# 20230323_Wirtschaftsinformatik_MPW2

March 23, 2023

[1]:
```
# Graphische Darstellung von Daten
```

[2]:
```
# Package. MATPLOTLIB
```

[3]:
```
!pip install matplotlib # wir downloaden und installieren MATPLOTLIB
```

```
Requirement already satisfied: matplotlib in
/Users/h4/anaconda3/lib/python3.9/site-packages (3.5.1)
Requirement already satisfied: packaging>=20.0 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from matplotlib) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from matplotlib) (3.0.4)
Requirement already satisfied: cycler>=0.10 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from matplotlib) (1.4.2)
Requirement already satisfied: pillow>=6.2.0 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from matplotlib) (9.0.1)
Requirement already satisfied: numpy>=1.17 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from matplotlib) (1.23.2)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in
/Users/h4/anaconda3/lib/python3.9/site-packages (from python-
dateutil>=2.7->matplotlib) (1.16.0)
```

[4]:
```
import matplotlib.pyplot as plt #wir rufen den Package matplotlib.pyplot auf
```

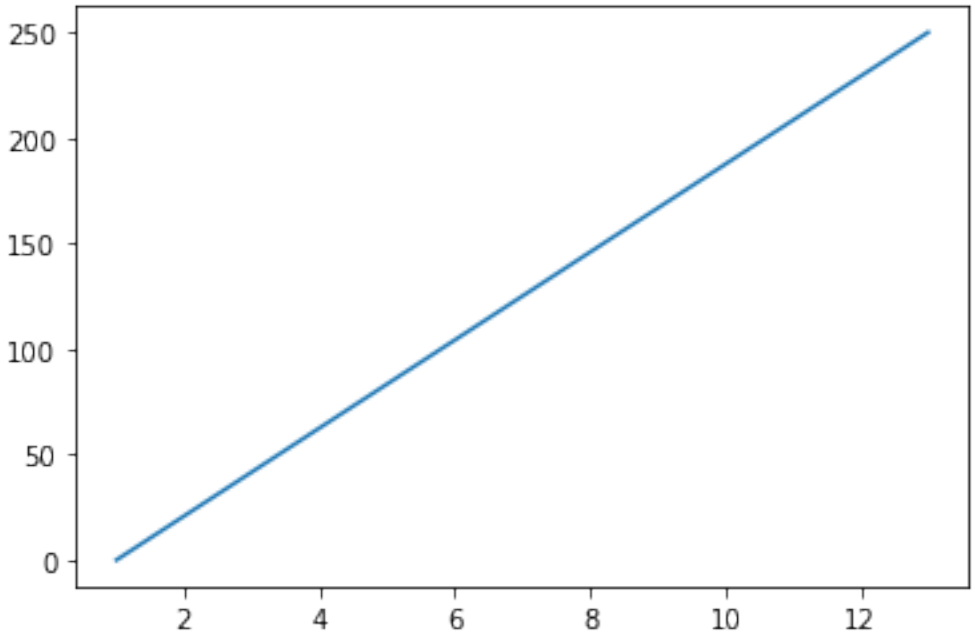
[5]:
```
!pip install numpy # dadurch installieren wir nummerisches python "numpy"
```

```
Requirement already satisfied: numpy in /Users/h4/anaconda3/lib/python3.9/site-
packages (1.23.2)
```

[6]:
```
import numpy as np
```

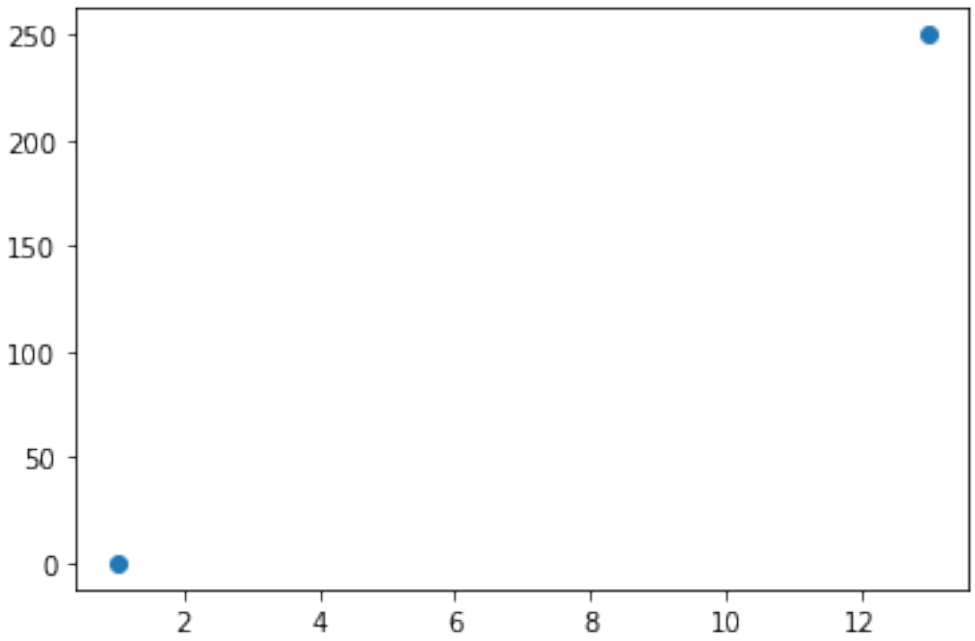
[7]:
```
# beispiel: eine Linie erstellen durch zwei Punkte
```

[8]:
```
x_koord = np.array([1, 13]) # erstellen wir einen Vektor np.array durch die [],␣
 ↪und dann eine liste innerhalb ()
y_koord = np.array([0, 250])

plt.plot(x_koord, y_koord) # plt.plot erzeugt eine linie, wenn die Koordinaten␣
 ↪als arrays gegeben werden
plt.show()
```



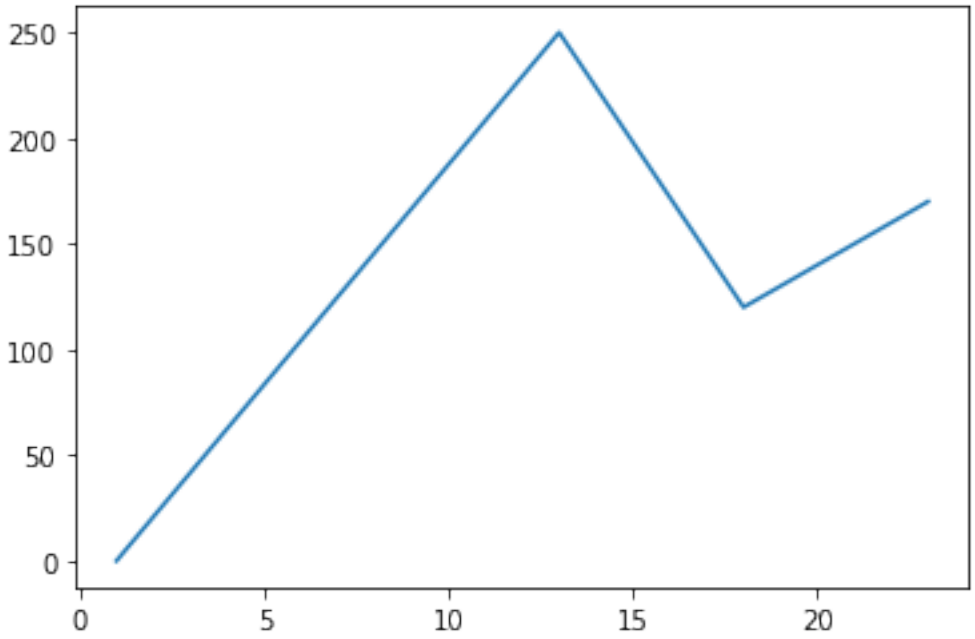

[9]:
```
# beispiel: darstellung von zwei Punkten ohne Linie
```

[12]:
```
x_koord = np.array([1, 13]) # erstellen wir einen Vektor np.array durch die [],␣
 ↪und dann eine liste innerhalb ()
y_koord = np.array([0, 250])

plt.plot(x_koord, y_koord, 'o')
plt.show()
```



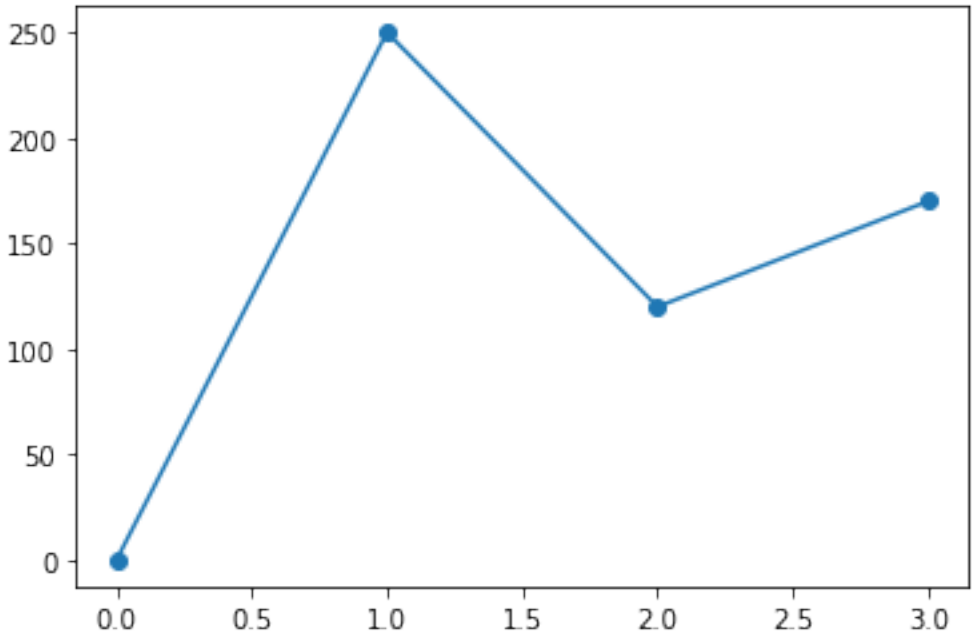

[13]:
```
# beispiel linie mit mehreren Eckpunkten
```

[14]:
```
x_koord = np.array([1, 13, 18, 23]) # erstellen wir einen Vektor np.array durch
↪die [], und dann eine liste innerhalb ()
y_koord = np.array([0, 250, 120, 170])

plt.plot(x_koord, y_koord) # plt.plot erzeugt eine linie, wenn die Koordinaten
↪als arrays gegeben werden
plt.show()
```



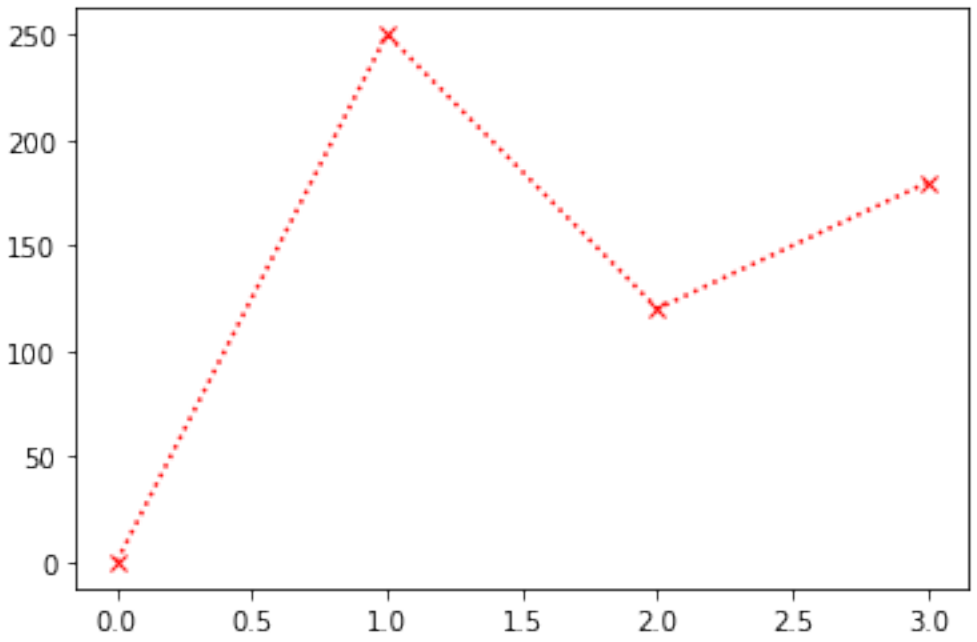

[15]:
```
# beispiel linie+punkte mit einem "marker"
```

[16]:
```
y_koord = np.array([0, 250, 120, 170])
plt.plot(y_koord, marker = 'o')
# per default versteht er die x koordinate als fortlaufend
plt.show()
```



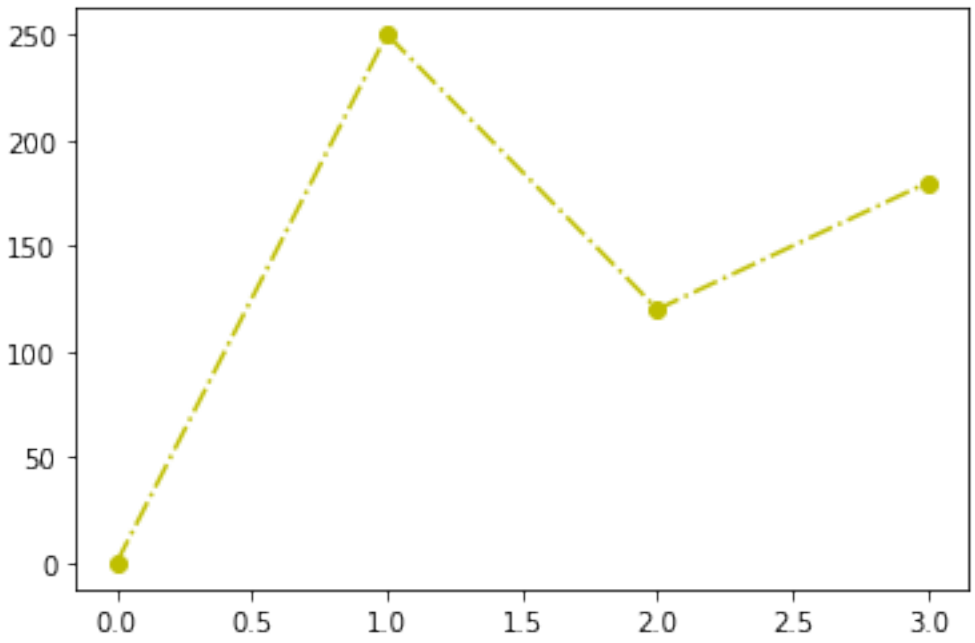
[17]:
```
# beispiel mit einer roten linie durchgestrichen,
#welche durch 4 punkte geht, wobei die 4 punkte mit einem "X"
# besonders darstellt werden
```

[19]:
```
y_koord = ([0, 250, 120, 180])
plt.plot(y_koord, 'x:r')
# "x" steht für den X marker
# ":" steht für doie Linie
# "r" steht für "rot"
plt.show()
```



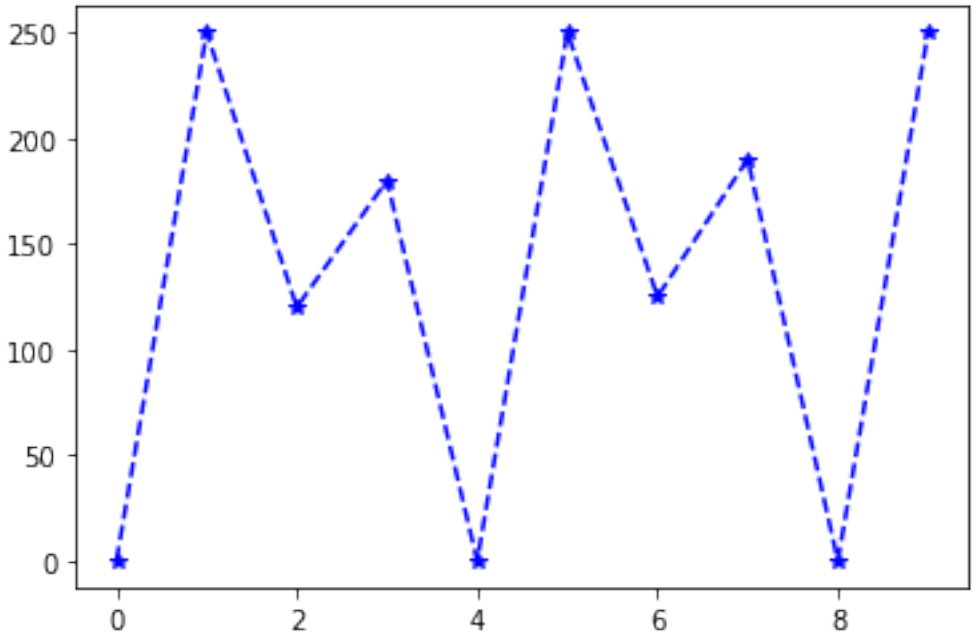

[20]:
```
# beispiel mit einer gelben linie: "linie-punkt"
```

[22]:
```
y_koord = np.array([0, 250, 120, 180])
plt.plot(y_koord, 'o-.y')
# "o" steht für marker "o"
# "-." steht für die linie
# "y" steht für "yellow"
plt.show()
```



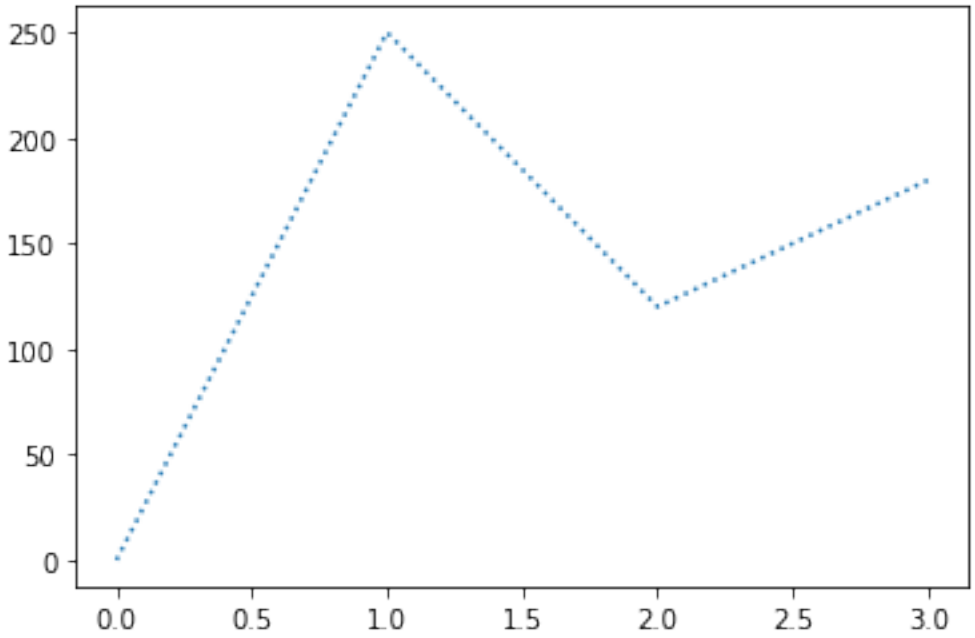

[23]:
```
# beispiel prüfung: bitte stellen Sie eine linie mit farbe blau,
# 10 Punkte, Linie --, marker "*"
```

[24]:
```
y_koord = np.array([0, 250, 120, 180, 0, 250, 125, 190, 0, 250])
# hier werden die Y-Koordinaten von 10 Punkten definiert
plt.plot(y_koord, "*--b")
# hier werden mit "*" die Marker definiert
# hier wird mit "--" die Linie definiert
# hier wird mit "b" die Farbe blau festgelegt
plt.show()
```



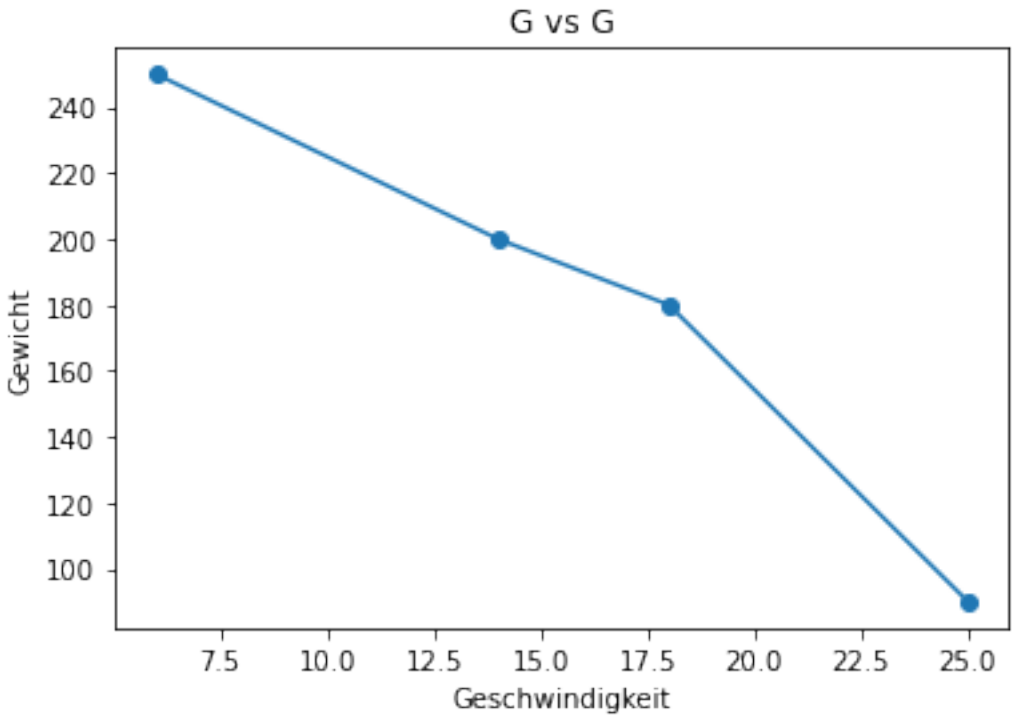

[25]:
```
# beispiel mit linestyle "dotted"
```

[26]:
```
y_koord = np.array([0, 250, 120, 180])
# hier werden die Y-Koordinaten von 4 Punkten definiert
plt.plot(y_koord, linestyle = 'dotted')
plt.show()
```



[27]:
```
# beispiel mit "labels"
```

[36]:
```
x_koord = np.array([6, 14, 18, 25]) # [] für vektor und () für np.array
y_koord = np.array([250, 200, 180, 90])

plt.plot(x_koord, y_koord, marker = 'o') # zunächst die x_koordinaten, dann die␣
 ↪y_koordinaten

plt.xlabel('Geschwindigkeit') # label auf X-Achse
plt.ylabel('Gewicht') # label auf Y-Achse
plt.title('G vs G') # title

plt.show()
```

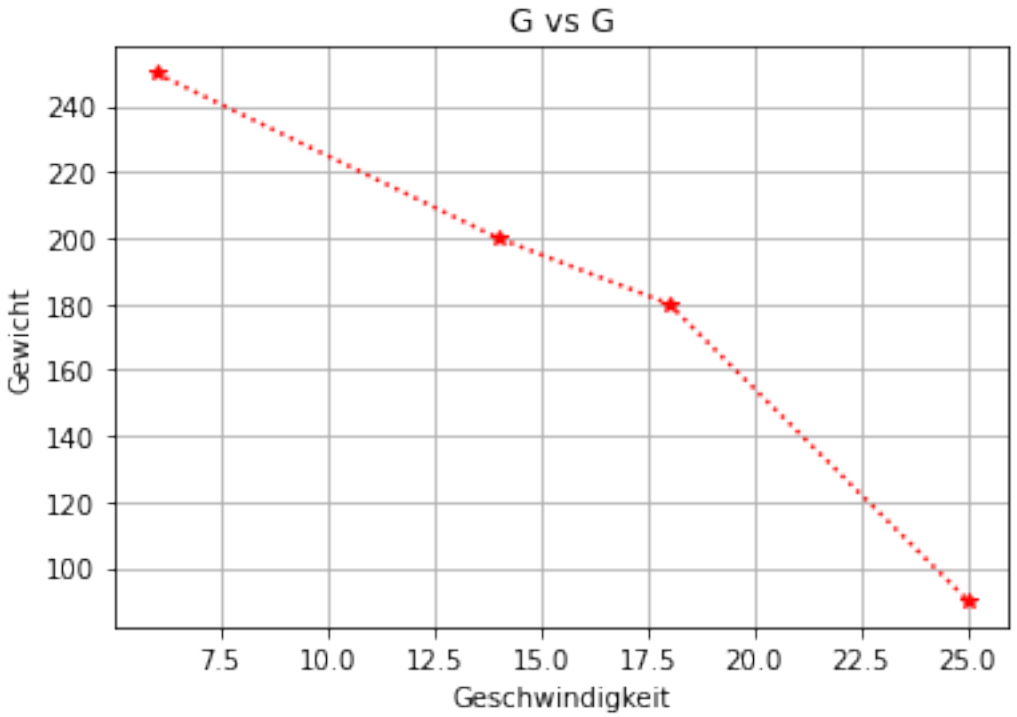



[37]:
```
# beispiel prüfungsfrage: bitte stellen Sie eine Graphik mit Python wie oben␣
 ↪gezeigt.
```

[38]:
```
# beispiel mit "Grid"
```

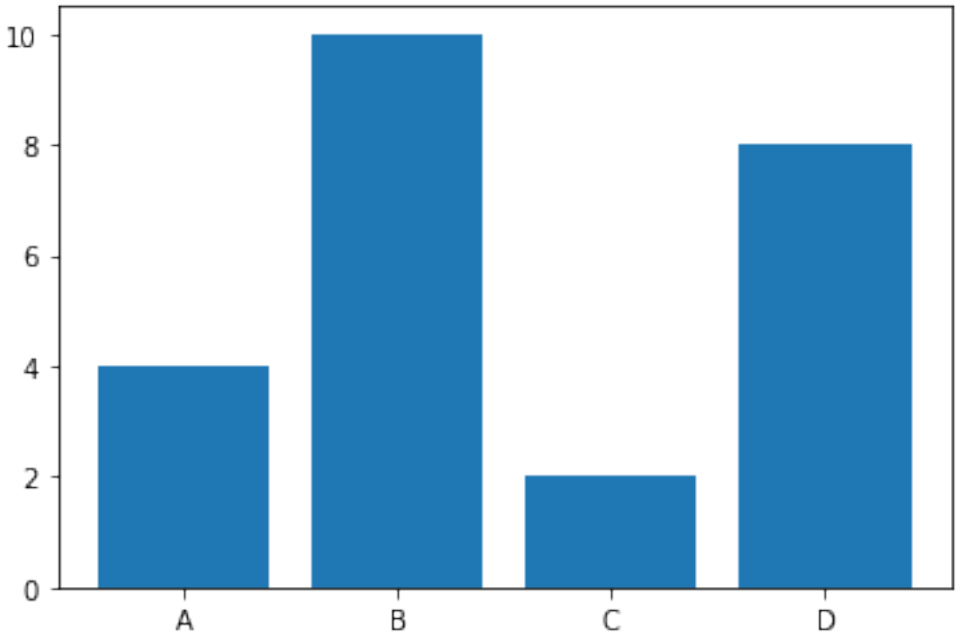
[39]:
```
x_koord = np.array([6, 14, 18, 25]) # [] für vektor und () für np.array
y_koord = np.array([250, 200, 180, 90])
```

```
plt.plot(x_koord, y_koord, '*:r') # zunächst die x_koordinaten, dann die␣
 ↪y_koordinaten

plt.xlabel('Geschwindigkeit') # label auf X-Achse
plt.ylabel('Gewicht') # label auf Y-Achse
plt.title('G vs G') # title

plt.grid() # hilfelinien werden erstellt

plt.show()
```



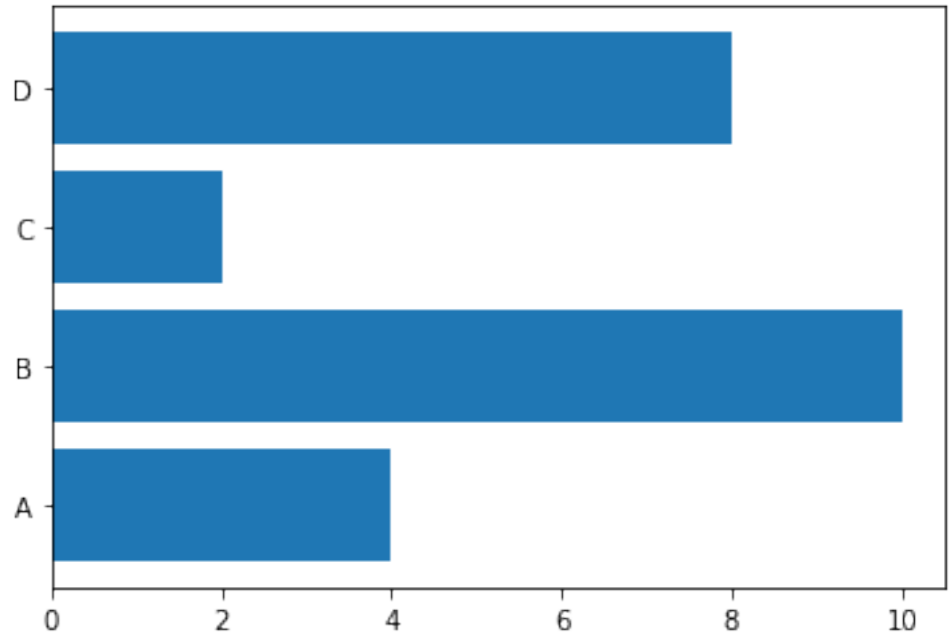

[42]:
```
# balkendiagramm vertikal
```

[43]:
```
x = np.array(['A', 'B', 'C', 'D']) # Kategorien
y = np.array([4, 10, 2, 8]) # Höhe vom Balken

plt.bar(x,y) #balkendiagramm
plt.show()
```



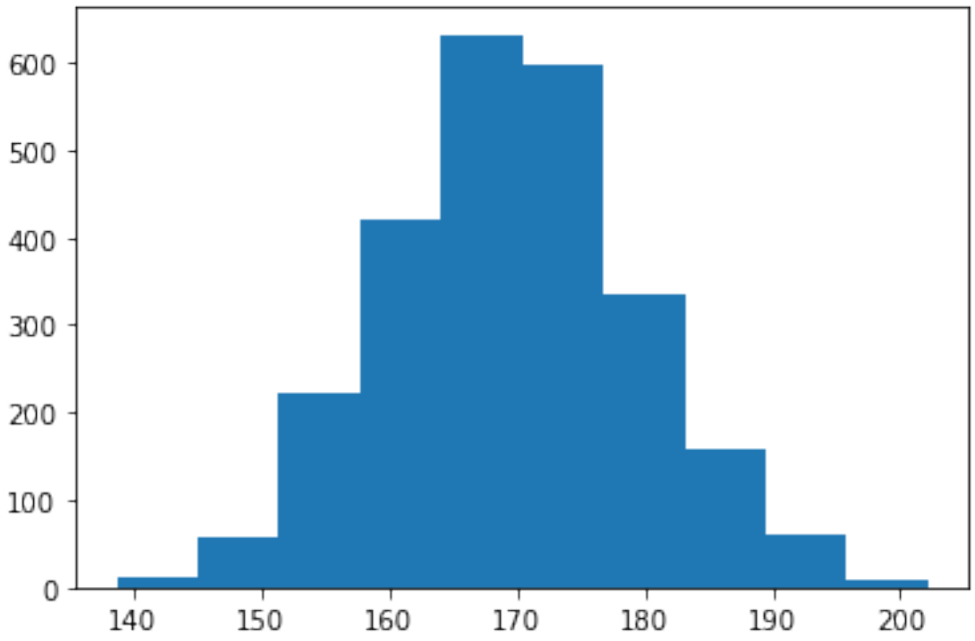

[44]:
```
# balkendiagramm horizontal
```

[46]:
```
x = np.array(['A', 'B', 'C', 'D']) # Kategorien
y = np.array([4, 10, 2, 8]) # Höhe vom Balken

plt.barh(x,y) #balkendiagramm horizontal
plt.show()
```



[47]:
```
# beispiel "histogram"
```

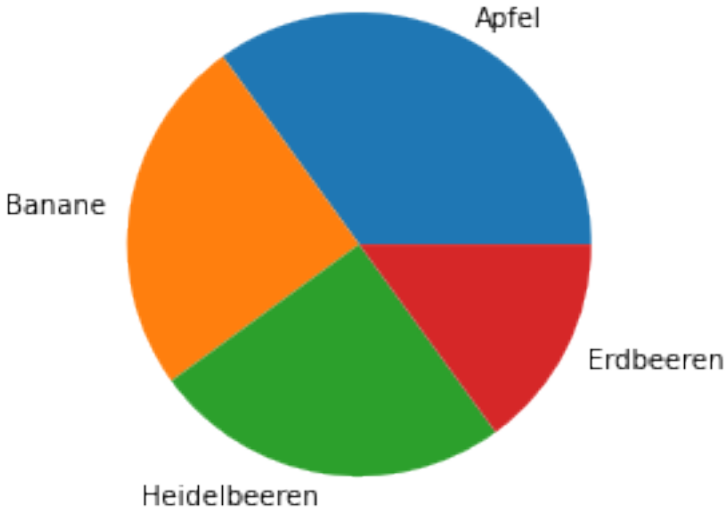
[48]:
```
x = np.random.normal(170, 10, 2500)
# zufallszahlen in einer Normalverteilung mit Mittelwert 170,
# Std Abweichung 10 und 2500 Datensätze

plt.hist(x) # hier wird ein Histogram dargestellt
plt.show()
```




[52]:
```
# beispiel Kuchendiagramm
```

[53]:
```
y = np.array([12, 25, 15, 29])

plt.pie(y) # hier wird ein Kuchendiagramm dargestellt
plt.show()
```

[54]:
```
# beispiel Kuchendiagramm mit Labels

y = np.array([35, 25, 25, 15])
mylabels = ['Apfel', 'Banane', 'Heidelbeeren', 'Erdbeeren']
plt.pie(y, labels = mylabels) # hier wird ein Kuchendiagramm dargestellt
plt.show()
```



[ ]: