20211611_Informationsmanagement_MV1

November 16, 2021

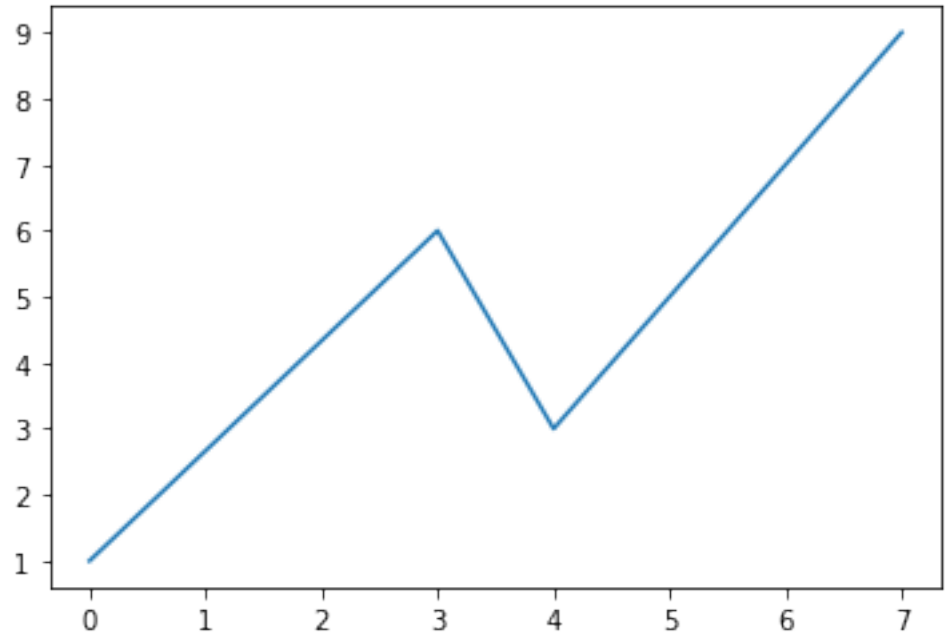
[1]:
```
# Graphische Darstellung von Daten
# MATPLOTLIB.PYPLOT
```

[2]:
```
!pip install matplotlib
```

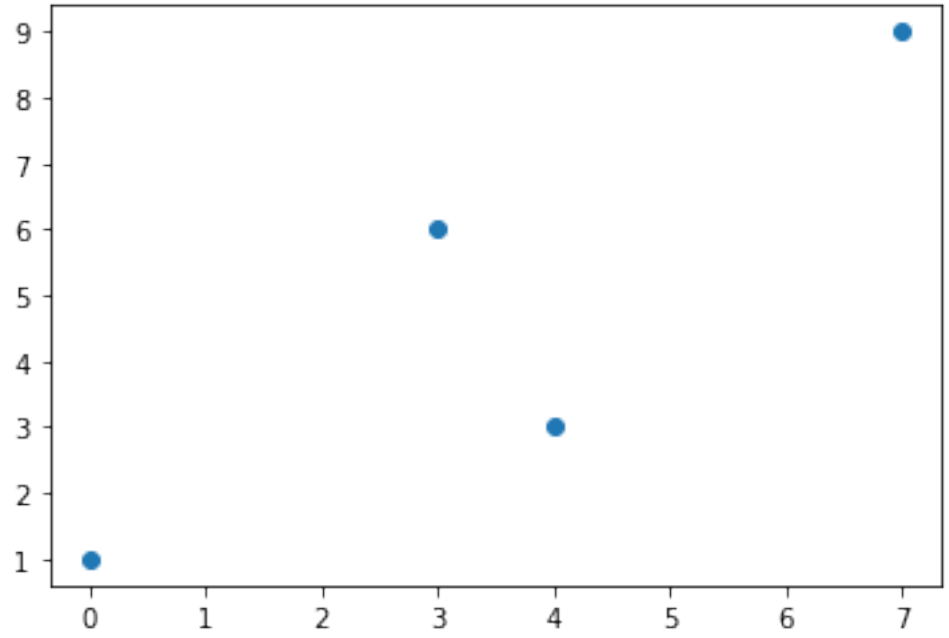
```
Requirement already satisfied: matplotlib in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (3.4.2)
Requirement already satisfied: pillow>=6.2.0 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (8.3.2)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: cycler>=0.10 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: numpy>=1.16 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (1.21.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: pyparsing>=2.2.1 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: six in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from
cycler>=0.10->matplotlib) (1.16.0)
```

[3]:
```
import matplotlib.pyplot as plt
import numpy as np
```

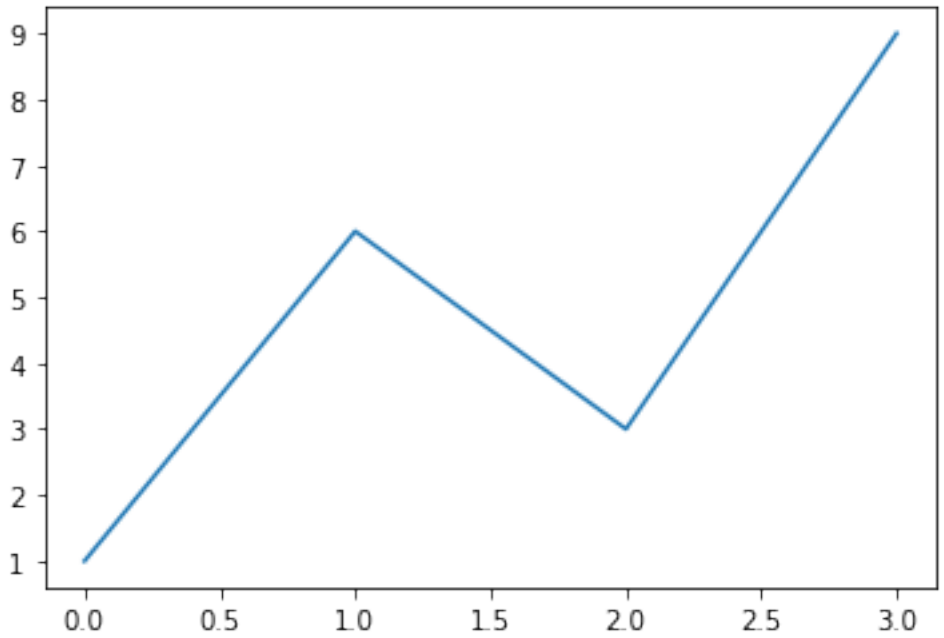
[14]:
```
# Beispiel Plot einer Linie

xpoints = np.array([0,3,4,7])
ypoints = np.array([1,6,3,9])

plt.plot(xpoints, ypoints) # plot steht für "zeichnen"
plt.show()                 # show wird gebraucht, damit die Graphik gezeigt wird
```



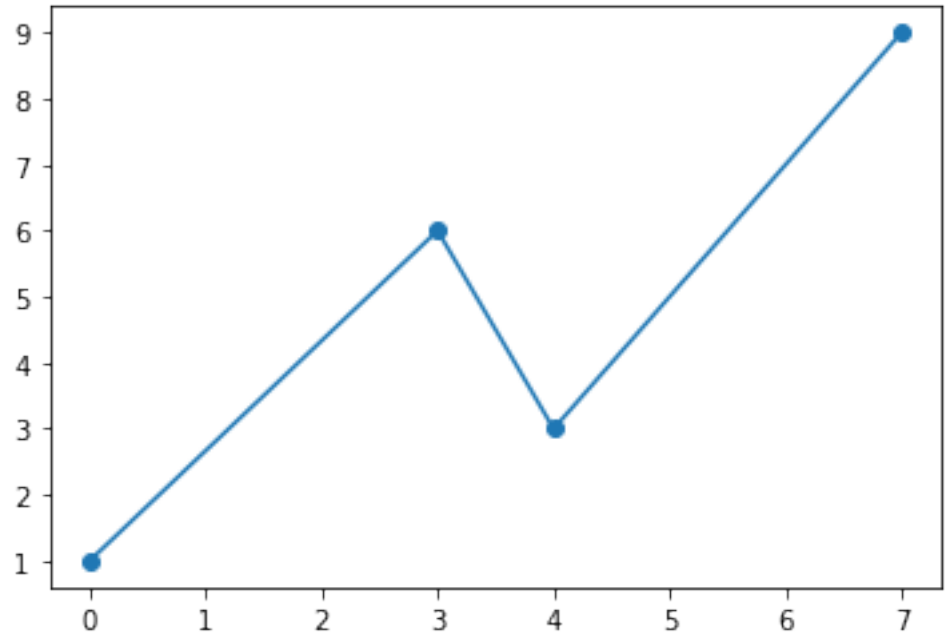
[5]:
```
# ohne Linie, nur die Punkte

plt.plot(xpoints, ypoints, 'o')
plt.show()
```



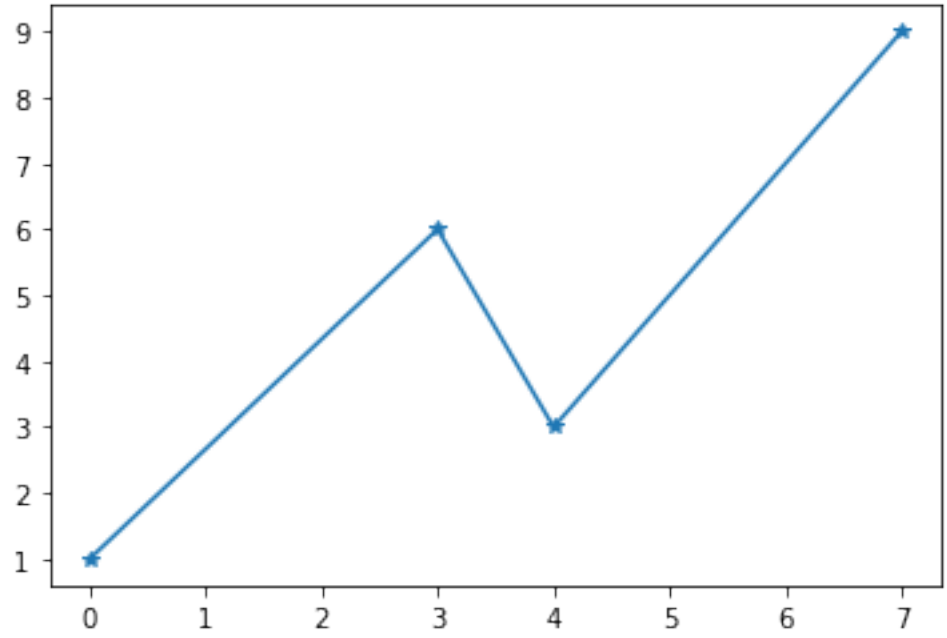
[6]:
```
# default nur die y--points geben. die x--points werden als Positionen␣
↪dargestellt

plt.plot(ypoints)
plt.show()
```



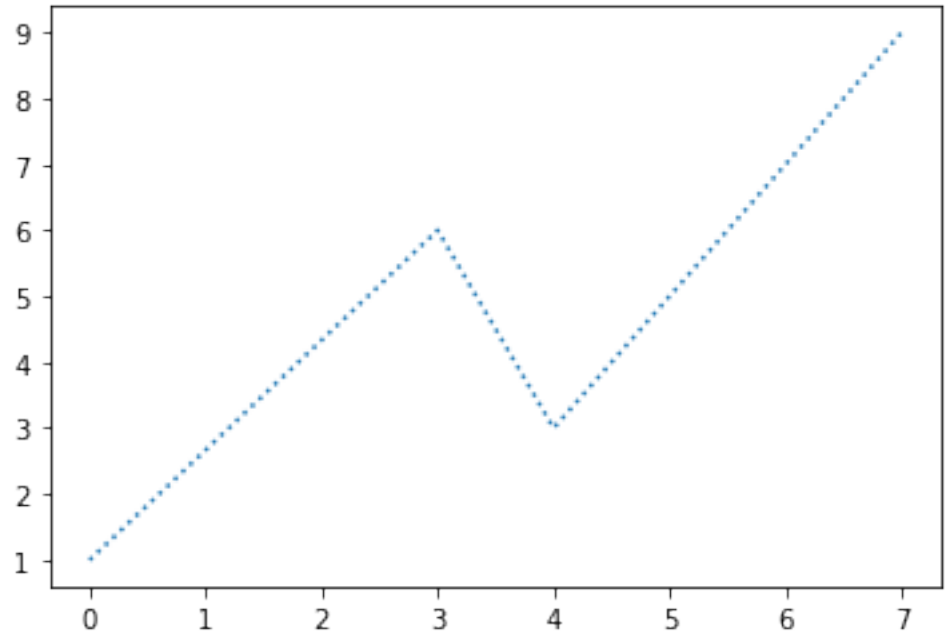
[7]:
```
# Punkte mit einem 'o' darstellen und die Linie

plt.plot(xpoints, ypoints, marker='o')
plt.show()
```



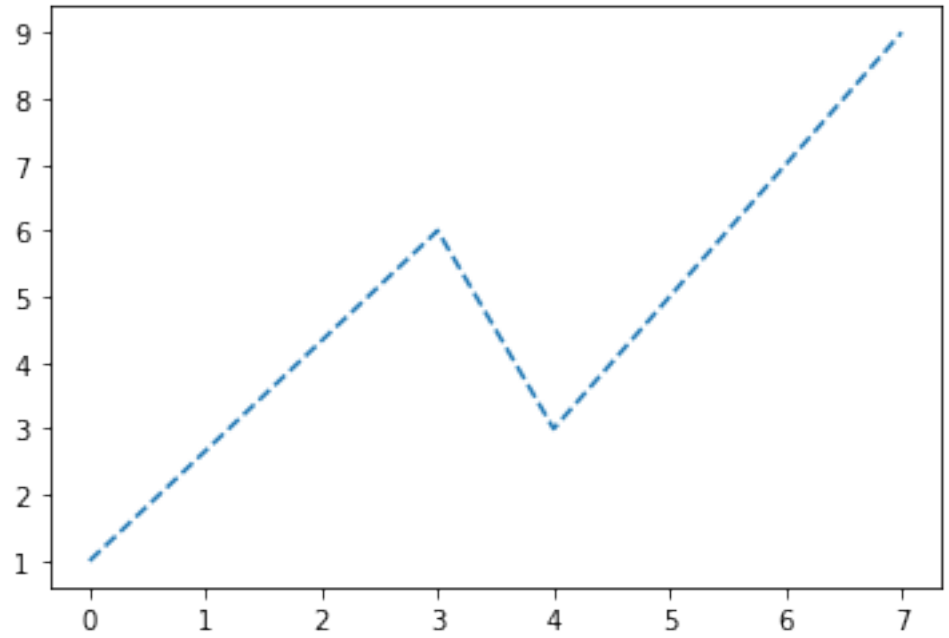
[44]:
```
# marker kann auf einem Stern '*' geändert werden

plt.plot(xpoints, ypoints, marker='*')
plt.show()
```



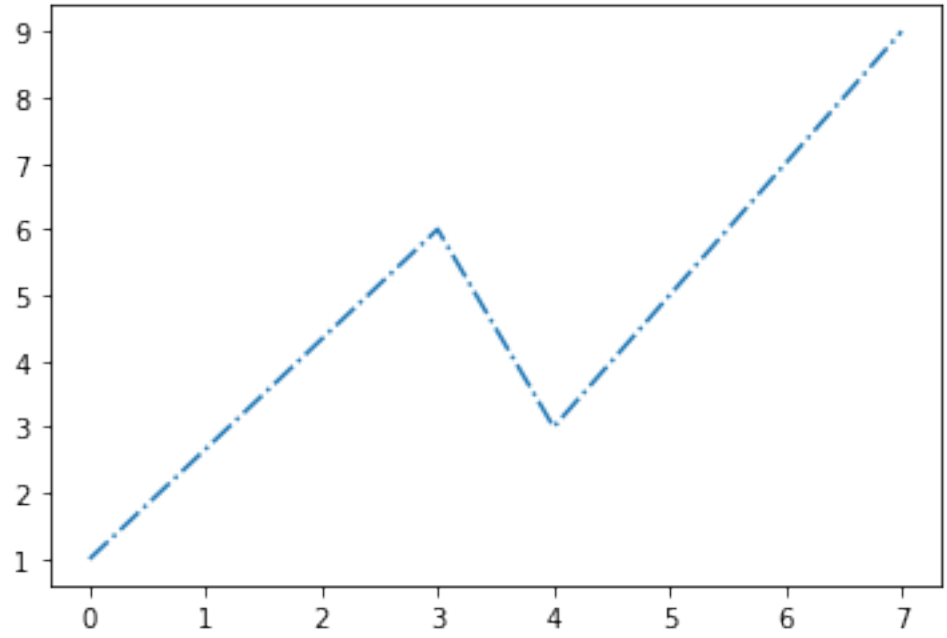
[9]:
```
# Linien Styl kann geändert werden

plt.plot(xpoints, ypoints, linestyle='dotted')
plt.show()
```



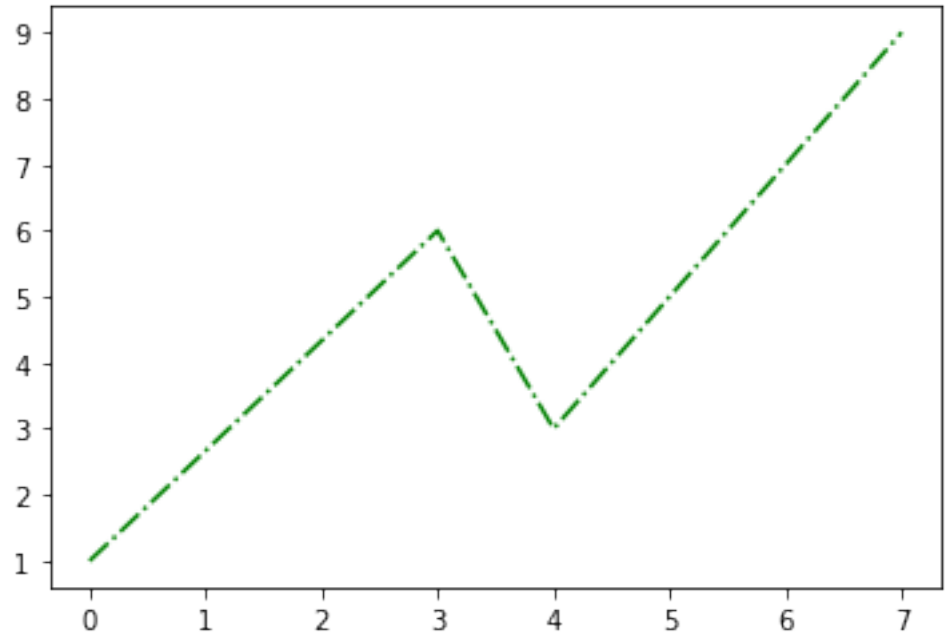
[10]:
```
plt.plot(xpoints, ypoints, linestyle='dashed')
plt.show()
```



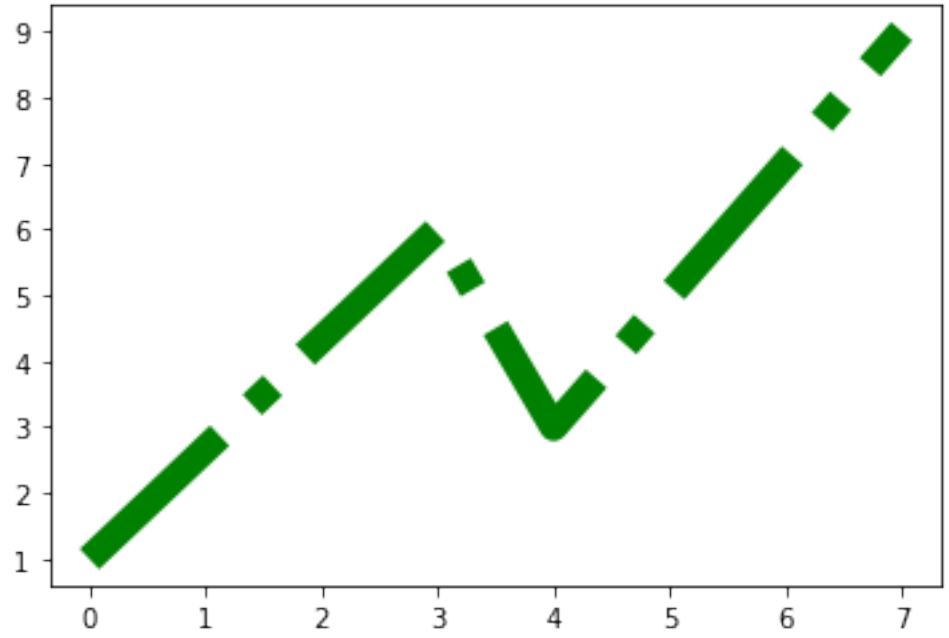
[11]:
```
plt.plot(xpoints, ypoints, linestyle='dashdot')
plt.show()
```



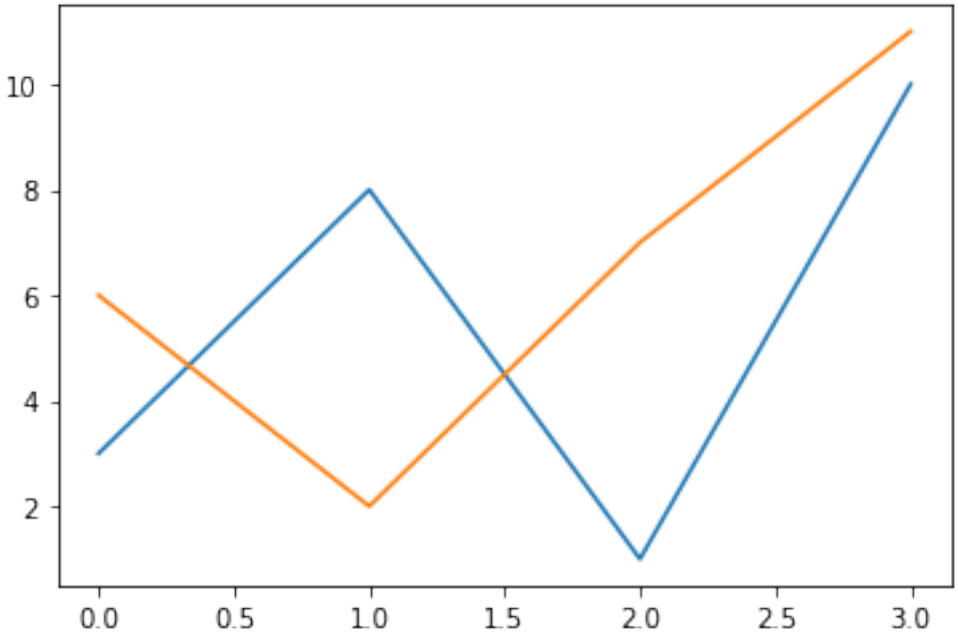
[12]:
```
# Linienfarbe kann auch angepasst werden

plt.plot(xpoints, ypoints, linestyle='dashdot', color='g')
plt.show()
```



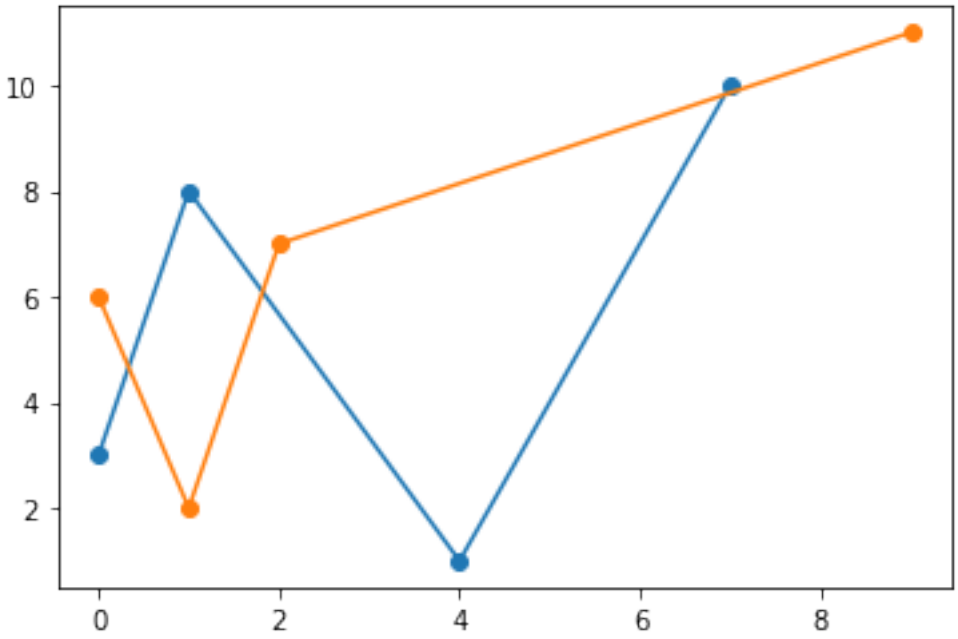
[15]:
```
# Liniendicke kann angepasst werden

plt.plot(xpoints, ypoints, linestyle='dashdot', color='g', linewidth='10.5')
plt.show()
```



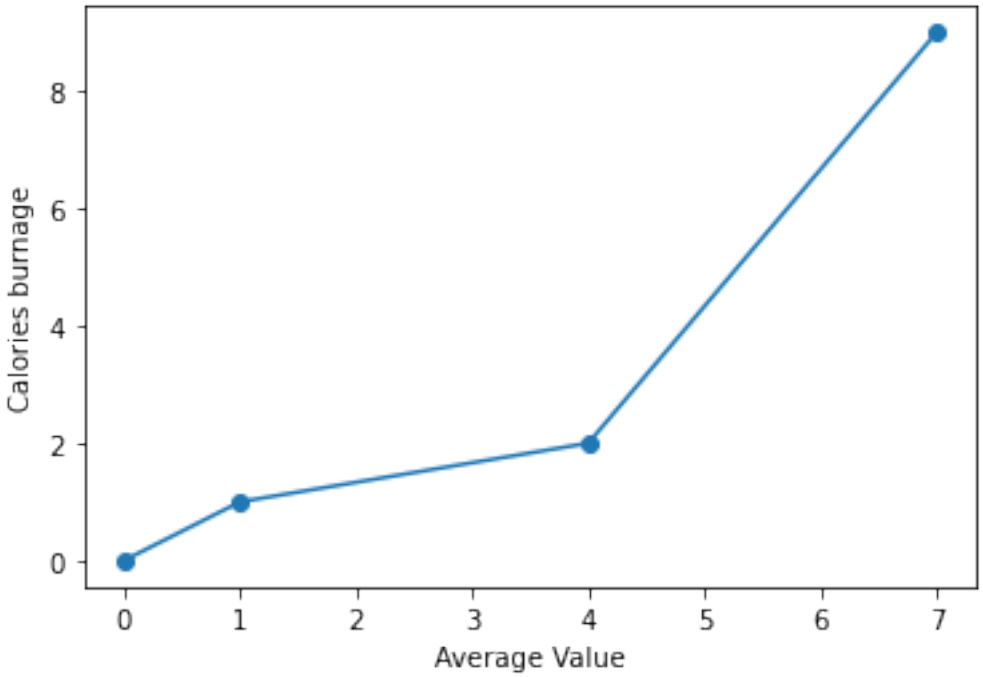
[18]:
```
# Mehrere Linien in einem Plot darstellen

y1 = np.array([3,8,1,10])
y2 = np.array([6,2,7,11])

plt.plot(y1)
plt.plot(y2)

plt.show()
```



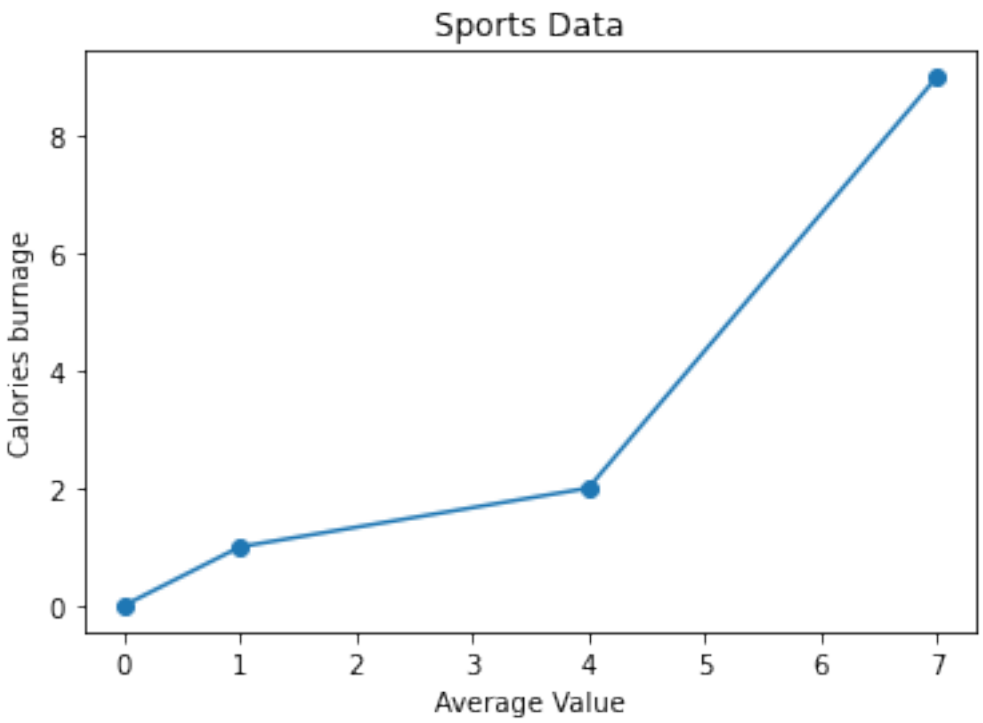
[22]:
```
# x Daten und y Daten können beliebig dargestellt werden

x1 = np.array([0,1,4,7])
y1 = np.array([3,8,1,10])

x2 = np.array([0,1,2,9])
y2 = np.array([6,2,7,11])

plt.plot(x1,y1,x2,y2, marker='o')

plt.show()
```



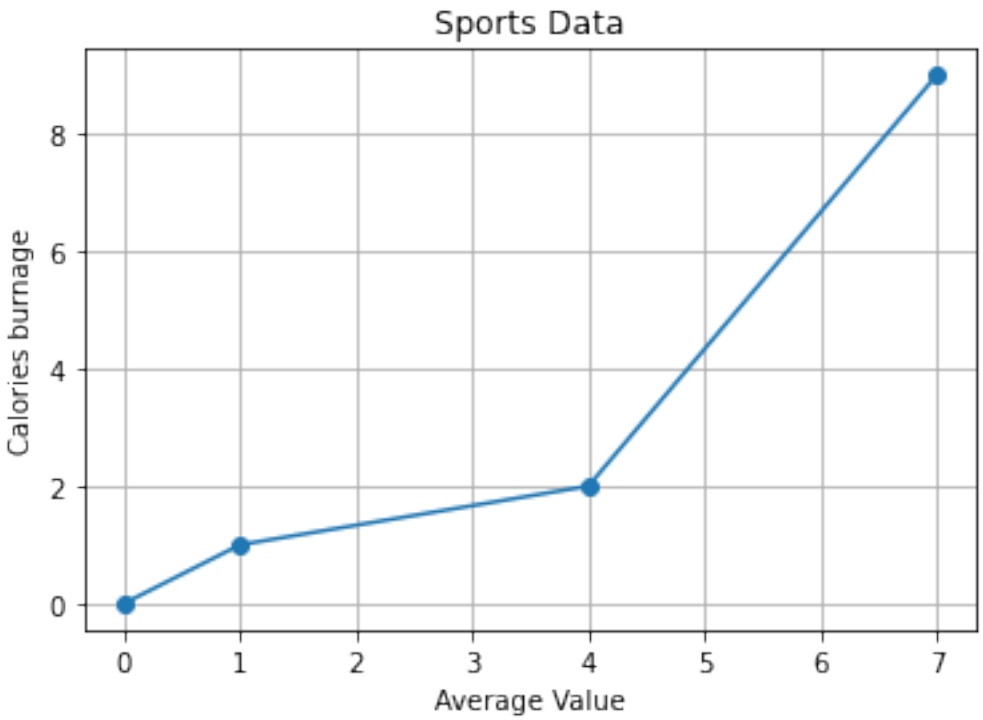
[23]:
```
# Labels können auch dargestellt werden (Namen von den Axen)

x1 = np.array([0,1,4,7])

y1 = np.array([0,1,2,9])

plt.plot(x1,y1,marker='o')

plt.xlabel('Average Value')
plt.ylabel('Calories burnage')

plt.show()
```



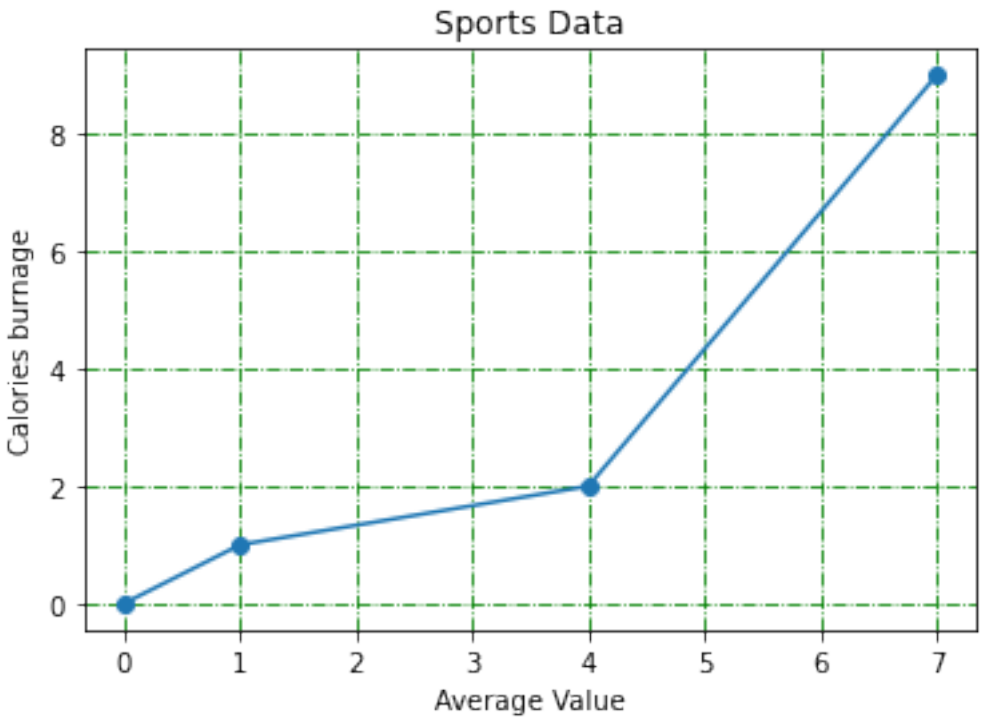
[24]:
```
# Titel hinfügen

x1 = np.array([0,1,4,7])

y1 = np.array([0,1,2,9])

plt.plot(x1,y1,marker='o')

plt.xlabel('Average Value')
plt.ylabel('Calories burnage')

plt.title('Sports Data')

plt.show()
```



[25]:
```
# Grid oder Hilfelinien einfügen

x1 = np.array([0,1,4,7])

y1 = np.array([0,1,2,9])

plt.plot(x1,y1,marker='o')

plt.xlabel('Average Value')
plt.ylabel('Calories burnage')

plt.title('Sports Data')

plt.grid()

plt.show()
```

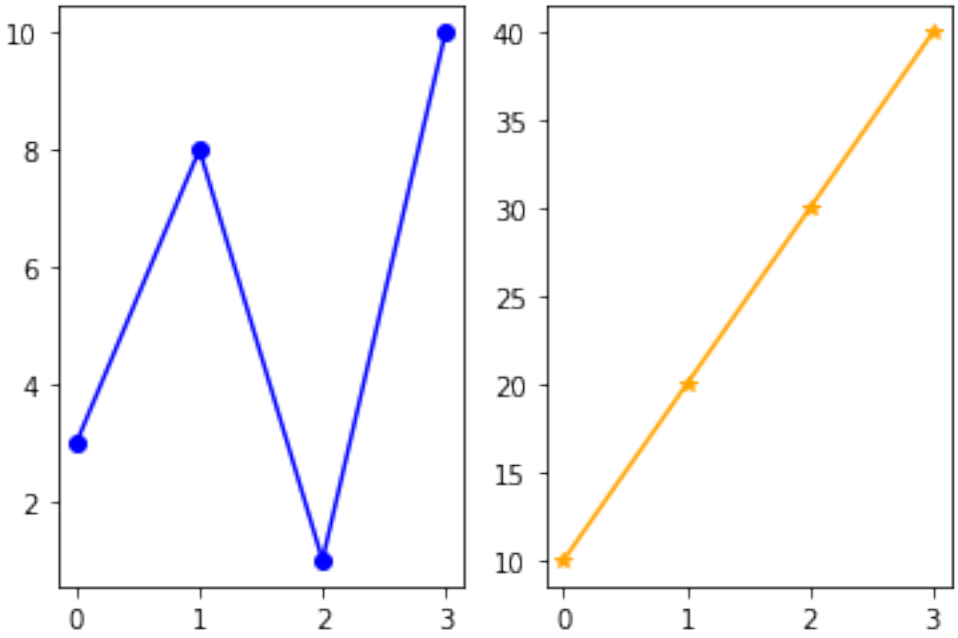


[28]:
```
# grid Farben und Linestyle können angepasst werden

x1 = np.array([0,1,4,7])

y1 = np.array([0,1,2,9])

plt.plot(x1,y1,marker='o')

plt.xlabel('Average Value')
plt.ylabel('Calories burnage')

plt.title('Sports Data')

plt.grid(color='g', linestyle='dashdot')

plt.show()
```

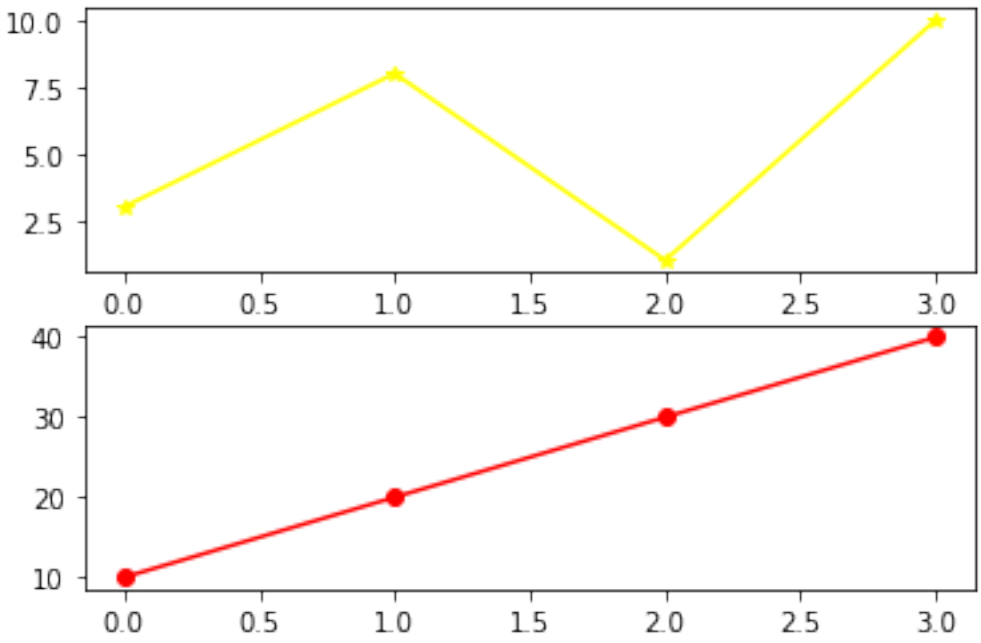


[36]:
```
# Mit dem Befehl "Subplots" können wir mehrere Plots in einem Bild darstellen.

# Die Suplots werden in einem Matrix Modell dargestellt.

# Die Subplots Funktion benötigt drei Argumente
# die das Layout abbilden (Anzahl Zeilen, Anzahl Spalten, Position)

# plot 1
x = np.array([0,1,2,3])
y = np.array([3,8,1,10])

plt.subplot(1,2,1)
# es wird ein Subplot von 1 Zeile, 2 spalten, und der erste ist in Position 1

plt.plot(x,y, marker='o', color='blue')

# plot 2
x = np.array([0,1,2,3])
y = np.array([10,20,30,40])

plt.subplot(1,2,2)
# es wird ein Subplot von 1 Zeile, 2 Spalten, und der zweite ist in Position 2
```

```
plt.plot(x,y, marker='*', color='orange')

plt.show()
```

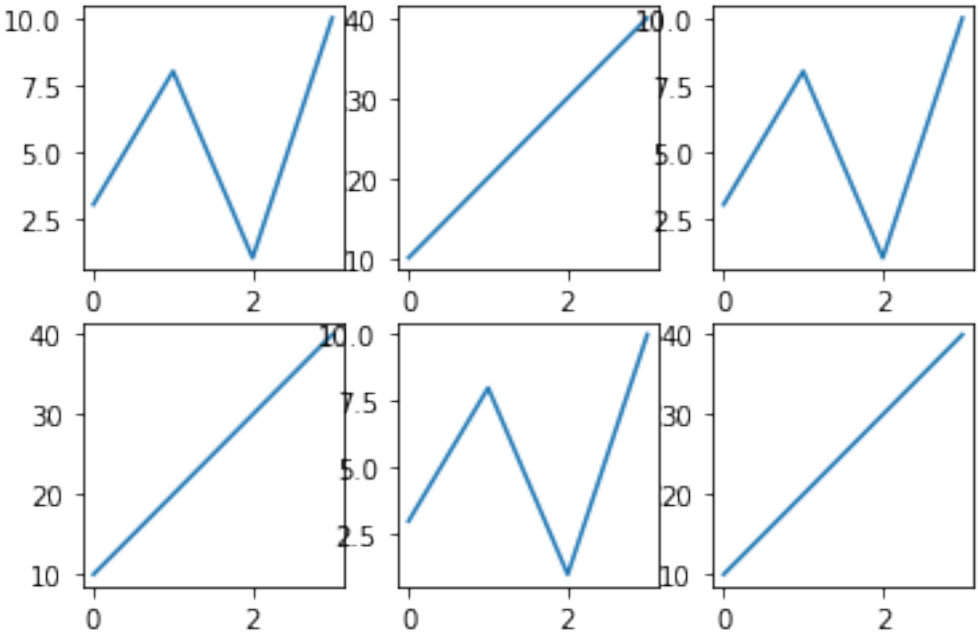


[39]:
```
 #plot 1
x = np.array([0,1,2,3])
y = np.array([3,8,1,10])
plt.subplot(2,1,1) # es ist ein Plot mit 2 Zeilen, 1 Spalte. Subplot in 1.␣␣
 ↪ →Position.
plt.plot(x,y, marker='*', color='yellow')
#plot 2
x = np.array([0,1,2,3])
y = np.array([10,20,30,40])
plt.subplot(2,1,2) # es ist ein Plot mit 2 Zeilen, 1 Spalte. Subplot in 2.␣␣
 ↪ →Position.
plt.plot(x,y, marker='o', color='red')
plt.show()
```



[41]:
```
# Beliebig viele Plots auf eine Figur zeichnen

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 3, 1) # es ist ein Plot mit 2 Zeilen, 3 Spalte. Subplot in 1.␣␣
 ↪ →Position.
plt.plot(x,y)
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 3, 2) # es ist ein Plot mit 2 Zeilen, 3 Spalte. Subplot in 2.␣␣
 ↪ →Position.
plt.plot(x,y)
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 3, 3) # es ist ein Plot mit 2 Zeilen, 3 Spalte. Subplot in 3.␣␣
 ↪ →Position.
plt.plot(x,y)
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 3, 4) # es ist ein Plot mit 2 Zeilen, 3 Spalte. Subplot in 4.␣␣
 ↪ →Position.
plt.plot(x,y)
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 3, 5)
```

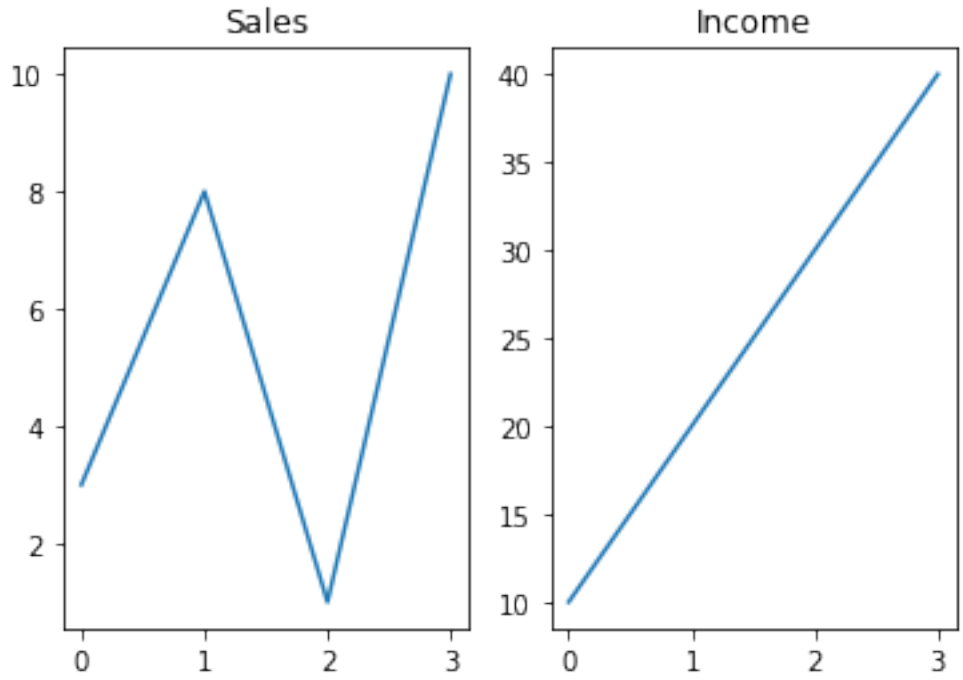
```
plt.plot(x,y)
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 3, 6)
plt.plot(x,y)
plt.show()
```



[40]:
```
# Übung. bitte erstelle einen Subplot mit 4 Zeilen und 2 Spalten
# In Jedem Subplot sollte eine Line mit Markers 'o'
# und unterscheidliche Farbe dargestellt werden.

# Hinweis: es gibt insgesamt 8 Linien. (4x2).
# Die Position der Linien ist fortlaufend.
```

[45]:
```
# Bei Jedem Plot ein titel einfügen

# Plot 1
x = np.array([0,1,2,3])
y = np.array([3,8,1,10])
plt.subplot(1,2,1) # es ist ein Plot mit 1 Zeilen, 2 Spalten. Subplot in 1.␣␣
 ↪ →Position.
plt.plot(x,y)
plt.title('Sales')

#plot 2
```

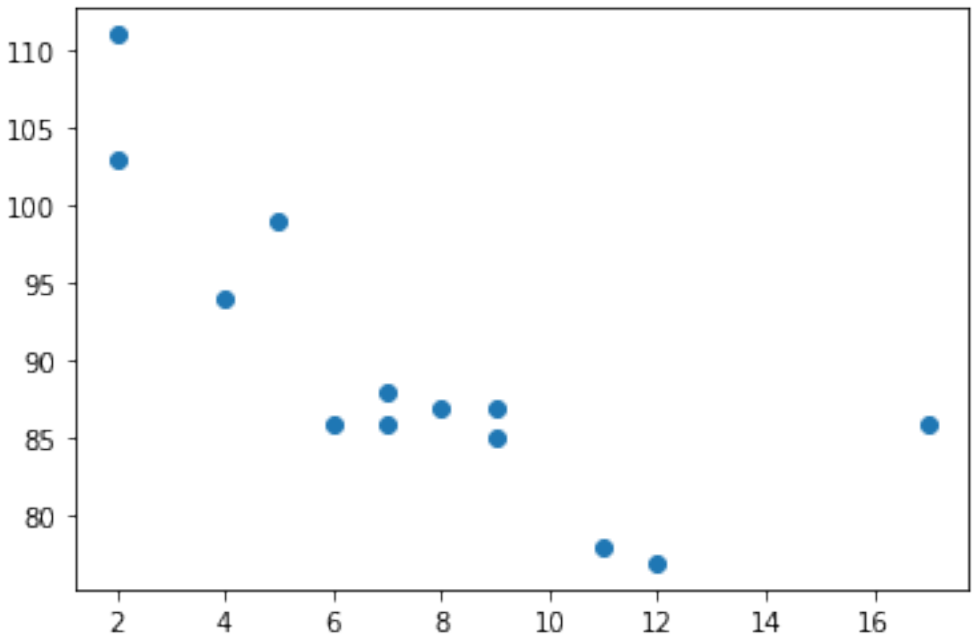
```
x = np.array([0,1,2,3])
y = np.array([10,20,30,40])
plt.subplot(1,2,2) # es ist ein Plot mit 1 Zeilen, 2 Spalten. Subplot in 2.␣␣
 ↪ →Position.
plt.plot(x,y)
plt.title('Income')

plt.show()
```



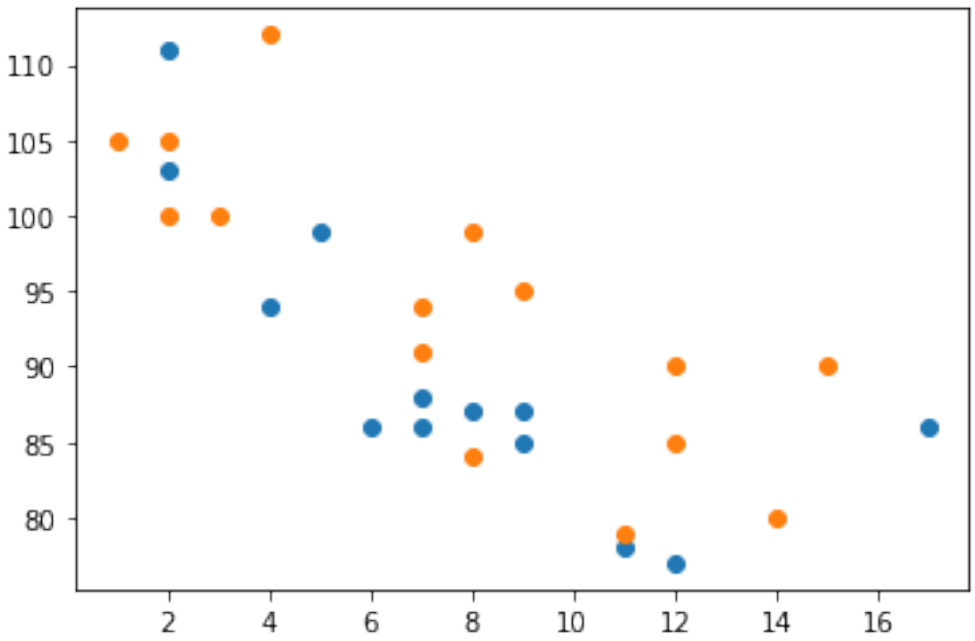
[46]:
```
# Gesamttitle kann dargestellt werden

# Plot 1
x = np.array([0,1,2,3])
y = np.array([3,8,1,10])
plt.subplot(1,2,1) # es ist ein Plot mit 1 Zeilen, 2 Spalten. Subplot in 1.␣␣
 ↪ →Position.
plt.plot(x,y)
plt.title('Sales')

#plot 2
x = np.array([0,1,2,3])
y = np.array([10,20,30,40])
plt.subplot(1,2,2) # es ist ein Plot mit 1 Zeilen, 2 Spalten. Subplot in 2.␣␣
 ↪ →Position.
```

```
plt.plot(x,y)
plt.title('Income')

plt.suptitle('My Shop')

plt.show()
```

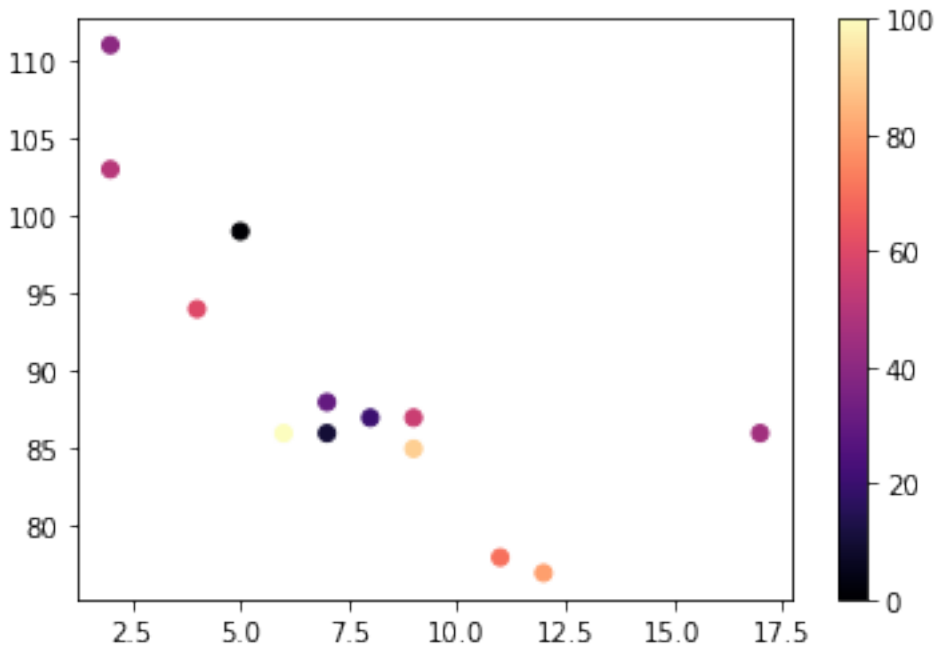
[47]:
```
# Erstellung von Streudiagramme (Scatterplot)

# Die Scatter Funktion zeichnet für Jede Beobachtung einen Punkt.
# Es benötigt gleiche Anzahl Punkte (gleiche Länge von Arrays)

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

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



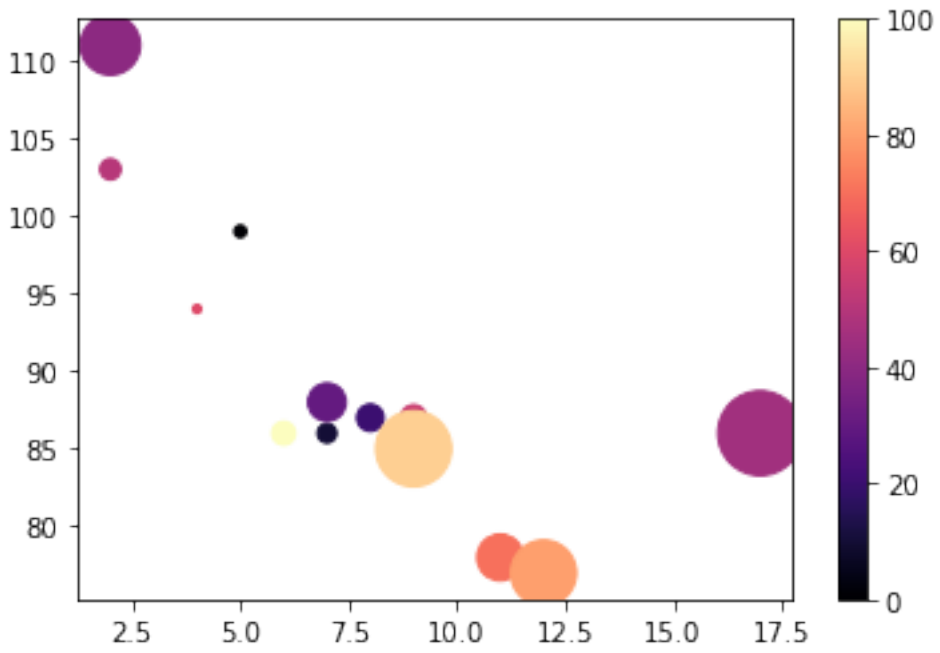
[48]:
```
# zwei Arten von Punkten in einer Graphik

# Tag 1 Daten
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)

# Tag 2 Daten
x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x, y)

plt.show()
```



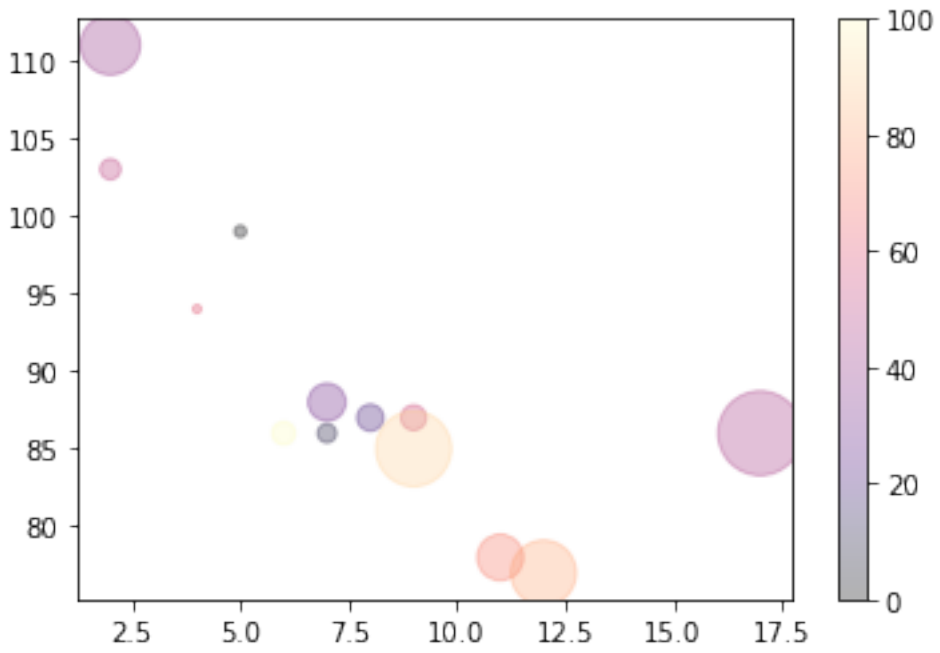
[52]:
```
# Color Map oder Legende darstellen

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x,y, c=colors, cmap='magma')

plt.colorbar()

plt.show()
```



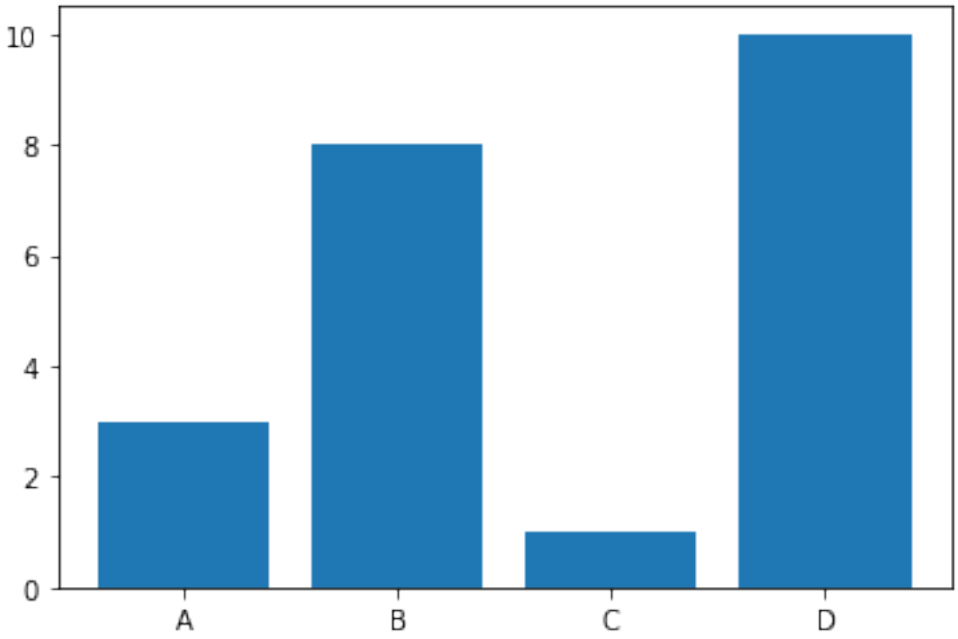
[53]:
```
# grösse von den Punkten selber bestimmen

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x,y, s=sizes, c=colors, cmap='magma')

plt.colorbar()

plt.show()
```



[54]:
```
# Transparenz in den Darstellung einbringen

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x,y, s = sizes, c = colors, cmap='magma', alpha = 0.3)
plt.colorbar()
plt.show()
```

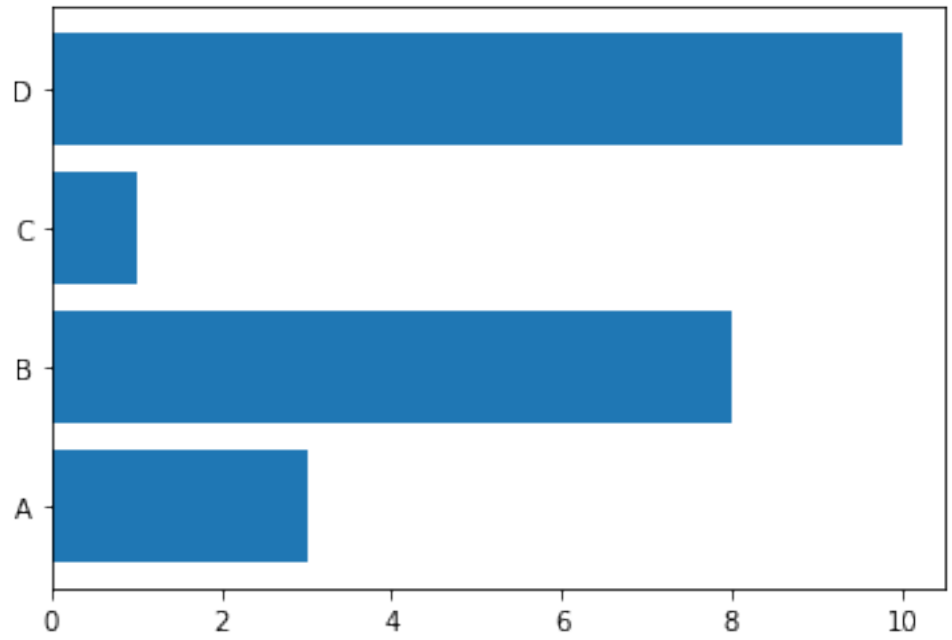


[55]:
```
# Balkendiagramme können mit 'bar()' dargestellt werden

x = np.array(['A', 'B', 'C', 'D'])
y = np.array([3,8,1,10])

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

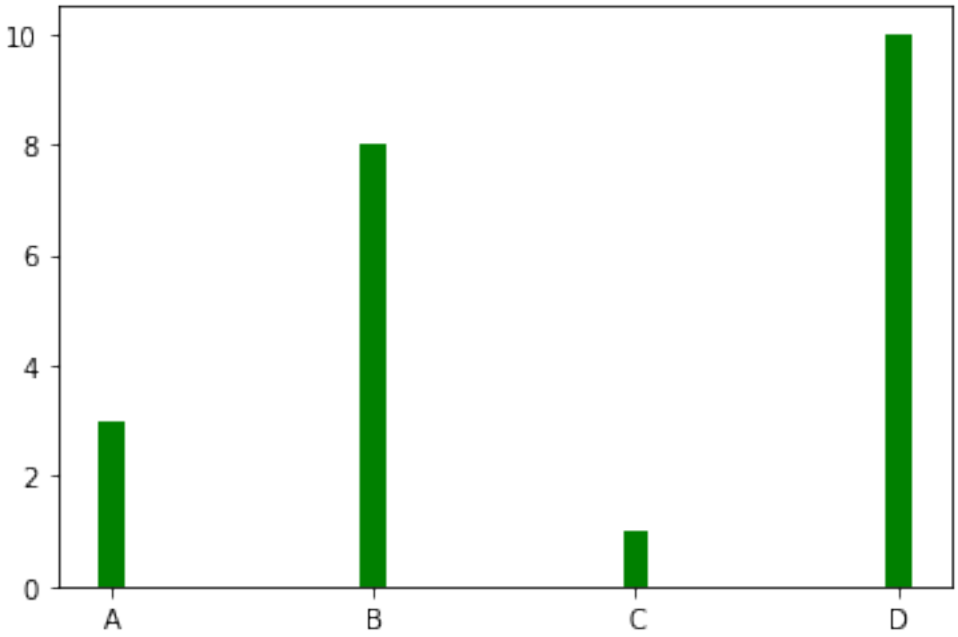


[56]:
```
# quer Balkendiagramme darstellen

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

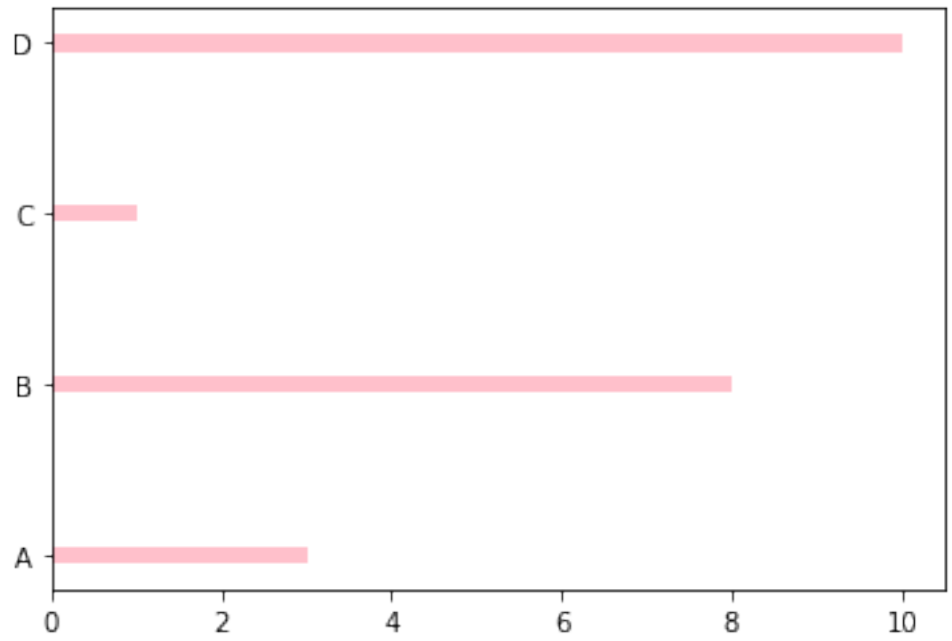
[56]: <BarContainer object of 4 artists>



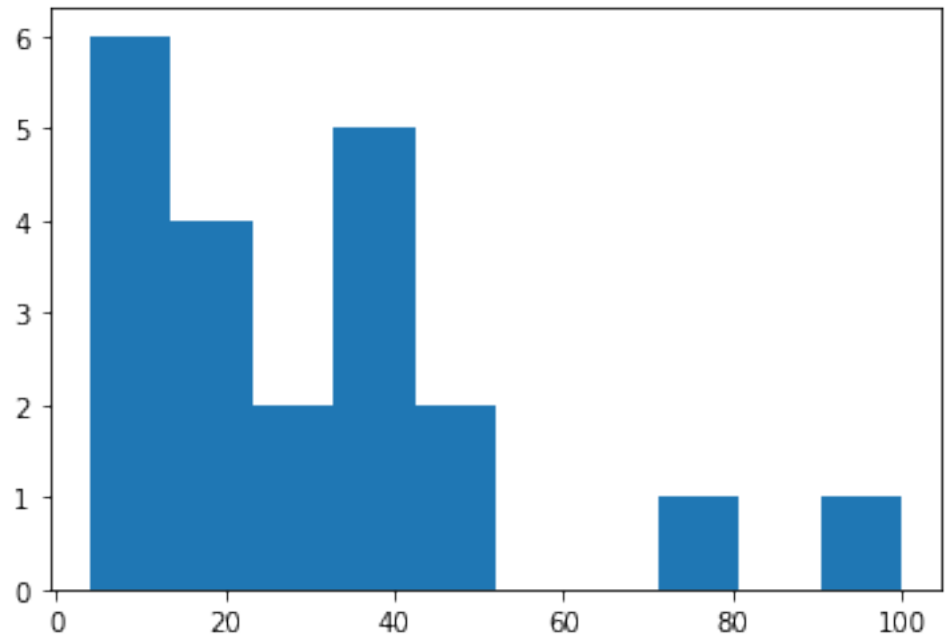
[57]:
```
# Balkenbreite kann angepasst werden

x = np.array(['A', 'B', 'C', 'D'])
y = np.array([3,8,1,10])
plt.bar(x,y, color='green', width=0.1)
plt.show()
```



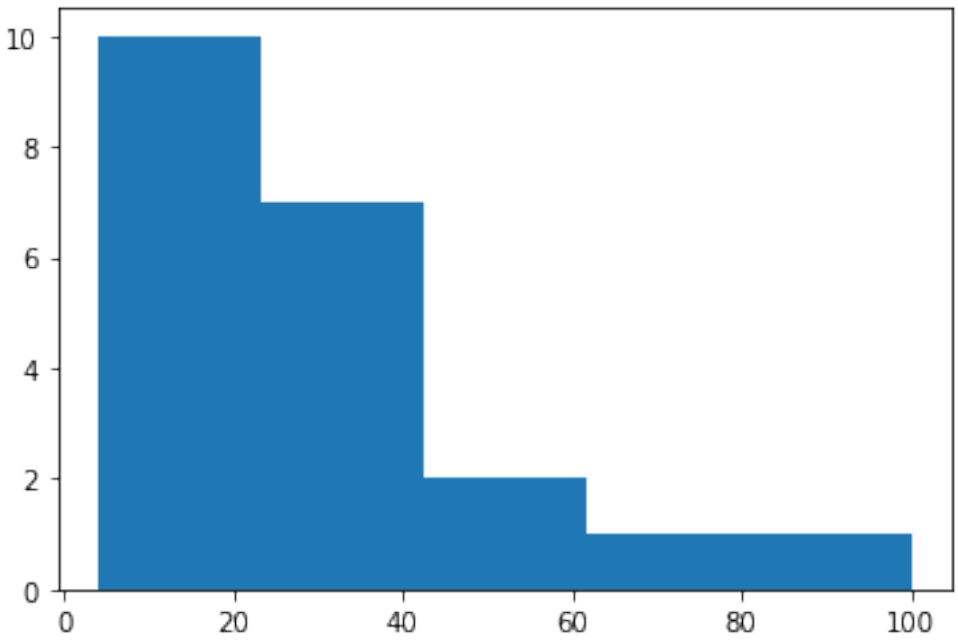
[58]:
```
# Balkenhöhe kann angepasst werden

x = np.array(['A', 'B', 'C', 'D'])
y = np.array([3,8,1,10])
plt.barh(x,y, color='pink', height=0.1)
plt.show()
```



[60]:
```
# Histogramme (!)

# ein Histogram ist ein Diagramm, das Häufigkeitsverteilungen zeigt.

# Es ist ein Diagramm,
# das die Anzahl der Beobachtungen innerhalb jedes
# gegebenen Intervalls zeigt.

x = [21,22,23,4,5,6,77,8,9,10,31,32,33,34,35,36,37,18,49,50,100]

plt.hist(x)
plt.show()

# hier wird gezeigt, dass im Interval [0,10] 6 Datensätze gibt
# im Interval [10,22] 4 Datensätze gibt,
# im Interval [23,31] 2 Datensätze gibt,
# in Imterval [32,37] 5 Datensätze gibt,...
```

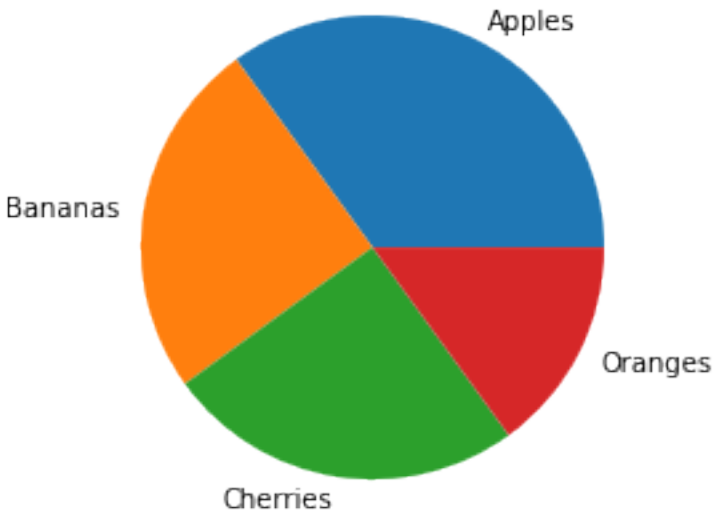


[82]:
```
# Anzahl Intervalle kann angepasst werden "num_bins"

num_bins = 5 # Anzahl intervalle

plt.hist(x,
         num_bins)
plt.show()

# hier wird gezeigt, dass im Interval [0,20] 10 Datensätze gibt
# im Interval [20,40] 7 Datensätze gibt, usw.
```

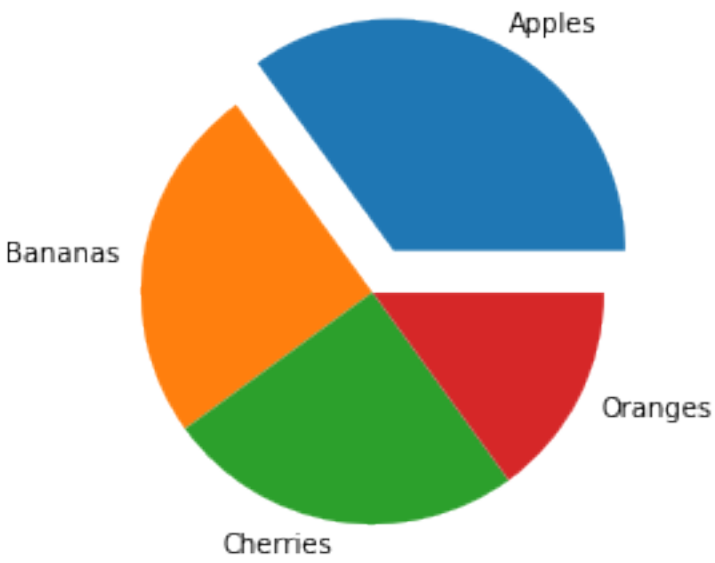


[64]:
```
# Kreisdiagramme

y = np.array([35, 25, 25, 15])

plt.pie(y)
plt.show()
```

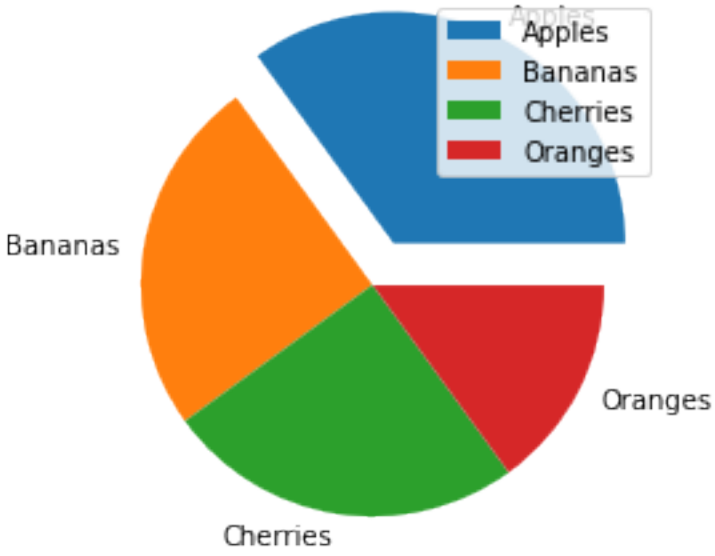
[69]:
```
# Labels von Jedem Datensatz einfügen

y = np.array([35, 25, 25, 15])
mylabels = ['Apples', 'Bananas', 'Cherries', 'Oranges']

plt.pie(y, labels=mylabels)
plt.show()
```



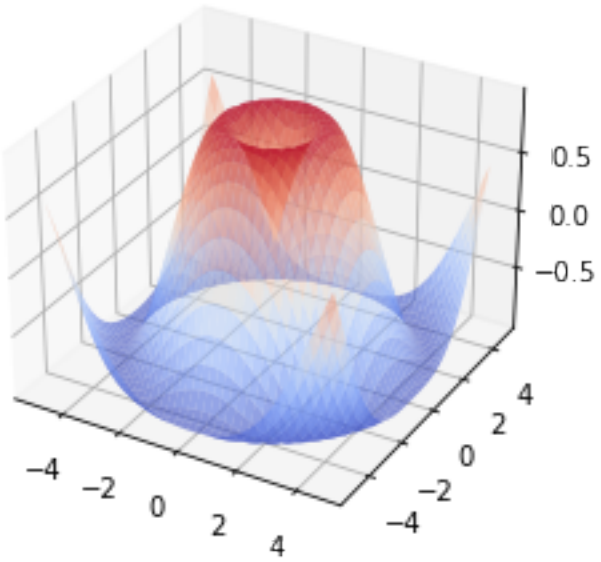
[70]:
```
# Vielleicht möchtet ihr beim Kunde Apfel eine Keile hervorheben...

y = np.array([35, 25, 25, 15])
mylabels = ['Apples', 'Bananas', 'Cherries', 'Oranges']
myexplode = [0.2, 0, 0, 0]

plt.pie(y, labels=mylabels, explode=myexplode)
plt.show()
```



[72]:
```
# Legende

y = np.array([35, 25, 25, 15])
mylabels = ['Apples', 'Bananas', 'Cherries', 'Oranges']
myexplode = [0.2, 0, 0, 0]

plt.pie(y, labels=mylabels, explode=myexplode)
plt.legend()
plt.show()
```

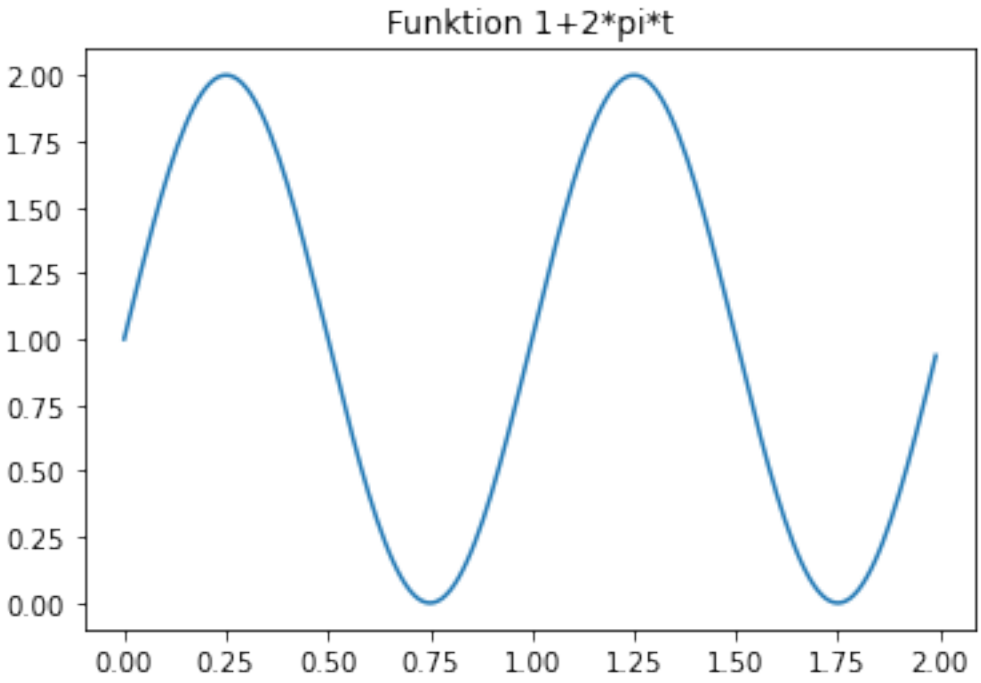


[76]:
```
# 3D Darstellung (nicht Prüfungsrelevant)

import matplotlib.pyplot as plt
from matplotlib import cm # color manager
import numpy as np
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})

# Data
X = np.arange(-5,5,0.25)
Y = np.arange(-5,5,0.25)
X,Y = np.meshgrid(X,Y)

R = np.sqrt(X**2 + Y**2)

Z = np.sin(R)
# Plot Surface
surf = ax.plot_surface(X,Y,Z,cmap=cm.coolwarm, alpha=0.6)
plt.show()
```



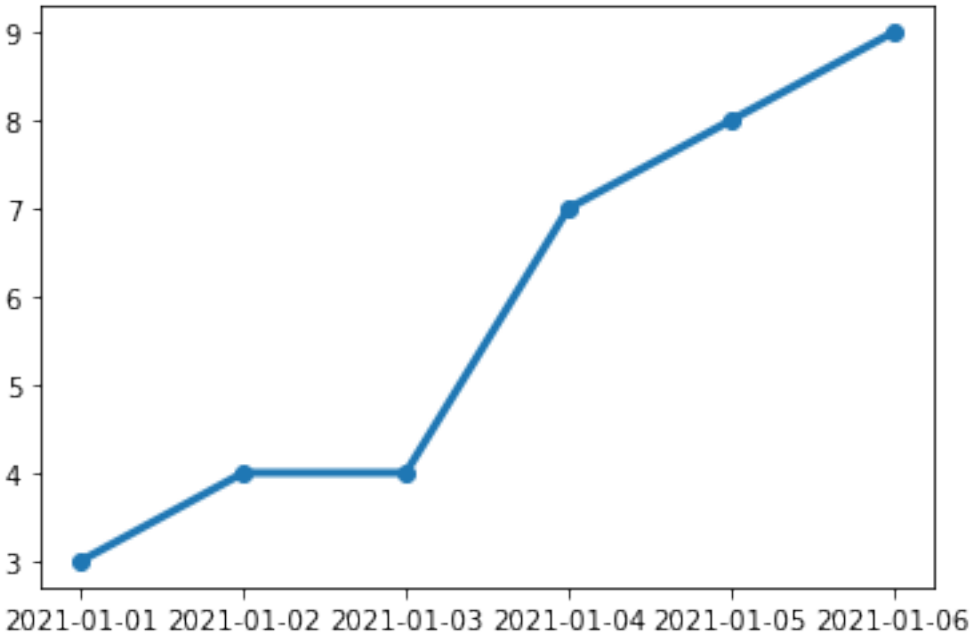
[77]:
```
# 2D Funktion

t = np.arange(0,2,0.01) # Daten zw 0 und 2 in Intervale von 0.01

s = 1 + np.sin(2*np.pi*t)
```

```
plt.title('Funktion 1+2*pi*t')
plt.plot(t,s)
plt.show()
```



[78]:
```
# Zeitreihen (nicht für die Prüfung)

import matplotlib.pyplot as plt
import datetime
import numpy as np
import pandas as pd
#define data
df = pd.DataFrame({'date': np.array([datetime.datetime(2021, 1, i+1) for i in␣
 ↪range(6)]),
                   'sales': [3, 4, 4, 7, 8, 9]})

#plot time series
plt.plot(df.date, df.sales, linewidth=3, marker='o')
```

[78]:
```
[<matplotlib.lines.Line2D at 0x7f93cde66730>]
```


9
8
7
6
5
4
3
2021-01-01
2021-01-02
2021-01-03
2021-01-04
2021-01-05
2021-01-06

[ ]: