In [ ]:

```
```

4 HISTOGRAM GRAFİĞİ

In [ ]:

```
```

Tanım:

Histogram nümerik bir değişkenin dağılımını göstermek çin kullanılan bir grafik türüdür. Histogram veri seti sadece nümerik değerler alır.
Histogram grafikleri daha çok bir aralıktaki verileri gösterir. Yaş aralığı ve populasyondaki yoğunluğu gibi...

In [5]:

```
import numpy as np
# `numpy.random` uses its own PRNG.
np.random.seed(444)
np.set_printoptions(precision=3)

d = np.random.laplace(loc=15, scale=3, size=500)
d[:5]
```

Out[5]:

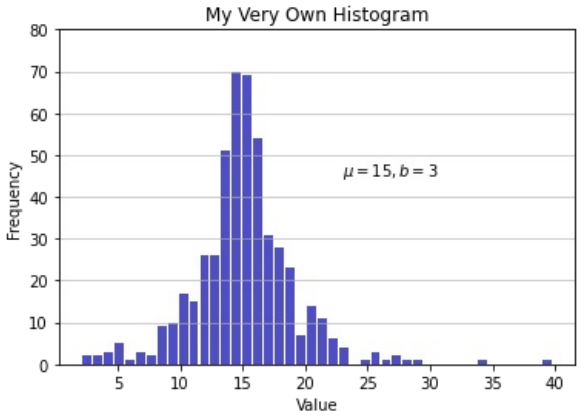
```
array([18.406, 18.087, 16.004, 16.221,  7.358])
```

In [22]:

```
import matplotlib.pyplot as plt

# An "interface" to matplotlib.axes.Axes.hist() method
n, bins, patches = plt.hist(x=d, bins='auto', color='#0504aa',
                            alpha=0.7, rwidth=0.85)
"""

# To see too small values use log scale

n, bins, patches = plt.hist(x=d, bins='auto', color='#0504aa',
                            alpha=0.7, rwidth=0.85,log=True)

"""
#bins for how many bins we'll use, and range of our graph
#median =15
#plt.axvline(median,color='red', label='mu=15')
#plt.legend()

plt.grid(axis='y', alpha=0.75)
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('My Very Own Histogram')
plt.text(23, 45, r'$\mu=15, b=3$')
maxfreq = n.max()
# Set a clean upper y-axis limit.
plt.ylim(ymax=np.ceil(maxfreq / 10) * 10 if maxfreq % 10 else maxfreq + 10)
```

Out[22]:

```
(0.0, 80.0)
```



Ne için Kullanılır:

Histogramlar bir aralıktaki verilerin gösterimi için kullanılır.

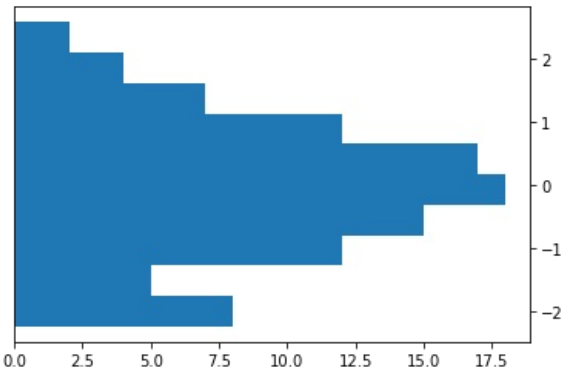
Ayrıca veri setindeki hatalı verilerin yakalanmasında oldukça etkilidir.

Çeşitleri:

En çok kullanılan çeşidi mirror ayna histogramıdır ki iki değerin karşılaştırılmasında kullanılır.

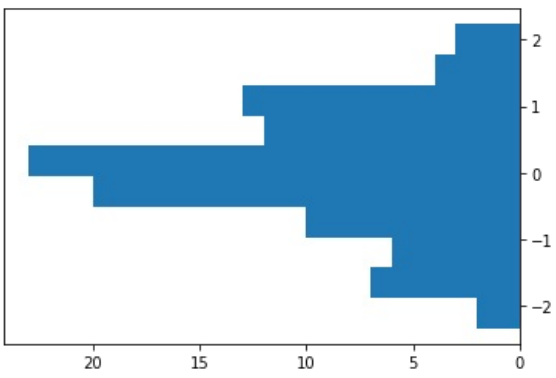
In [9]:

```
import numpy as np
import matplotlib.pyplot as plt

#generate some data
data = np.random.normal(size=100)



#define the plot
fig, ax = plt.subplots()

#plot the data as a histogram
ax.hist(data, orientation=u'horizontal')
#ax.hist(data, orientation=u'horizontal',alpha=0.8,color='r')

#move ticks to the right
ax.yaxis.tick_right()

plt.show()
```



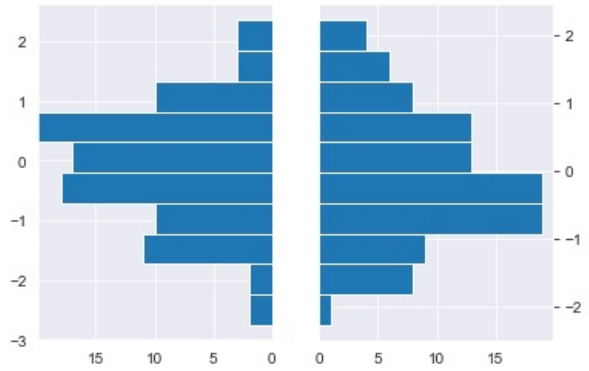
In [3]:

```
import numpy as np
import matplotlib.pyplot as plt

#generate some data
data = np.random.normal(size=100)

#define the plot
fig, ax = plt.subplots()

#plot the data as a histogram
ax.hist(data, orientation=u'horizontal')

#invert the order of x-axis values
ax.set_xlim(ax.get_xlim()[::-1])

#move ticks to the right
ax.yaxis.tick_right()

plt.show()
```



In [21]:

```
import numpy as np
import matplotlib.pyplot as plt

#generate some data
data = np.random.normal(size=100)
data2 = np.random.normal(size=100)

#print(min(data))
#print(max(data))
#print(min(data2))
#print(max(data2))

#define the plot
fig, (ax2,ax1) = plt.subplots(1,2)

#plot the data as a histogram
ax1.hist(data, orientation=u'horizontal')


#invert the order of x-axis values
ax2.set_xlim(ax1.get_xlim()[::-1])
ax2.hist(data2, orientation=u'horizontal')

#move ticks to the right
ax1.yaxis.tick_right()

plt.show()
```



Yukarıdaki grafikte kullanıldığı gibi dikey veya yatay ayna özelliği ile iki veri kaşılaştırılabilir. Tabi y ekseninin eşit olmasına dikkat ederek :)

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Kullanım Hataları:

Histogram garfiklerini 3 veya daha fazla verinin kıyaslanması için kullanmayınız iki veri için de mümkün ise farklı renkte ve alpha değeri ekleyerek kullanın.

Çok fazla sayıda kutu boyutu kullanmayın. ilginç renkler kullanmaya çalışmayın size daha uygun bir bakış açısı verecektir. Bar plot ile karıştırmayın bunda sadece nümerkik değerler vardır.

3 veya daha fazla grubu birlikte kullanmayın. fazla sayıda verinin dağılımı incelemek için violin daha kullanışlıdır.

Eşit olmayan genişlikte çubuklar kullanmayın.

In [ ]:

In [11]:

```
import numpy as np
# `numpy.random` uses its own PRNG.
np.random.seed(444)
np.set_printoptions(precision=3)

d1 = np.random.laplace(loc=15, scale=3, size=500)
d2 = np.random.laplace(loc=15, scale=3, size=500)
d1[:5]
d2[:5]
```

Out[11]:

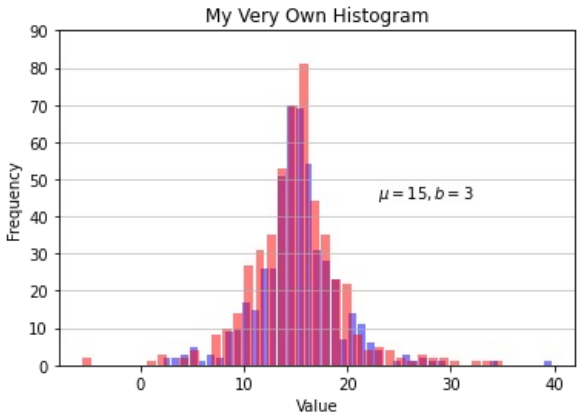
```
array([13.813, 16.601, 13.666, 16.312,  5.15 ])
```

In [12]:

```
import matplotlib.pyplot as plt

# An "interface" to matplotlib.axes.Axes.hist() method
n, bins, patches = plt.hist(x=d1, bins='auto', color='b',
                            alpha=0.5, rwidth=0.85)
n, bins, patches = plt.hist(x=d2, bins='auto', color='r',
                            alpha=0.5, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('My Very Own Histogram')
plt.text(23, 45, r'$\mu=15, b=3$')
maxfreq = n.max()
# Set a clean upper y-axis limit.
plt.ylim(ymax=np.ceil(maxfreq / 10) * 10 if maxfreq % 10 else maxfreq + 10)
```

Out[12]:

```
(0.0, 90.0)
```



Tek grafikte üst üste yazılmada okuma zorluğunu engellemek için alpha değeri verip yarı saydam bir grafik çizmek daha uygun olacaktır.

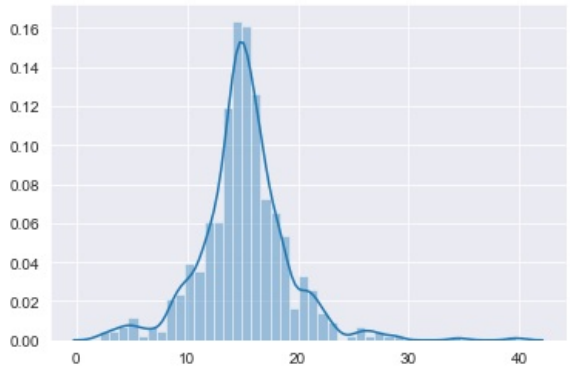
In [ ]:

In [19]:

```
import seaborn as sns

sns.set_style('darkgrid')
sns.distplot(d1)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f45f423a90>
```



Yukarıdaki gibi bir histogram çizilerek verinin okunması daha kolay hale getiilebilir.

In [ ]:

```

```