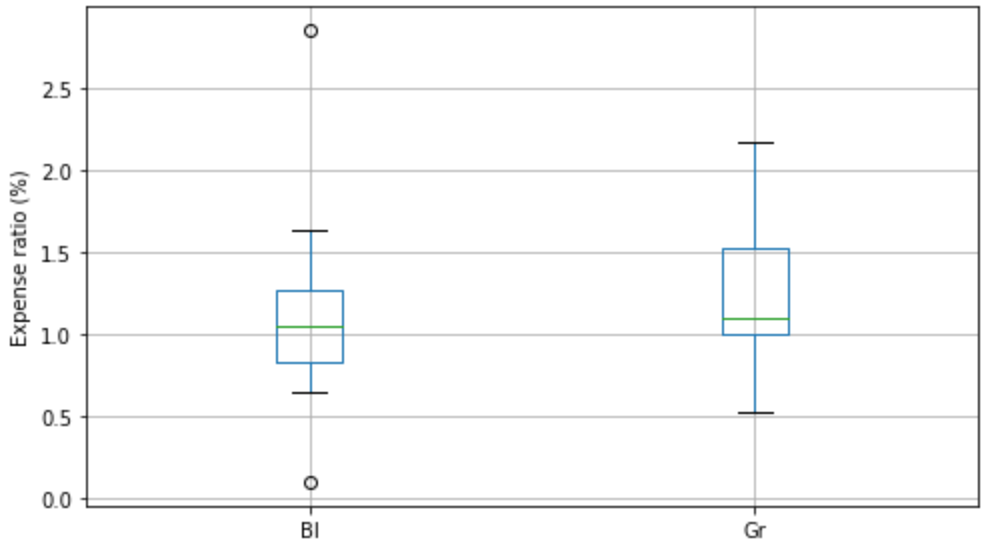
```
The mean of the Bl set is 1.12
The mean of the Gr set is 1.24

The median of the Bl set is 1.05
The median of the Gr set is 1.1

The standard deviation of the Bl set is 0.52
The standard deviation of the Gr set is 0.44
```

Judging by these calculations, it appears that the 'Bl' set averages very slightly lower in its values but has a larger degree of spread than the 'Gr' set.

b. Construct a comparative boxplot the two types of funds, and comment on interesting features.

In [35]:
```
expense_df.boxplot(return_type=None)
plt.ylabel('Expense ratio (%)')
```

Out[35]:
```
Text(0, 0.5, 'Expense ratio (%)')
```



Juding by the graph it seems that 'Gr' actually has more spread than 'Bl', I would guess that it was the outliers that were inflating the value.

54.

Grip is applied to produce normal surface forces that compress the object being gripped. Examples include two people shaking hands, or a nurse squeezing a patient's forearm to stop bleeding. The article "Investigation of Grip Force, Normal Force, Contact Area, Hand Size, and Handle Size for Cylindrical Handles" (*Human Factors*, 2008: 734–744) included the following data on grip strength (N) for a sample of 42 individuals:

[16, 18, 18, 26, 33, 41, 54, 56, 66, 68, 87, 91, 95, 98, 106, 109, 111, 118, 127, 127, 135, 145, 147, 149, 151, 168, 172, 183, 189, 190, 200, 210, 220, 229, 230, 233, 238, 244, 259, 294, 329, 403]

a. Construct a stem-and-leaf display based on repeating each stem value twice, and comment on interesting features.

In [36]:
```
grip_vec = [16, 18, 18, 26, 33, 41, 54, 56, 66, 68, 87, 91, 95, 98, 106,
            109, 111, 118, 127, 127, 135, 145, 147, 149, 151, 168, 172,
            183, 189, 190, 200, 210, 220, 229, 230, 233, 238, 244, 259,
            294, 329, 403]
stemgraphic.stem_graphic(grip_vec)
```

Out[36]:
```
(<Figure size 540x216 with 1 Axes>,
 <matplotlib.axes._axes.Axes at 0x7f48a02d1bb0>)
```

```
   403
42 |  4 0
41 |
41 |  3 3
40 |    69
38 |  2 01233344
30 |    55577899
22 |  1 0011123344
12 |    567799
 6 |  0 222334
   16
```

```
Key: aggr|stem|leaf
4 | 0 = 4.0x100 = 400.0
```

What we see here is that there an immediate drop-off after around 250. The data goes from being very concentrated to having only a scant few data points.

b. Determine the values of the fourths and the fourthspread.

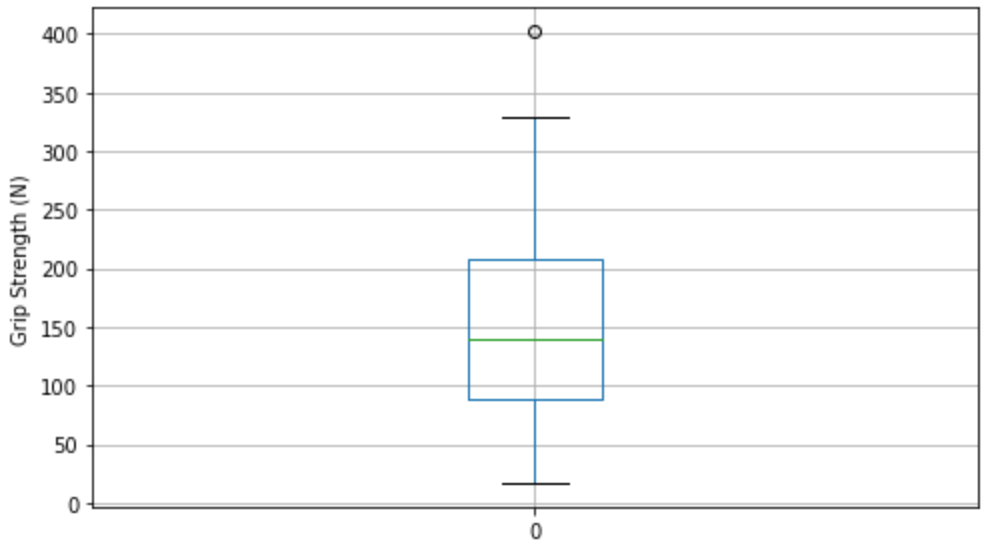
In [37]:
```
grip_75, grip_25 = np.percentile(grip_vec, [75, 25])
print('The upper fourth is', grip_75, 'and the lower fourth is', grip_25)
print('The fourthspread is', grip_75 - grip_25)
```

```
The upper fourth is 207.5 and the lower fourth is 88.0
The fourthspread is 119.5
```

c. Construct a boxplot and comment on its features.

In [38]:
```
grip_df = pd.DataFrame(data=grip_vec)
grip_df.boxplot(return_type=None)
plt.ylabel('Grip Strength (N)')
```

Out[38]:
```
Text(0, 0.5, 'Grip Strength (N)')
```



It seems that matplotlib agrees with my assessment on 403 being an outlier. It doesn't consider 329 one which I find interesting though. But as I mentioned, the boxplot actually looks fairly tight up until you reach the upper quadrant. Those upper few values stretched things out a lot.

59.

lood cocaine concentration (mg/L) was determined both for a sample of individuals who had died from cocaine-induced excited delirium (ED) and for a sample of those who had died from a cocaine overdose

without excited delirium; survival time for people in both groups was at most 6 hours. The accompanying data was read from a comparative boxplot in the article “Fatal Excited Delirium Following Cocaine Use” (*J. of Forensic Sciences*, 1997: 25–31).

ED: [0, 0, 0, 0, .1, .1, .1, .1, .2, .2, .3, .3, .3, .4, .5, .7, .8, 1.0, 1.5, 2.7, 2.8, 3.5, 4.0, 8.9, 9.2, 11.7, 2.7, 2.8]

Non-ED: [0, 0, 0, 0, 0, .1, .1, .1, .1, .2, .2, .2, .3, .3, .3, .4, .5, .5, .6, .8, .9, 1.0, 1.2, 1.4, 1.5, 1.7, 2.0, 3.2, 3.5, 4.1, 4.3, 4.8, 5.0, 5.6, 5.9, 6.0, 6.4, 7.9, 8.3, 8.7, 9.1, 9.6, 9.9, 11.0, 11.5, 12.2, 12.7, 14.0, 16.6, 17.8]

a. Determine the medians, fourths, and fourth spreads for the two samples.

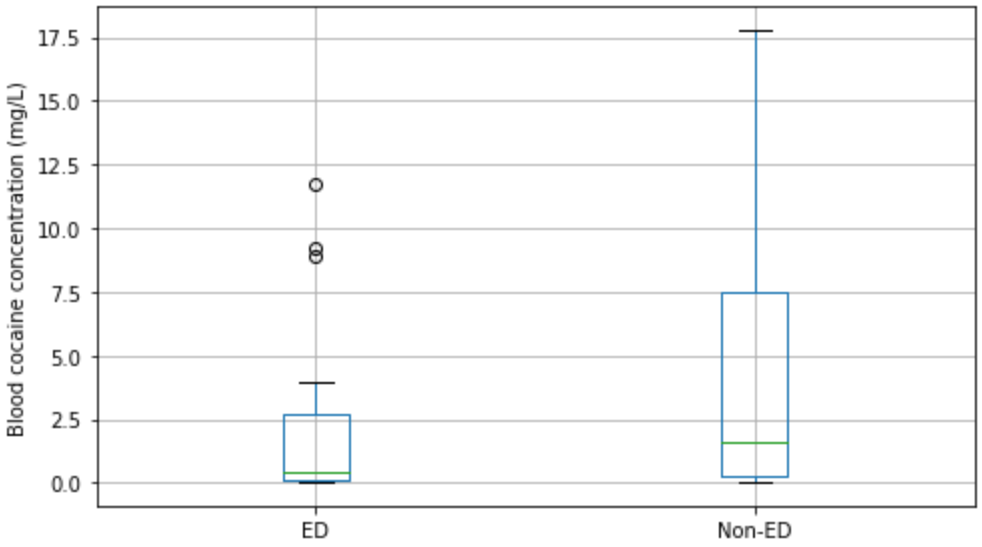
In [39]:
```
# Setup ------------------------
ed_df = pd.DataFrame({
    'ED': [0, 0, 0, 0, .1, .1, .1, .1, .2, .2, .3, .3, .3, .4, .5, .7, .8, 1.0,
           1.5, 2.7, 2.8, 3.5, 4.0, 8.9, 9.2, 11.7, 2.7, 2.8]
})
non_ed = pd.DataFrame({
    'Non-ED': [0, 0, 0, 0, 0, .1, .1, .1, .1, .2, .2, .2, .3, .3, .3, .4,
               .5, .5, .6, .8, .9, 1.0, 1.2, 1.4, 1.5, 1.7, 2.0, 3.2, 3.5,
               4.1, 4.3, 4.8, 5.0, 5.6, 5.9, 6.0, 6.4, 7.9, 8.3, 8.7, 9.1,
               9.6, 9.9, 11.0, 11.5, 12.2, 12.7, 14.0, 16.6, 17.8]
})

delirium_df = pd.concat([ed_df, non_ed], axis=1)

# Results ------------------------
ed_25, ed_50, ed_75 = np.nanpercentile(delirium_df['ED'], [25, 50, 75])
non_25, non_50, non_75 = np.nanpercentile(delirium_df['Non-ED'], [25, 50, 75])

print('The 25th percentile for the excited delirium sample is', round(ed_25, 2),
      '\nThe median for that sample is', round(ed_50, 2),
     '\nThe 75th percentile for that sample is', round(ed_75, 2),
     '\nThe fourth spread for that sample is', round((ed_75 - ed_25),2))

print('\nThe 25th percentile for the sample without excited delirium is',
      round(non_25, 2),
      '\nThe median for that sample is', round(non_50, 2),
     '\nThe 75th percentile for that sample is', round(non_75, 2),
     '\nThe fourth spread for that sample is', round((non_75 - non_25),2))
```

```
The 25th percentile for the excited delirium sample is 0.1
The median for that sample is 0.45
The 75th percentile for that sample is 2.72
The fourth spread for that sample is 2.62

The 25th percentile for the sample without excited delirium is 0.3
The median for that sample is 1.6
The 75th percentile for that sample is 7.52
The fourth spread for that sample is 7.22
```

Construct a comparative boxplot, and use it as a basis for comparing and contrasting the ED and non-ED samples.

In [40]:
```
delirium_df.boxplot()
plt.ylabel('Blood cocaine concentration (mg/L)')
```

Out[40]:
```
Text(0, 0.5, 'Blood cocaine concentration (mg/L)')
```



What we see is that there was a far higher cocaine concentration in the group that died without excited delirium alongside their overdose. There was also a far higher variance in cocaine concentration in the non-excited delirium group. The concentration does get quite high on the excited delirium group, but it's just a small group of outliers. Those outliers may have been a group that developed a sort of tolerance for the drug.

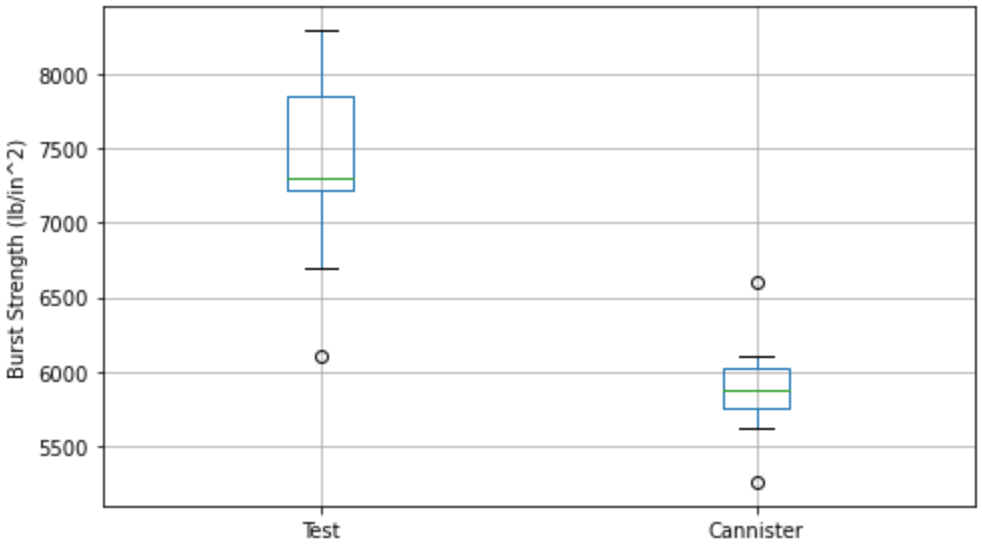
60.

Observations on burst strength $\frac{lb}{in^2}$ were obtained both for test nozzle closure welds and for production cannister nozzle welds ("Proper Procedures Are the Key to Welding Radioactive Waste Containers," *Welding J.* Aug. 1997: 61-67)

Test: `[7200, 6100, 7300, 8000, 7400, 7300, 7300, 8000, 6700, 8300]`
Cannister: `[5250, 5625, 5900, 5700, 6050, 5800, 6000, 5875, 6100, 5850, 6600]`

In [41]:
```
test_df = pd.DataFrame({
    'Test': [7200, 6100, 7300, 8000, 7400, 7300, 7300, 8000, 6700, 8300]
})
can_df = pd.DataFrame({
    'Cannister': [5250, 5625, 5900, 5700, 6050, 5800, 6000, 5875, 6100,
                  5850, 6600]
})

burst_df = pd.concat([test_df, can_df], axis=1)

burst_df.boxplot()
plt.ylabel('Burst Strength (lb/in^2)')
```

Out[41]:
```
Text(0, 0.5, 'Burst Strength (lb/in^2)')
```



What we can see is that the test values were across the board higher than the actual canister values. The cannister values exhibited far more consistency despire their lower values.