In [ ]:

```
```

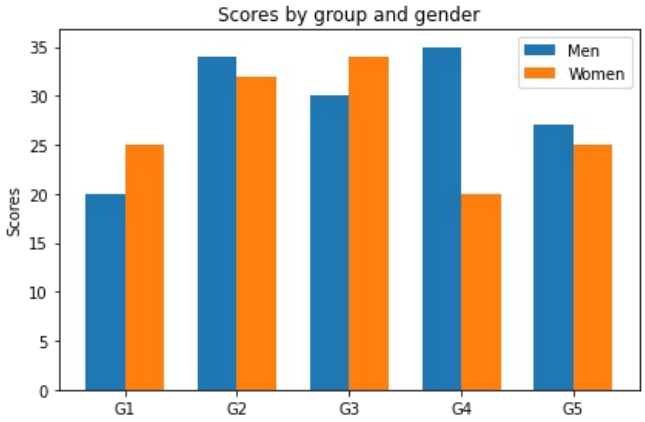
4- Bar Chart- Çubuk Grafiği

Tanım:

Bar plot en çok kullanılan grafik türlerinden biridir.Sayısal ve kategorik iki değişken arasındaki ilişkiyi gösterir. Her bir kategorinin değeri numerik olarak çubuk olarak gösterilir. Çubuğun boyu kategorinin numerik değeri olarak gösterilir.

In [3]:

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


labels = ['G1', 'G2', 'G3', 'G4', 'G5']
men_means = [20, 34, 30, 35, 27]
women_means = [25, 32, 34, 20, 25]

x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, men_means, width, label='Men')
rects2 = ax.bar(x + width/2, women_means, width, label='Women')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

#ax.bar_label(rects1, padding=3)
#ax.bar_label(rects2, padding=3)

fig.tight_layout()

plt.show()
```



Ne için Kullanılır:

Çubuk grafiği kategorik ve numerik iki değişkenin arasındaki ilişkiyi gösterir.

Gruplanmış çubuk grafiğinde birden fazla kategorik değişkenin numerik değerlerinin bir birileri ile ilişkisi gösterilir.

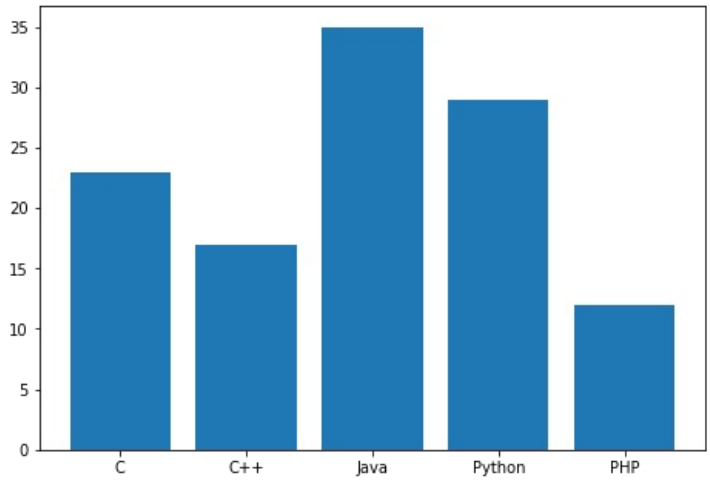
Gruplanmış çubuk grafiklerinde her bir değişken yanyana gösterilmek yerine üst üste yığılmış olarak da gösterilebilir.

In [ ]:

```
```

In [1]:

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23,17,35,29,12]
ax.bar(langs,students)
plt.show()
```



In [5]:

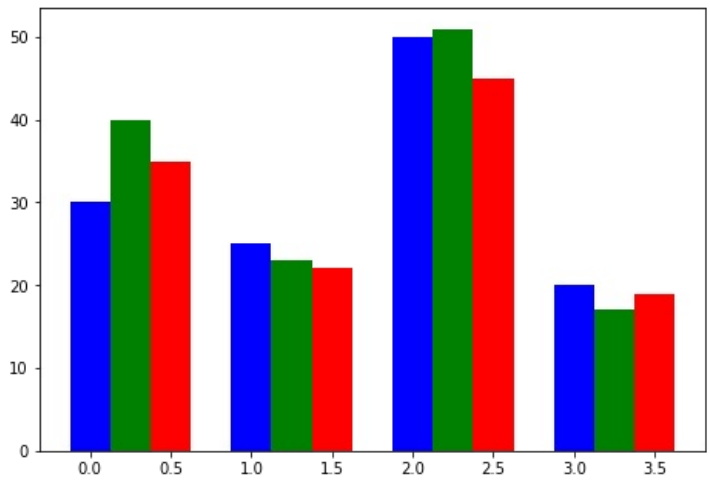
```
# Yukarıdaki grafik birden fazla kategorize edilmiş veriyi ve aralarındaki ilişkiyi göstermektedir.
```

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
data = [[30, 25, 50, 20],
[40, 23, 51, 17],
[35, 22, 45, 19]]
X = np.arange(4)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(X + 0.00, data[0], color = 'b', width = 0.25)
ax.bar(X + 0.25, data[1], color = 'g', width = 0.25)
ax.bar(X + 0.50, data[2], color = 'r', width = 0.25)
```

Out[2]:

```
<BarContainer object of 4 artists>
```



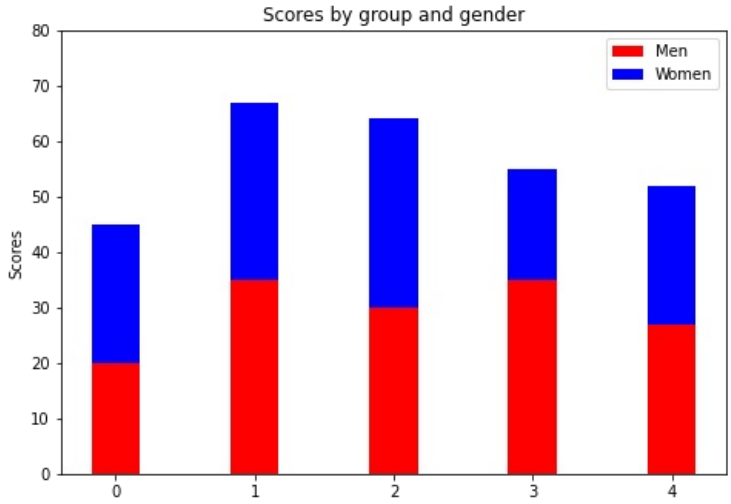
In [6]:

```
# Yukarıdaki grafik birden fazla katogorinin birden fazla değerinin birbirileri ile  karşılaştırılması içindir.
```

In [4]:

```
import numpy as np
import matplotlib.pyplot as plt
N = 5
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)
ind = np.arange(N) # the x locations for the groups
width = 0.35
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(ind, menMeans, width, color='r')
ax.bar(ind, womenMeans, width,bottom=menMeans, color='b')
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
ax.set_yticks(np.arange(0, 81, 10))
ax.legend(labels=['Men', 'Women'])
plt.show()
```

```
<ipython-input-4-efc6e9664b01>:14: MatplotlibDeprecationWarning: Passing the minor parameter of set_
xticks() positionally is deprecated since Matplotlib 3.2; the parameter will become keyword-only two
minor releases later.
  ax.set_xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
```



Yukarıdaki grafik birden fazla verinin yığım biçimde gösterimidir. Birden fazla kategorinin kümülatif değişimini göstermek için oldukça uygundur. Örneğin yenilenebilir enerji kaynaklarının toplamının yıllara göre değişimi gibi, ancak her bir alt verinin değişimini göstermek için ooldukça kötü bir yöntemdir.

In [ ]:

In [ ]:

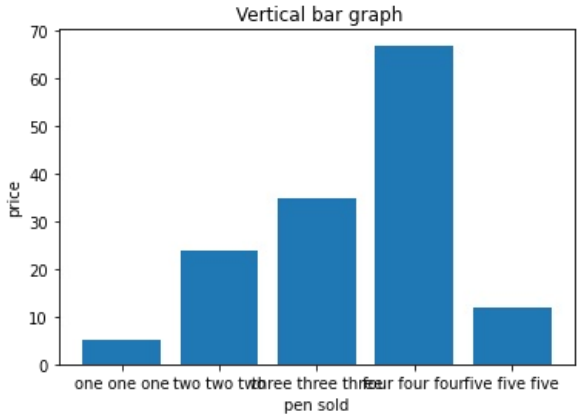
Çeşitleri:

Yatay ve Dikey çeşitleri mecuttur.

In [ ]:

In [3]:

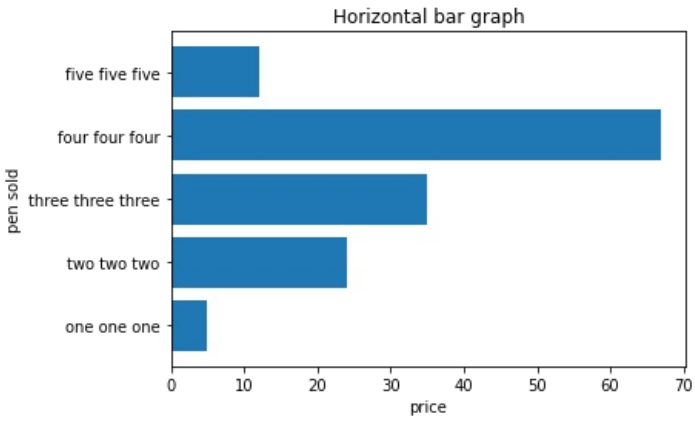
```
import matplotlib.pyplot as plt

x=['one one one', 'two two two', 'three three three', 'four four four', 'five five five']

# giving the values against
# each value at x axis
y=[5, 24, 35, 67, 12]
plt.bar(x, y)

# setting x-label as pen sold
plt.xlabel("pen sold")

# setting y_label as price
plt.ylabel("price")
plt.title(" Vertical bar graph")
plt.show()
```



Yukrıdaki grafikte görüldüğü gibi uzun isimlendirmeler yatay çubuk grafikleri için daha uygundur

In [5]:

```
import matplotlib.pyplot as plt
y=['one one one', 'two two two', 'three three three', 'four four four', 'five five five']

# getting values against each value of y
x=[5,24,35,67,12]
plt.barh(y, x)

# setting label of y-axis
plt.ylabel("pen sold")

# setting label of x-axis
plt.xlabel("price")
plt.title("Horizontal bar graph")
plt.show()
```



Yukrıdaki grafikte görüldüğü gibi uzun isimlendirmeler yatay çubuk grafikleri için daha uygundur

In [ ]:

Bar grafiğinin yüksek verimliliğine rağmen sıkıcı bulabilirsiniz ancak buna uygun değişik türleri mevcuttur.Lollipop plot -Stem Plot - bu durumda çubukların yerine çizgi konulmuş ve numerik değeri yuvarlak marker ile belirtilmiş ve vurgulanmış versiyonudur.

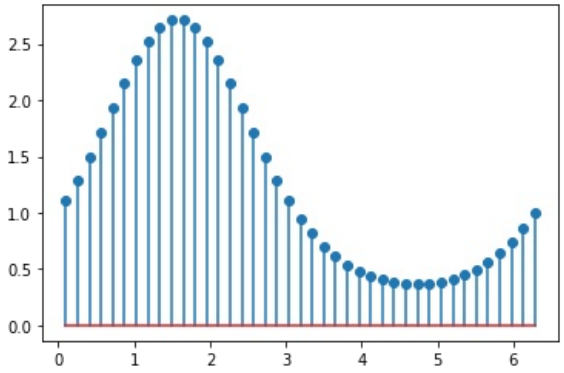
In [ ]:

```
```

In [6]:

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

x = np.linspace(0.1, 2 * np.pi, 41)
y = np.exp(np.sin(x))

plt.stem(x, y)
plt.show()
```

```
<ipython-input-6-8de8152579d3>:7: UserWarning: In Matplotlib 3.3 individual lines on a stem plot wil
l be added as a LineCollection instead of individual lines. This significantly improves the performa
nce of a stem plot. To remove this warning and switch to the new behaviour, set the "use_line_collec
tion" keyword argument to True.
  plt.stem(x, y)
```

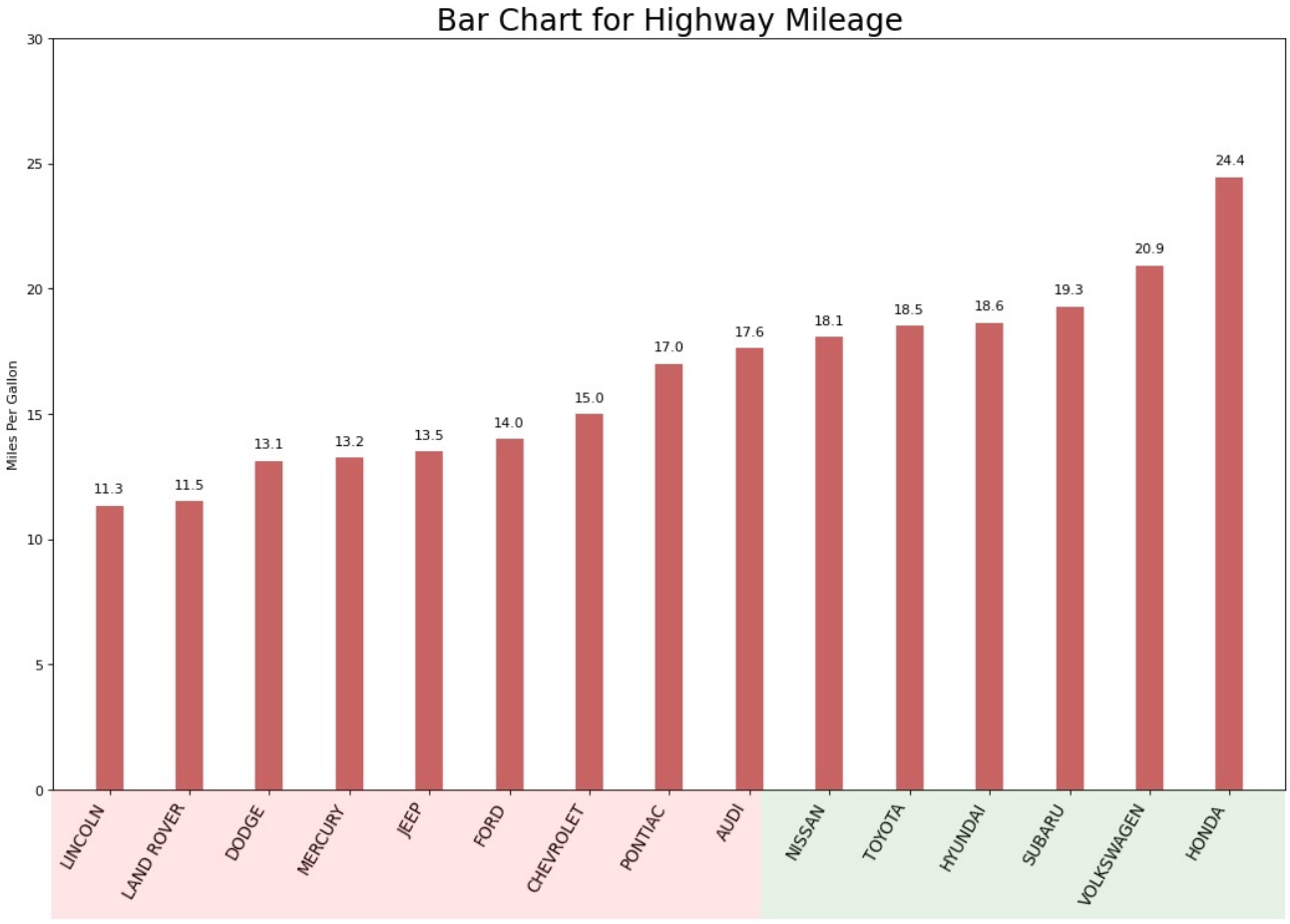


In [ ]:

```
```

In [7]:

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

# Prepare Data
df_raw = pd.read_csv("mpg_ggplot2.csv")
#df_raw = pd.read_csv("https://github.com/selva86/datasets/raw/master/mpg_ggplot2.csv")
df = df_raw[['cty', 'manufacturer']].groupby('manufacturer').apply(lambda x: x.mean())
df.sort_values('cty', inplace=True)
df.reset_index(inplace=True)

# Draw plot
import matplotlib.patches as patches

fig, ax = plt.subplots(figsize=(16,10), facecolor='white', dpi= 80)
ax.vlines(x=df.index, ymin=0, ymax=df.cty, color='firebrick', alpha=0.7, linewidth=20)

# Annotate Text
for i, cty in enumerate(df.cty):
    ax.text(i, cty+0.5, round(cty, 1), horizontalalignment='center')


# Title, Label, Ticks and Ylim
ax.set_title('Bar Chart for Highway Mileage', fontdict={'size':22})
ax.set(ylabel='Miles Per Gallon', ylim=(0, 30))
plt.xticks(df.index, df.manufacturer.str.upper(), rotation=60, horizontalalignment='right', fontsize=12)

# Add patches to color the X axis labels
p1 = patches.Rectangle((.57, -0.005), width=.33, height=.13, alpha=.1, facecolor='green', transform=fig.transFigu
re)
p2 = patches.Rectangle((.124, -0.005), width=.446, height=.13, alpha=.1, facecolor='red', transform=fig.transFigu
re)
fig.add_artist(p1)
fig.add_artist(p2)
plt.show()
```

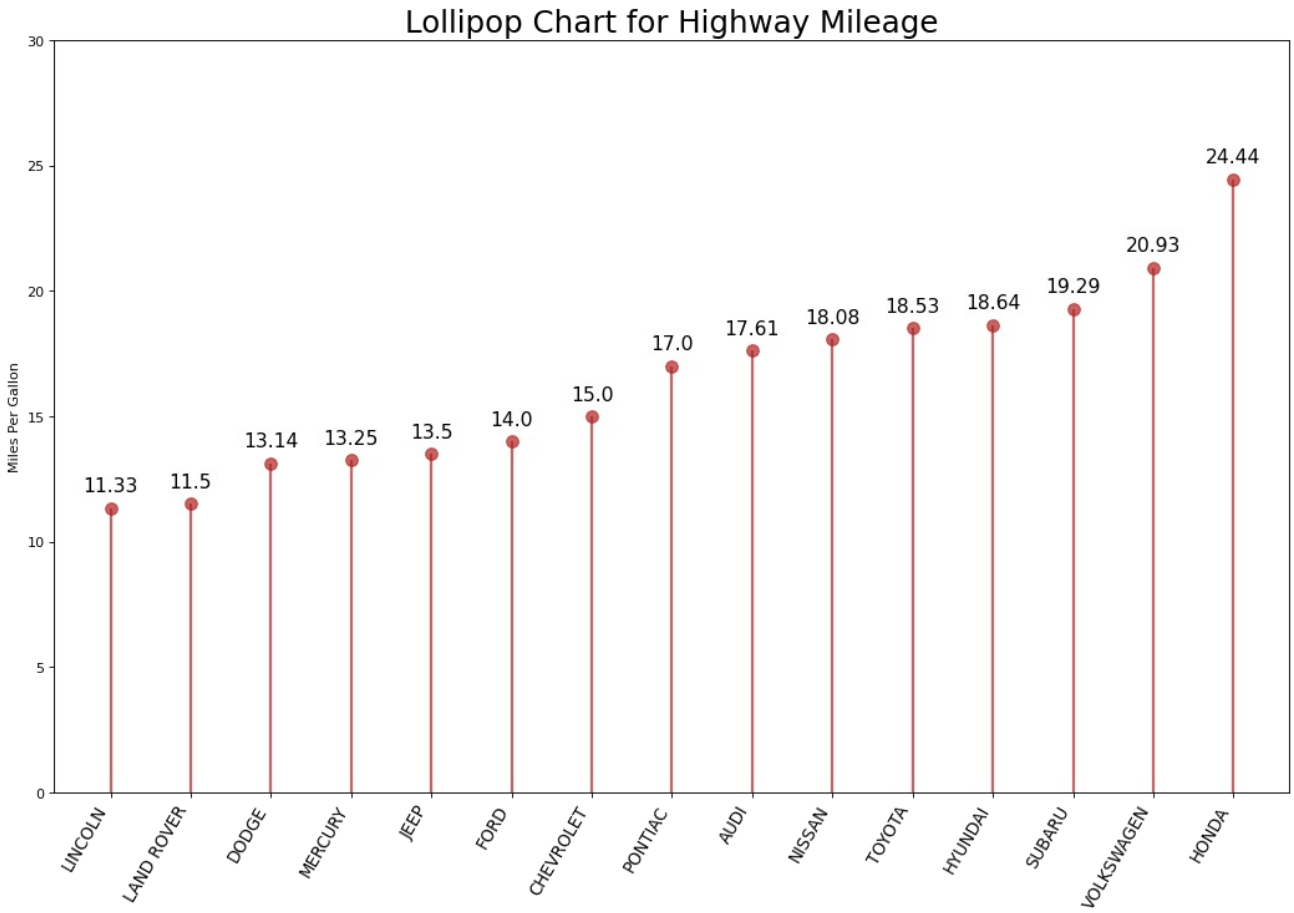


In [ ]:

```
```

In [8]:

```
# Prepare Data
#df_raw = pd.read_csv("https://github.com/selva86/datasets/raw/master/mpg_ggplot2.csv")
df = df_raw[['cty', 'manufacturer']].groupby('manufacturer').apply(lambda x: x.mean())
df.sort_values('cty', inplace=True)
df.reset_index(inplace=True)

# Draw plot
fig, ax = plt.subplots(figsize=(16,10), dpi= 80)
ax.vlines(x=df.index, ymin=0, ymax=df.cty, color='firebrick', alpha=0.7, linewidth=2)
ax.scatter(x=df.index, y=df.cty, s=75, color='firebrick', alpha=0.7)

# Title, Label, Ticks and Ylim
ax.set_title('Lollipop Chart for Highway Mileage', fontdict={'size':22})
ax.set_ylabel('Miles Per Gallon')
ax.set_xticks(df.index)
ax.set_xticklabels(df.manufacturer.str.upper(), rotation=60, fontdict={'horizontalalignment': 'right', 'size':12}
)
ax.set_ylim(0, 30)

# Annotate
for row in df.itertuples():
    ax.text(row.Index, row.cty+.5, s=round(row.cty, 2), horizontalalignment= 'center', verticalalignment='bottom'
, fontsize=14)

plt.show()
```



Yukarıdaki iki grafikte görüldüğü gibi ilk grafikte çubukların boyları ön planda ike ikinci grafikte değerler daha ön plandadır.

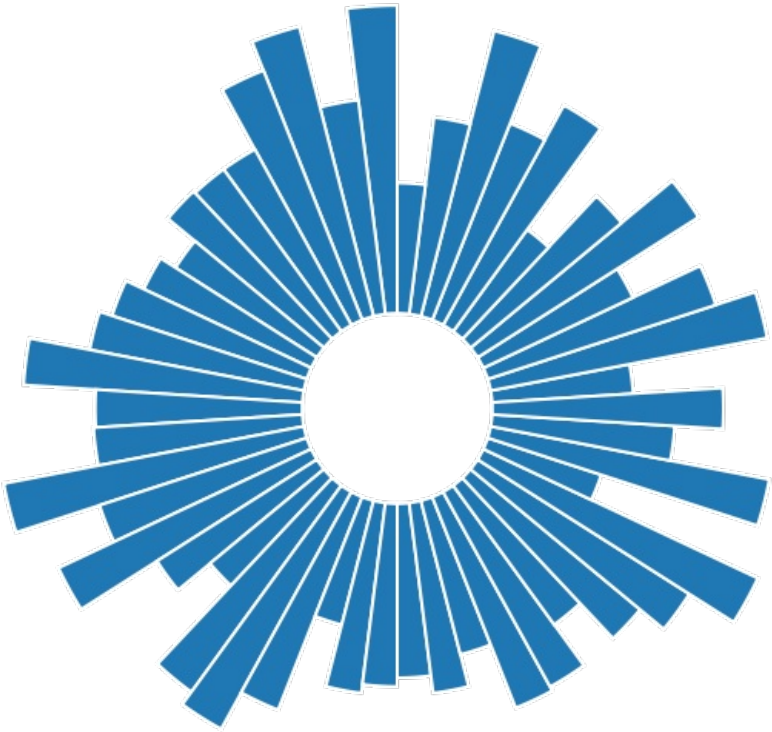
In [ ]:

```

```

Dairesel çubuk (circular bar plot) grafiği ise bar grafiğinin dairesel veriyonudur. Burada verinin değerlerinin gösterim hassasiyeti oldukça düşüktür bu nedenle çok bariz durumlarda kullanılabilir ama deneyip görmekte fayda var.

In [14]:

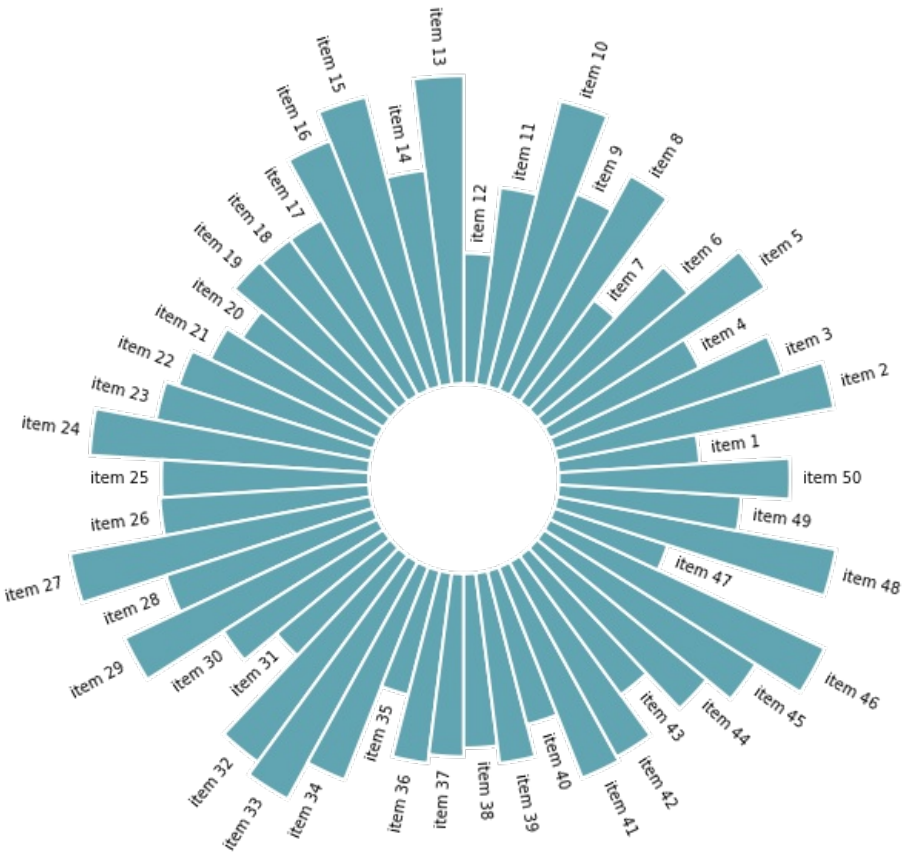
```
# import pandas for data wrangling
import pandas as pd

# Build a dataset
df = pd.DataFrame(
        {
            'Name': ['item ' + str(i) for i in list(range(1, 51)) ],
            'Value': np.random.randint(low=10, high=100, size=50)
        })

# Show 3 first rows
df.head(3)

# set figure size
plt.figure(figsize=(20,10))

# plot polar axis
ax = plt.subplot(111, polar=True)

# remove grid
plt.axis('off')

# Set the coordinates limits
upperLimit = 100
lowerLimit = 30

# Compute max and min in the dataset
max = df['Value'].max()

# Let's compute heights: they are a conversion of each item value in those new coordinates
# In our example, 0 in the dataset will be converted to the lowerLimit (10)
# The maximum will be converted to the upperLimit (100)
slope = (max - lowerLimit) / max
heights = slope * df.Value + lowerLimit

# Compute the width of each bar. In total we have 2*Pi = 360°
width = 2*np.pi / len(df.index)

# Compute the angle each bar is centered on:
indexes = list(range(1, len(df.index)+1))
angles = [element * width for element in indexes]
angles

# Draw bars
bars = ax.bar(
    x=angles,
    height=heights,
    width=width,
    bottom=lowerLimit,
    linewidth=2,
    edgecolor="white")
```



In [ ]:

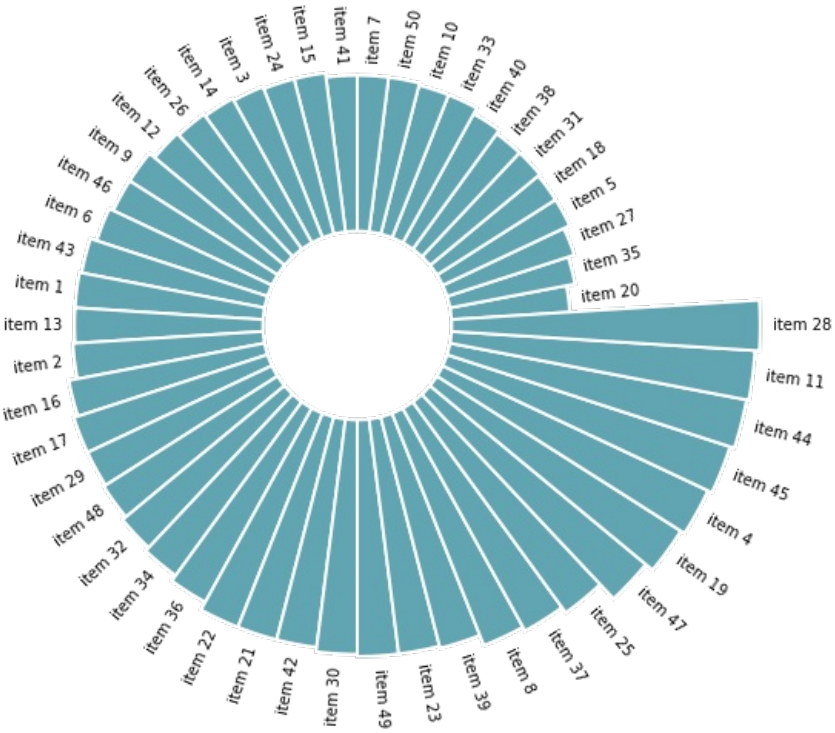
In [15]:

```
# initialize the figure
plt.figure(figsize=(20,10))
ax = plt.subplot(111, polar=True)
plt.axis('off')

# Draw bars
bars = ax.bar(
    x=angles,
    height=heights,
    width=width,
    bottom=lowerLimit,
    linewidth=2,
    edgecolor="white",
    color="#61a4b2",
)

# little space between the bar and the label
labelPadding = 4

# Add labels
for bar, angle, height, label in zip(bars,angles, heights, df["Name"]):

    # Labels are rotated. Rotation must be specified in degrees :(
    rotation = np.rad2deg(angle)

    # Flip some labels upside down
    alignment = ""
    if angle >= np.pi/2 and angle < 3*np.pi/2:
        alignment = "right"
        rotation = rotation + 180
    else:
        alignment = "left"

    # Finally add the labels
    ax.text(
        x=angle,
        y=lowerLimit + bar.get_height() + labelPadding,
        s=label,
        ha=alignment,
        va='center',
        rotation=rotation,
        rotation_mode="anchor")
```



In [16]:

```
# import pandas for data wrangling
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Build a dataset
df = pd.DataFrame(
        {
            'Name': ['item ' + str(i) for i in list(range(1, 51)) ],
            'Value': np.random.randint(low=10, high=100, size=50)
        })

# Reorder the dataframe
df = df.sort_values(by=['Value'])

# initialize the figure
plt.figure(figsize=(20,10))
ax = plt.subplot(111, polar=True)
plt.axis('off')

# Constants = parameters controling the plot layout:
upperLimit = 100
lowerLimit = 30
labelPadding = 4

# Compute max and min in the dataset
max = df['Value'].max()

# Let's compute heights: they are a conversion of each item value in those new coordinates
# In our example, 0 in the dataset will be converted to the lowerLimit (10)
# The maximum will be converted to the upperLimit (100)
slope = (max - lowerLimit) / max
heights = slope * df.Value + lowerLimit

# Compute the width of each bar. In total we have 2*Pi = 360°
width = 2*np.pi / len(df.index)

# Compute the angle each bar is centered on:
indexes = list(range(1, len(df.index)+1))
angles = [element * width for element in indexes]
angles

# Draw bars
bars = ax.bar(
    x=angles,
    height=heights,
    width=width,
    bottom=lowerLimit,
    linewidth=2,
    edgecolor="white",
    color="#61a4b2",
)

# Add labels
for bar, angle, height, label in zip(bars,angles, heights, df["Name"]):

    # Labels are rotated. Rotation must be specified in degrees :(
    rotation = np.rad2deg(angle)

    # Flip some labels upside down
    alignment = ""
    if angle >= np.pi/2 and angle < 3*np.pi/2:
        alignment = "right"
        rotation = rotation + 180
    else:
        alignment = "left"

    # Finally add the labels
    ax.text(
        x=angle,
        y=lowerLimit + bar.get_height() + labelPadding,
        s=label,
        ha=alignment,
        va='center',
        rotation=rotation,
        rotation_mode="anchor")
```



In [ ]:

Kullanım Hataları:

Yapılan en büyük hatalardan bir çubuk grafiklerinin histogram ile karıştırlmasıdır. Histogramlar tek bir kategorideki verinin sayısal dğerleridir.

Bar çubuklarının değerleri arasında bir sırlama olmaması verinin görselleştirilmesini zorlaştıracağından verileri öncelikel sıralamak daha uygun olacaktır.

In [ ]:

Her bir grupta çok fazla değer olması durumunda box plot veya violin kullanmak daha mantıklıdır.

In [ ]:

In [ ]: