20211111_Informationmanagement_FAT1

November 11, 2021

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

# MATPLOTLIB.PYPLOT as PLT
```

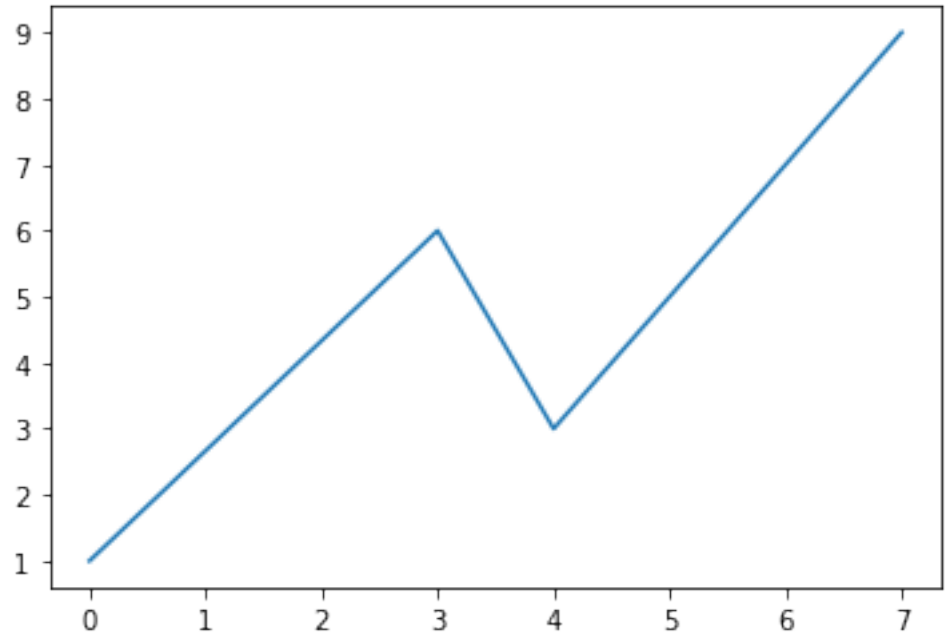
[2]:
```
!pip install matplotlib
```

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

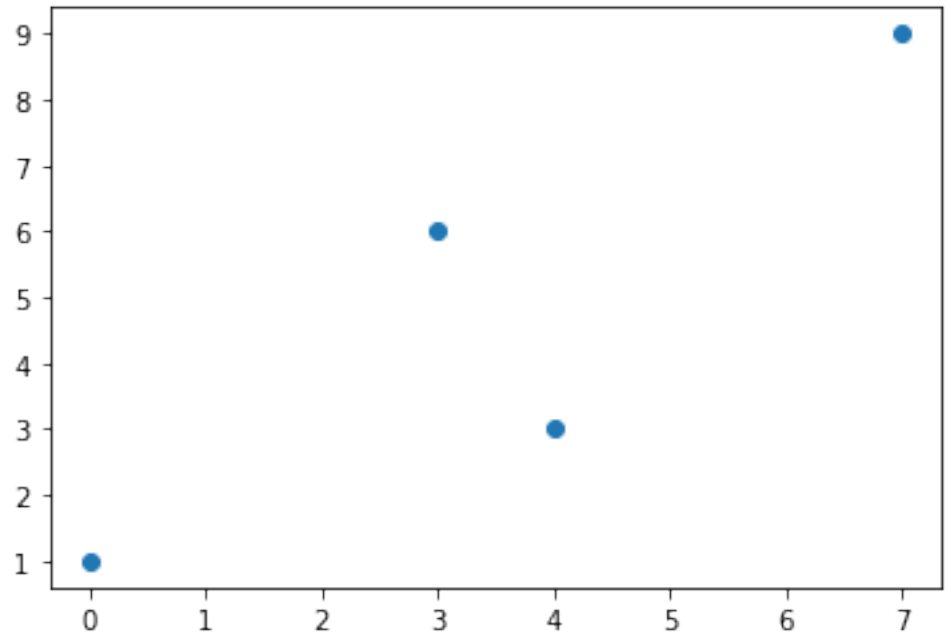
[3]:
```
import matplotlib.pyplot as plt
```

[4]:
```
import numpy as np
```

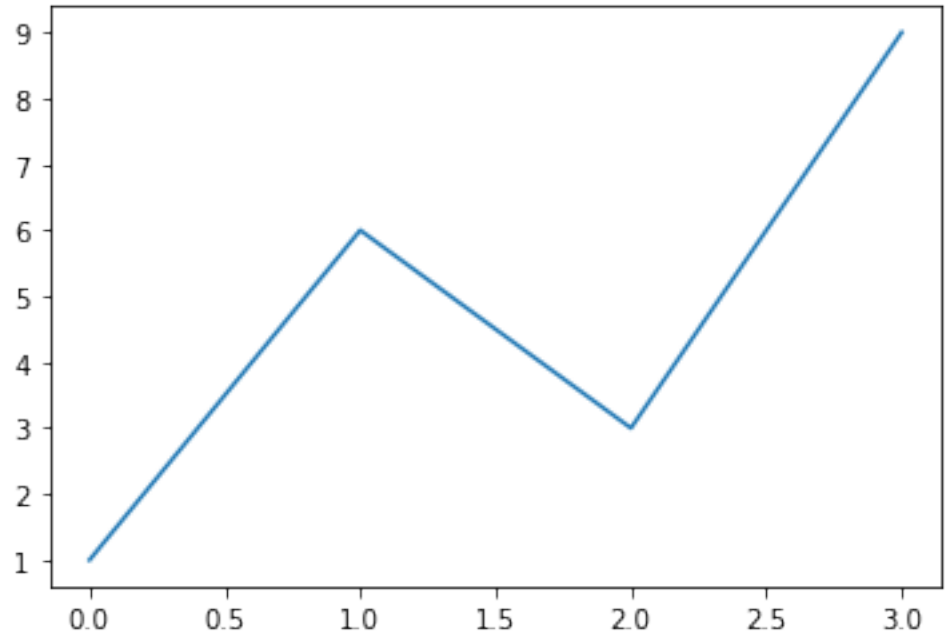
[5]:
```
# Beispiel einer Linie

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

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



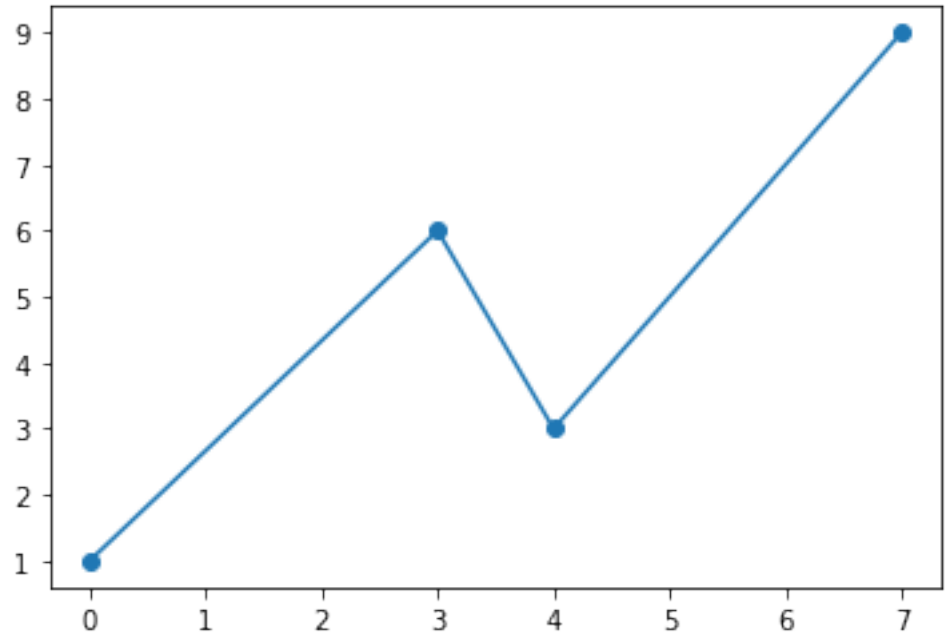
[6]:
```
# ohne line...

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



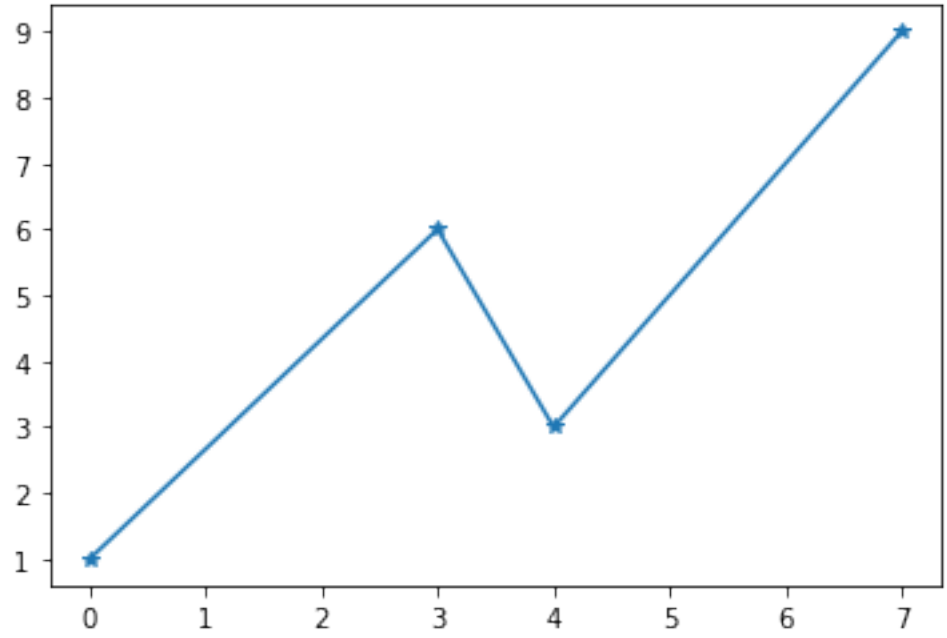
[7]:
```
# default x--points [0,1,2,...]

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



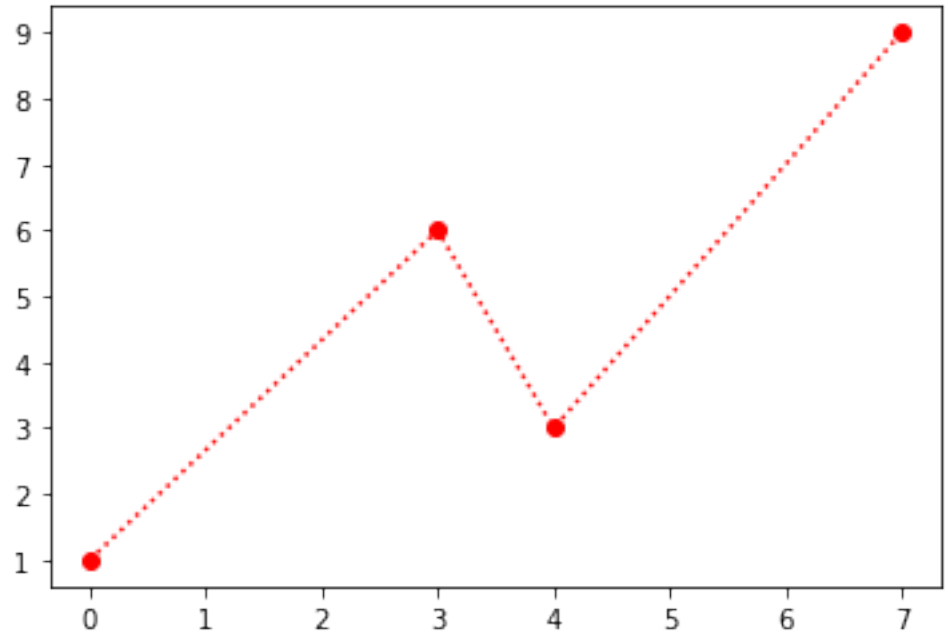
[8]:
```
# Punkte mit einen "o" darstellen und eine Linie

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



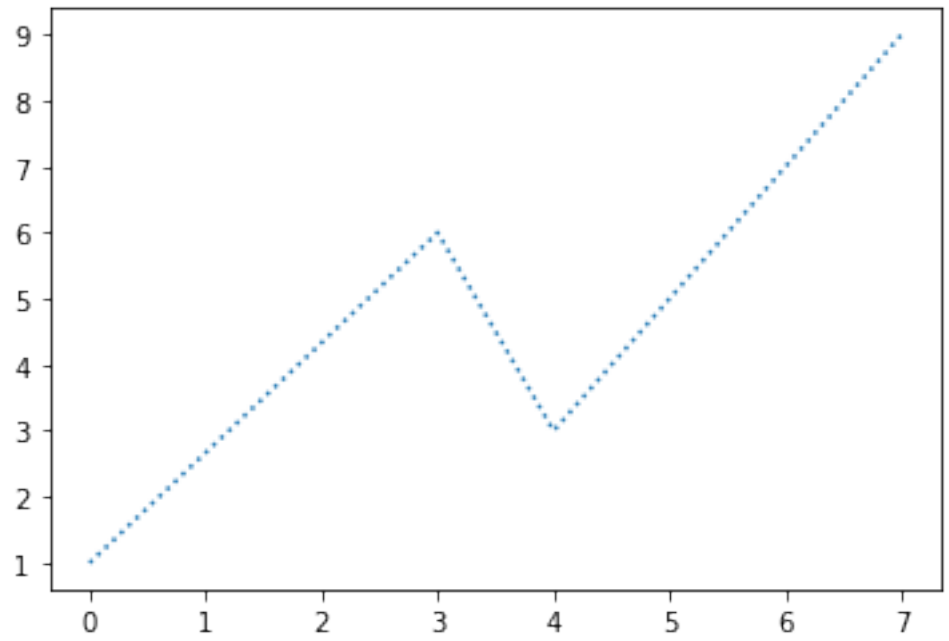
[9]:
```
# Punkte mit einen "*" darstellen und eine Linie

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



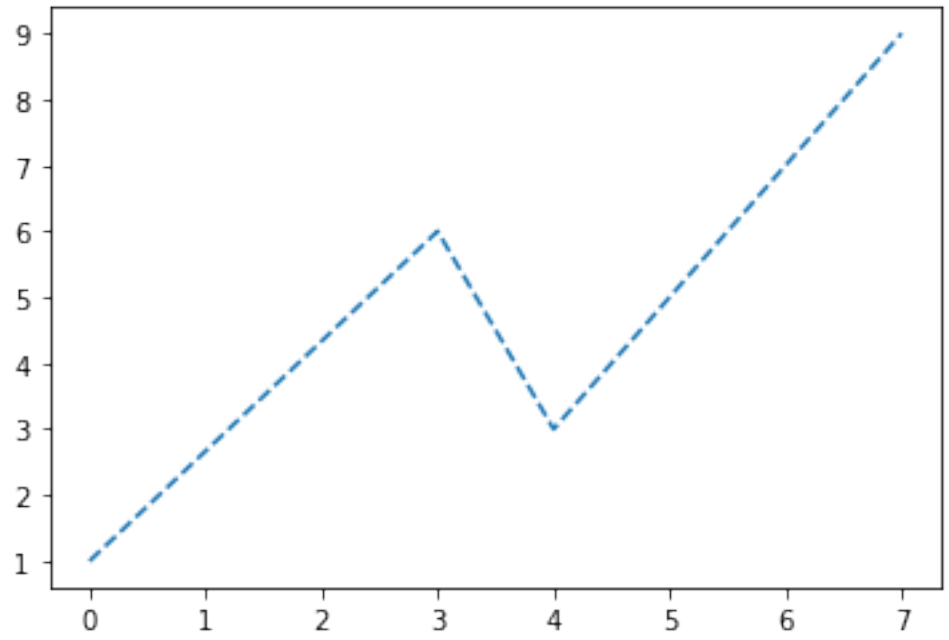
[13]:
```
# die Farbe kann auch angepasst werden

# Punkte mit einen "*" darstellen und eine rote Linie

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



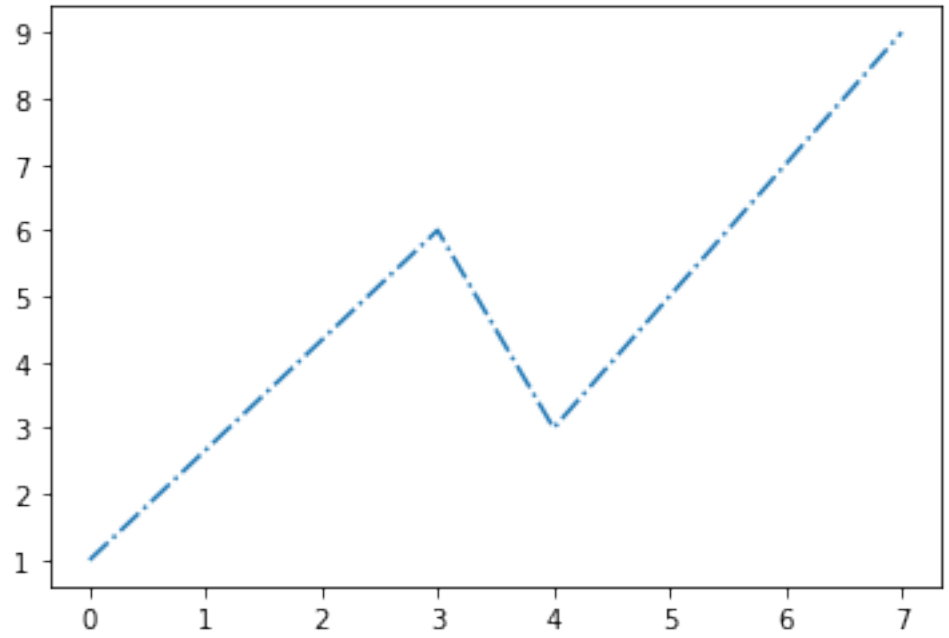
[14]:
```
# linien Styl kann geändert werden

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



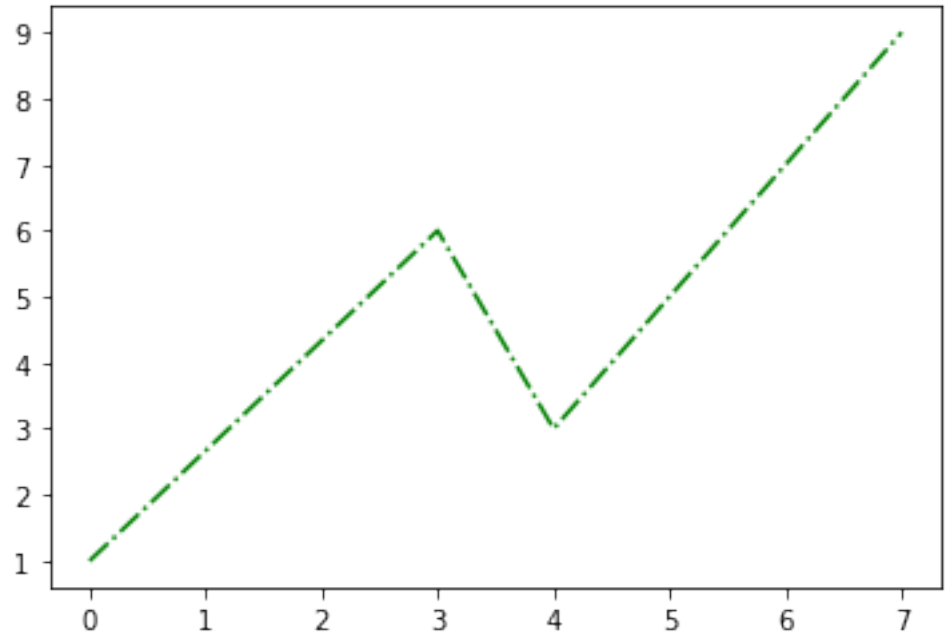
[15]:
```
plt.plot(xpoints, ypoints, linestyle='dashed')
plt.show()
```



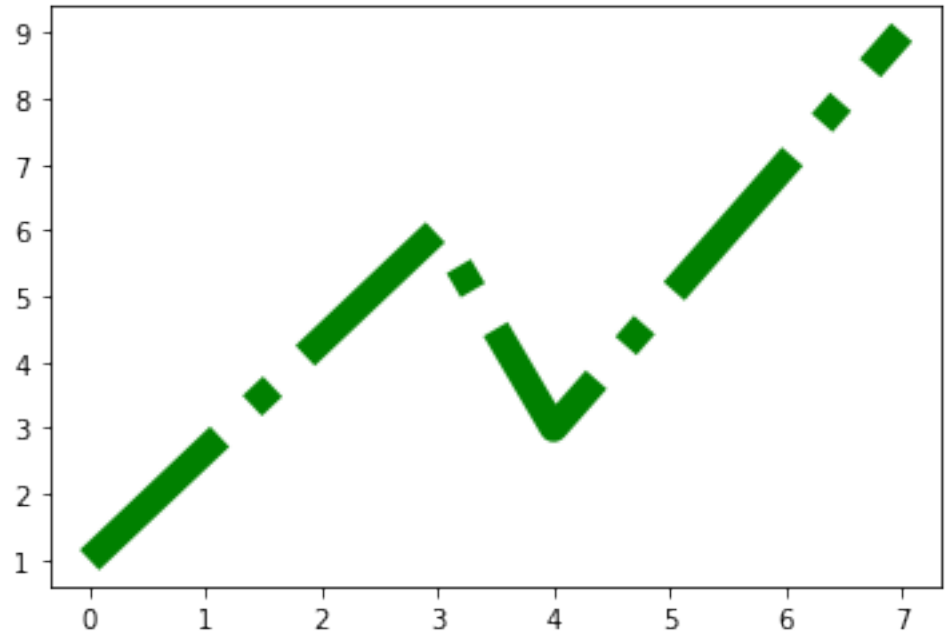
[16]:
```
plt.plot(xpoints, ypoints, linestyle='dashdot')
plt.show()
```



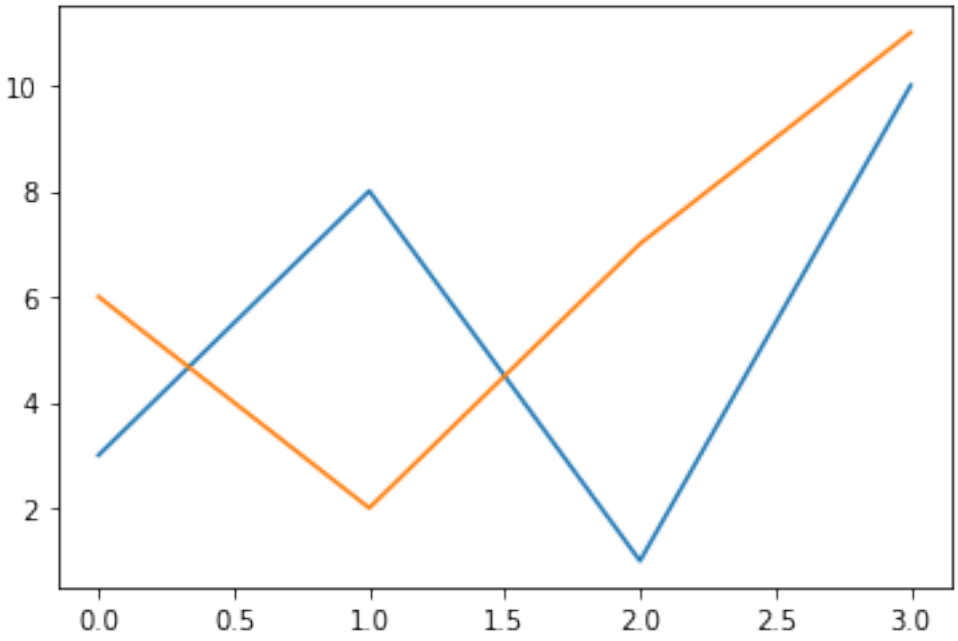
[17]:
```
# linien Farbe kann angepasst werden

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



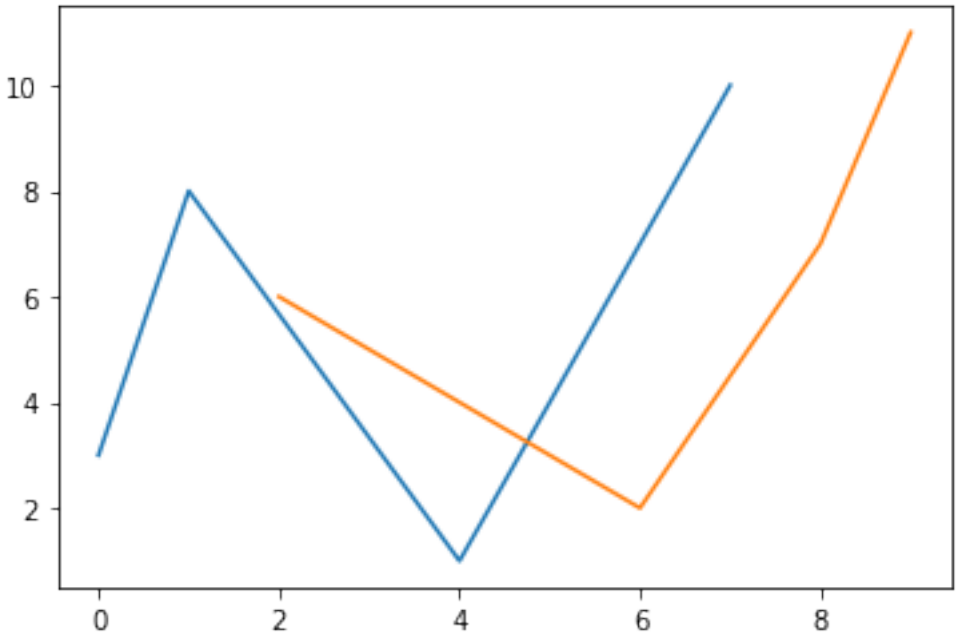
[18]:
```
# linien Dicke kann angepasst werden

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



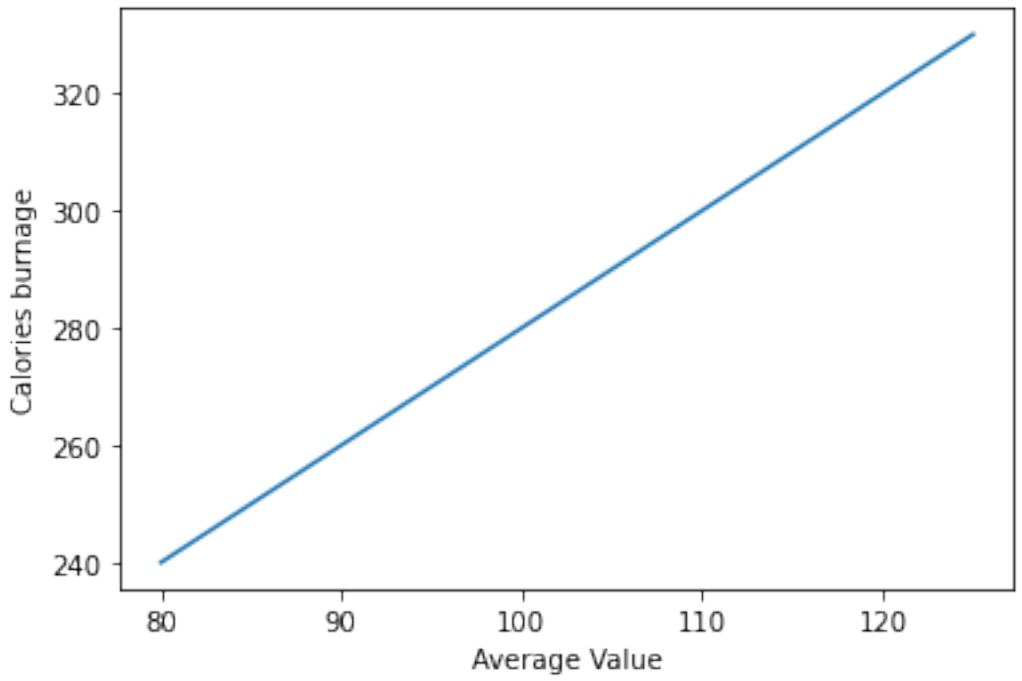
[19]:
```
# Mehrere Linien in einem Plot können dargestellt werden

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

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

plt.show()
```



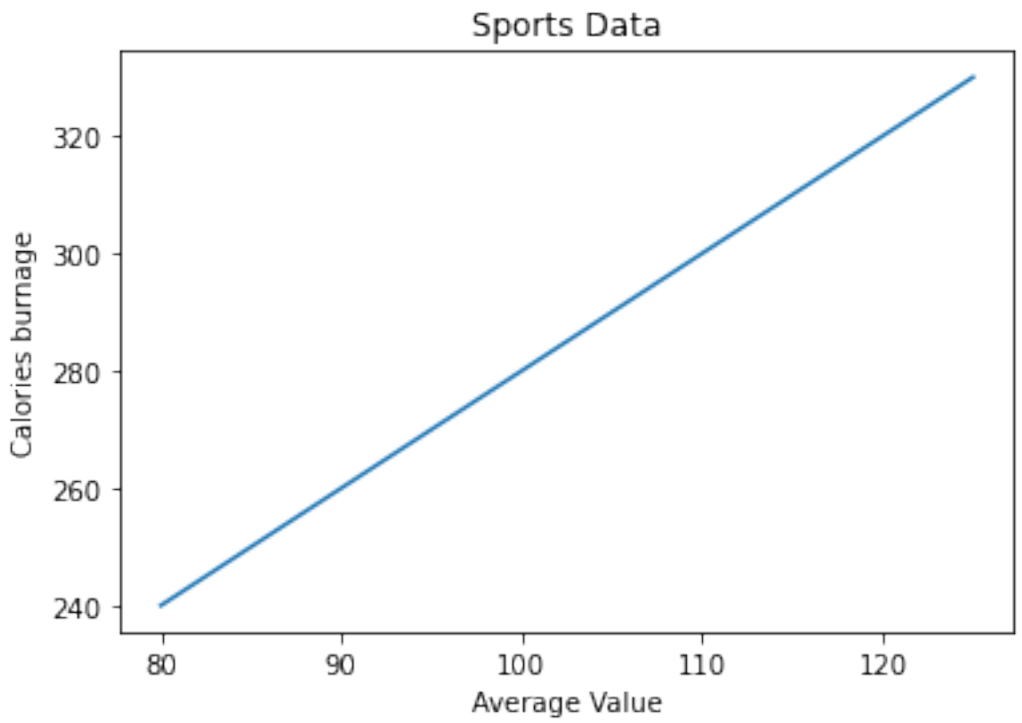
[20]:
```
# x Daten können belibig dargestellt werden

x1 = np.array([0,1,4,7])
x2 = np.array([2,6,8,9])

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

plt.plot(x1,y1,x2,y2)

plt.show()
```



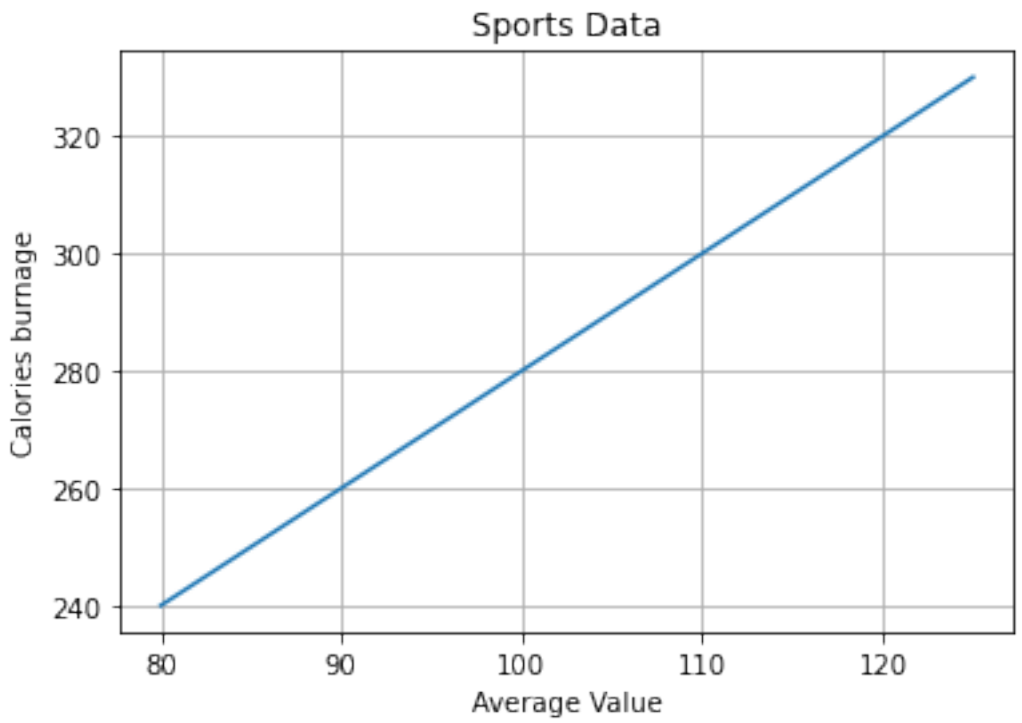
[22]:
```
# Labels vom Plot darstellen lassen

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x,y)

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

plt.show()
```



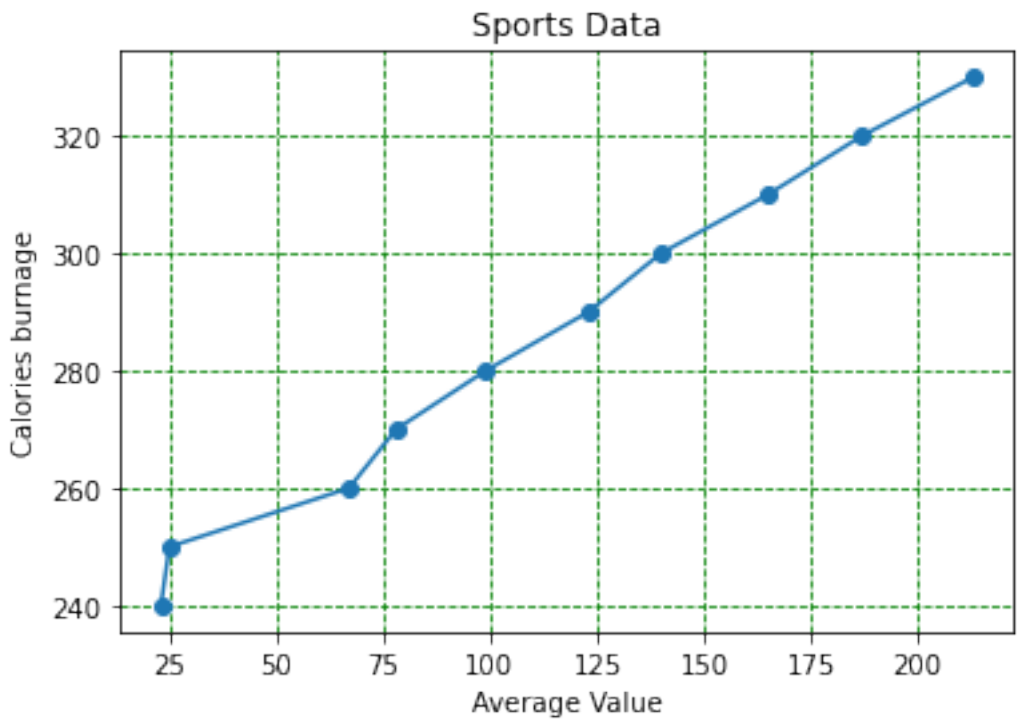
[23]:
```
# Titel Hinfügen

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x,y)

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

plt.title('Sports Data')

plt.show()
```



[24]:
```
# Grid (Hilfelininen hinfügen)

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x,y)

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

plt.title('Sports Data')

plt.grid()

plt.show()
```

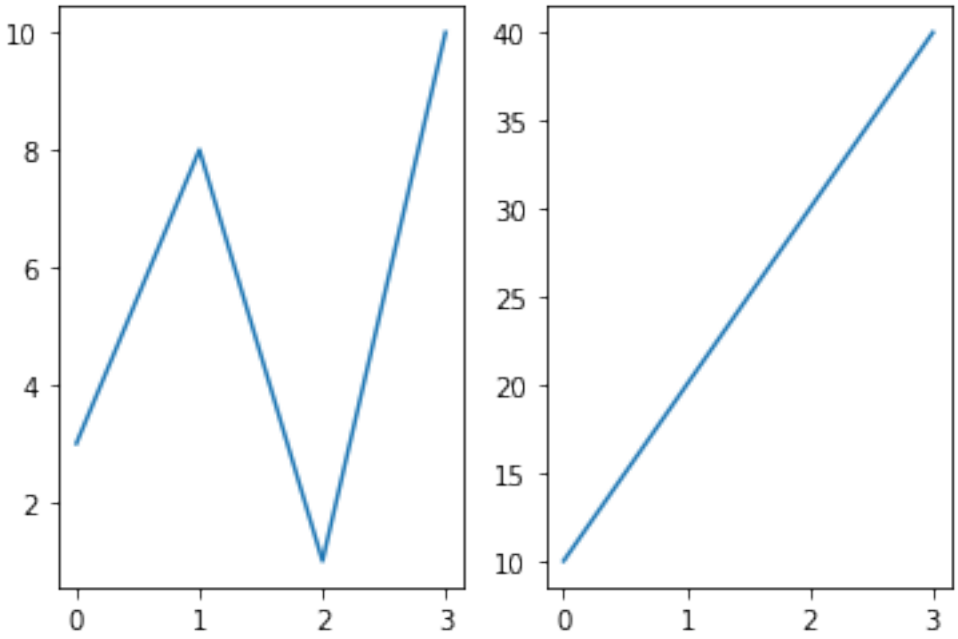


[27]:
```
# Merkmale vom Grid können angepasst werden

# Grid (Hilfelininen hinfügen)

x = np.array([23, 25, 67, 78, 99, 123, 140, 165, 187, 213])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

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

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

plt.title('Sports Data')

plt.grid(color='green', linestyle='dashed')

plt.show()
```

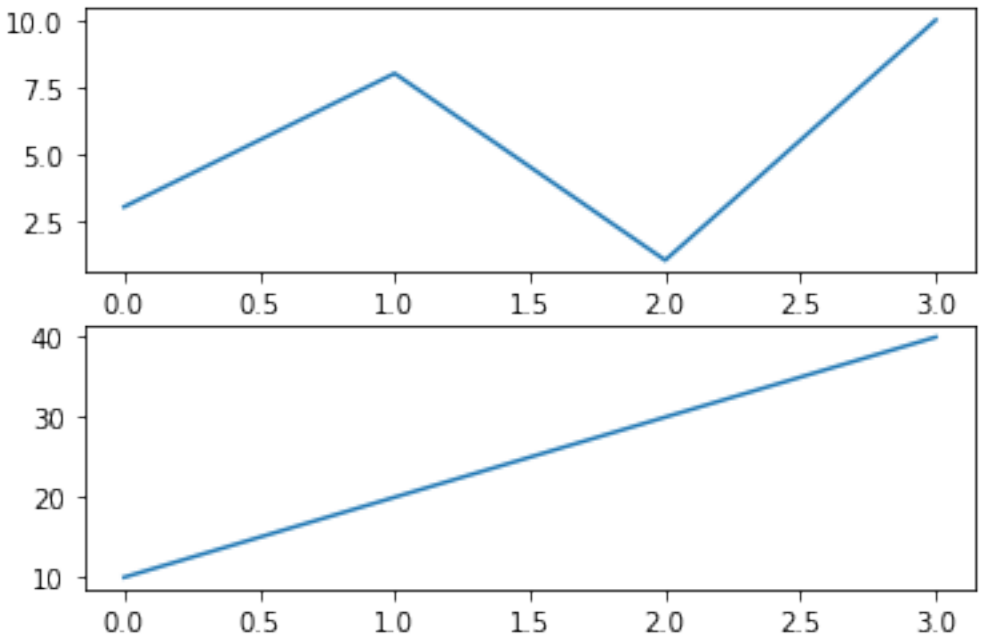


[29]:
```
# Mit der Subplots() Funktion können wir mehrere Plots in einem darstellen

#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)

#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.show()

# Die Subplots funktion benötigt drei argumente, die das Layout der Abbildung␣
 ↪beschreiben.
```

```
# Das Layout ist in Zeilen und spalten organisiert, die durch das erste und␣
 ↪zweite Argument repräsentiert werden.
# Das dritte Argument repräsentiert das Index des aktuellen Plots
```

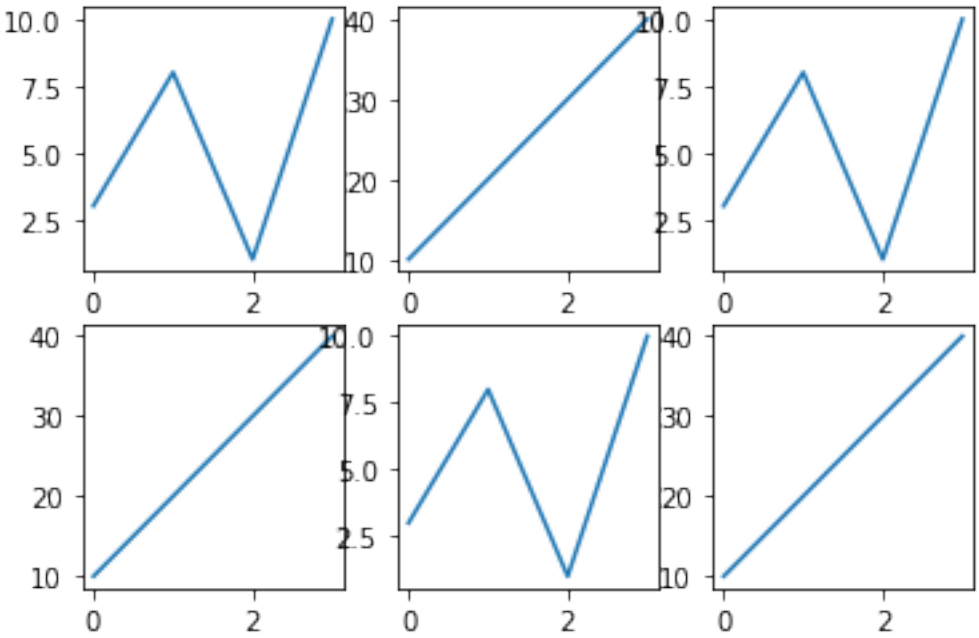


[30]:
```
#plot 1
x = np.array([0,1,2,3])
y = np.array([3,8,1,10])

plt.subplot(2,1,1) # es ist ein Plot mit 2 Zeilen, 1 Spalte. Subplot in 1.␣
 ↪Position.
plt.plot(x,y)

#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)

plt.show()
```



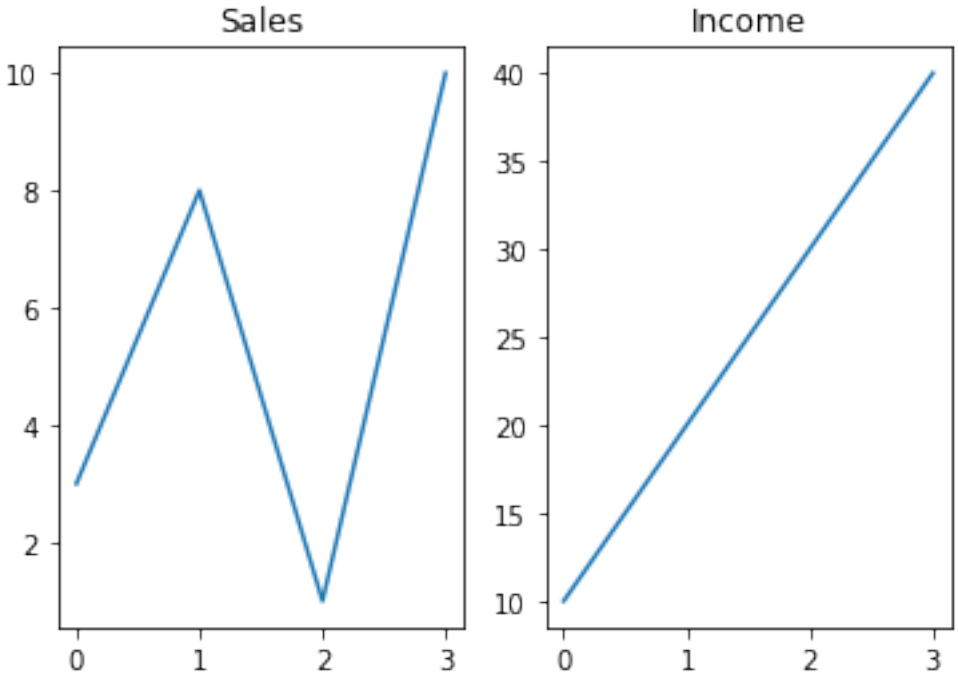
[31]:
```
# Beliebig viele Plots auf eine Figur zeichnen

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()
```



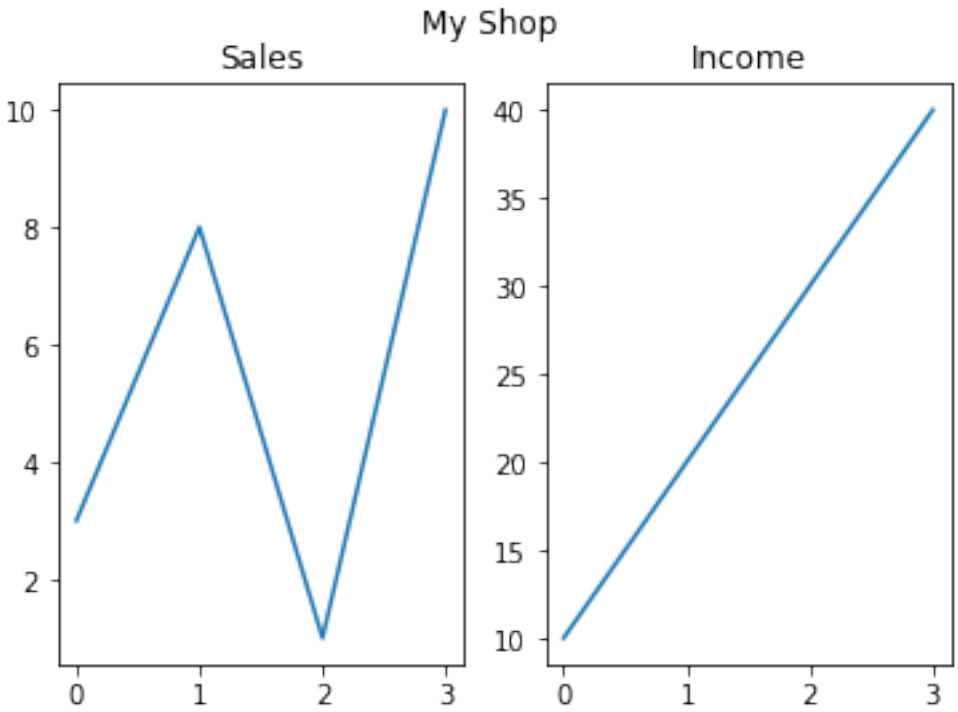
[32]:
```
# Bei Jedem Plot ein Titel einfügen

#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()
```



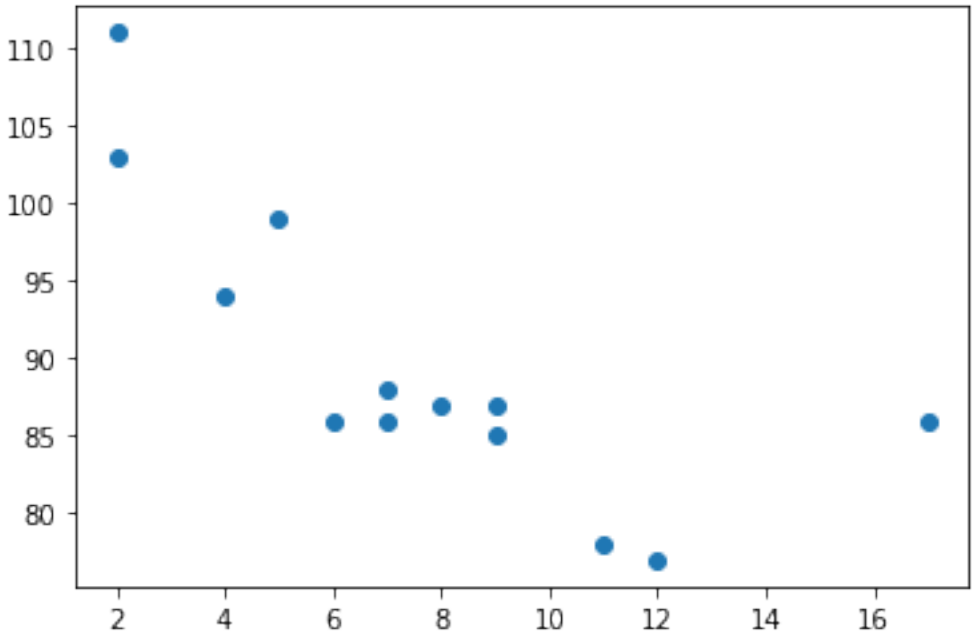
[33]:
```
# titel für die ganze Abbildung

#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()
```

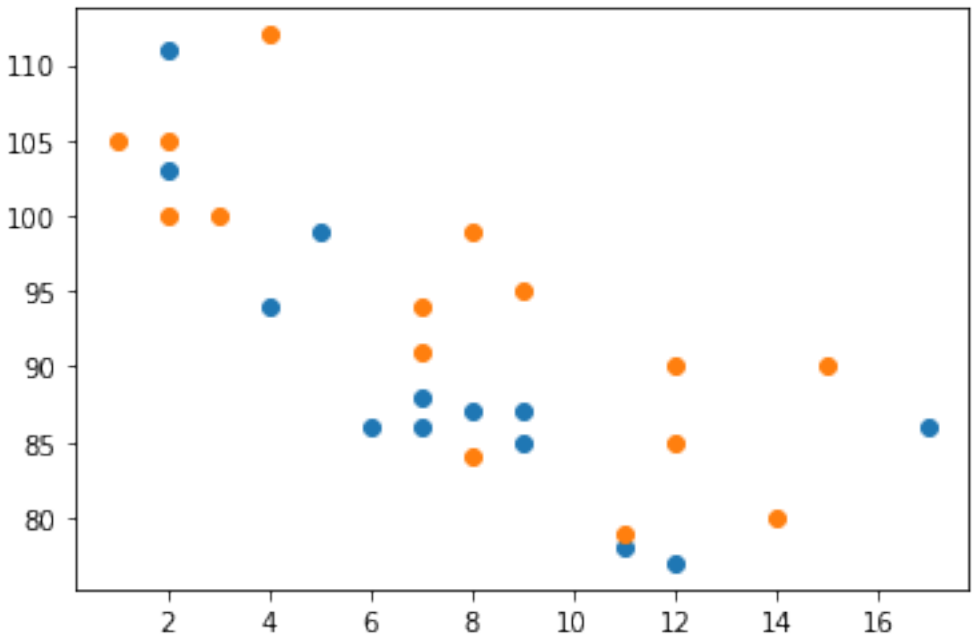


[34]:
```
# Erstellung von Streudiagrammen (Scatterplot)

# Die Scatter Funktion zeichnet für Jede Beobachtung einen Punkt.
# es benötigt gleich lange Arrays
```

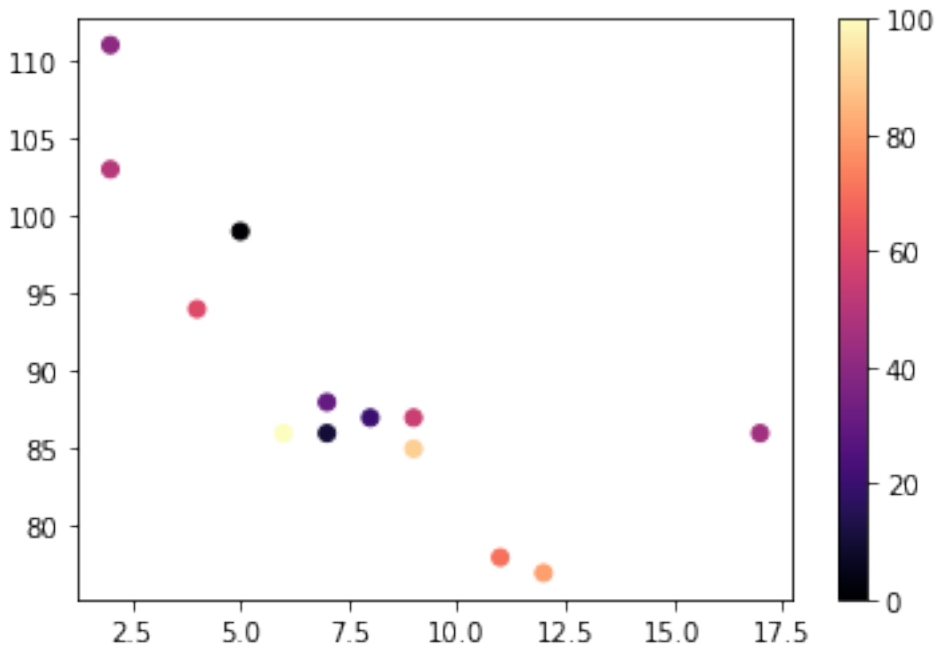
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

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



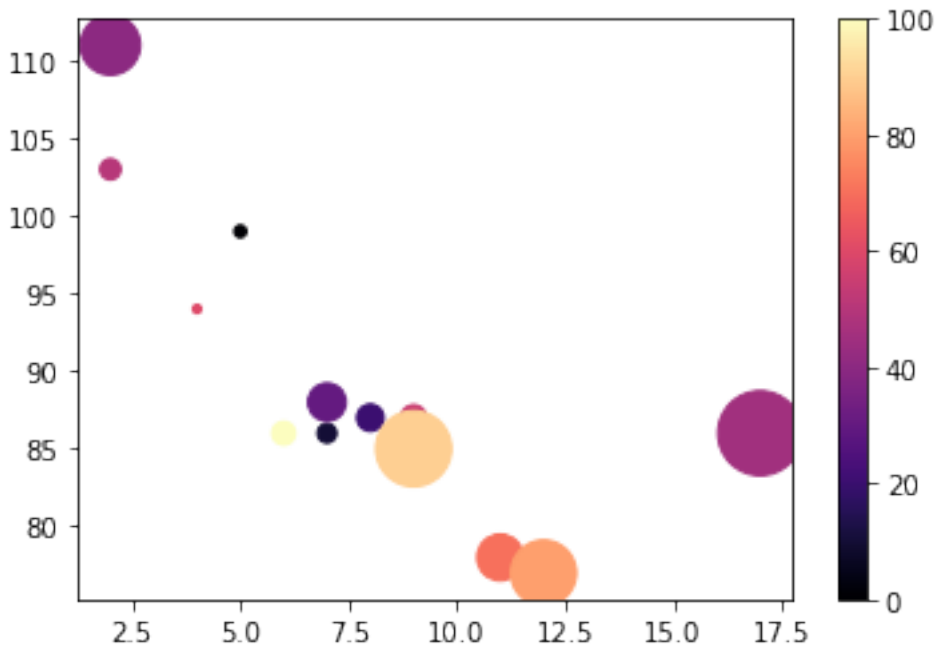
[35]:
```
# Zwei Arten von Punkten in einer graphik darstellen

# 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()
```



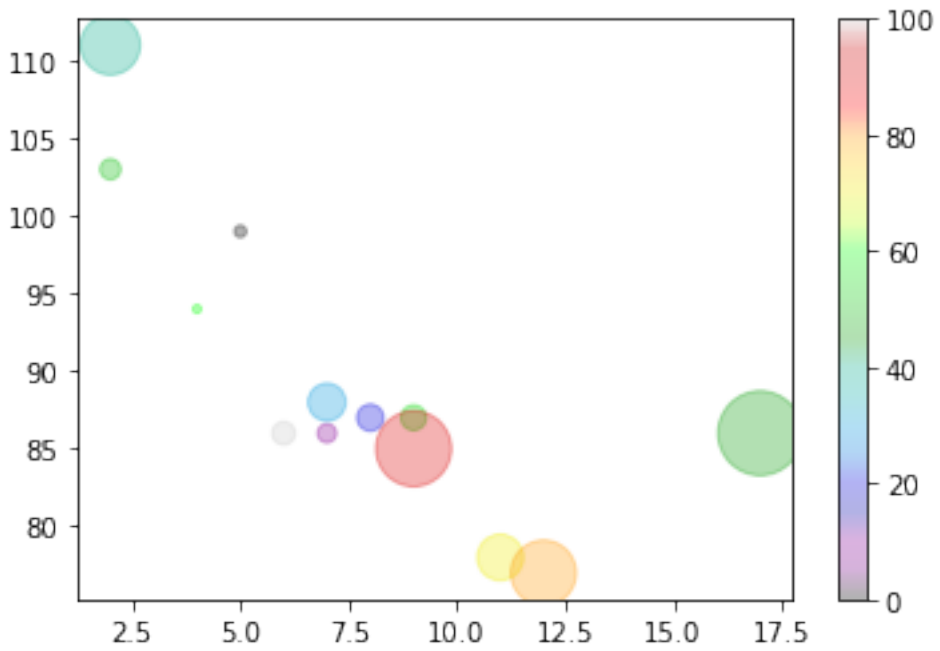
[40]:
```
# Color Map (Legende einfügen)

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()
```



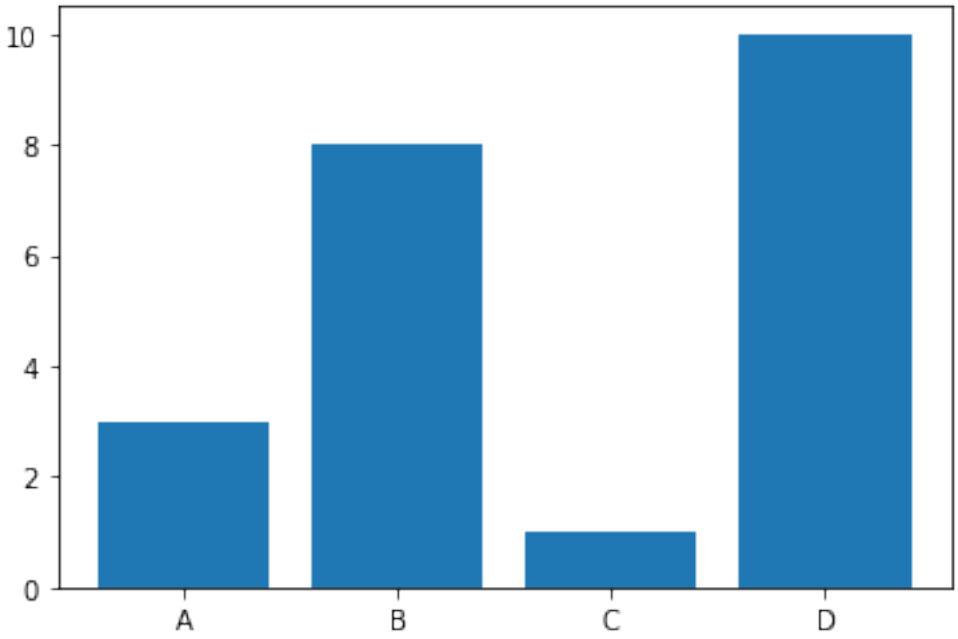
[41]:
```
# grösse von den Punkten selber bestimmen

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()
```



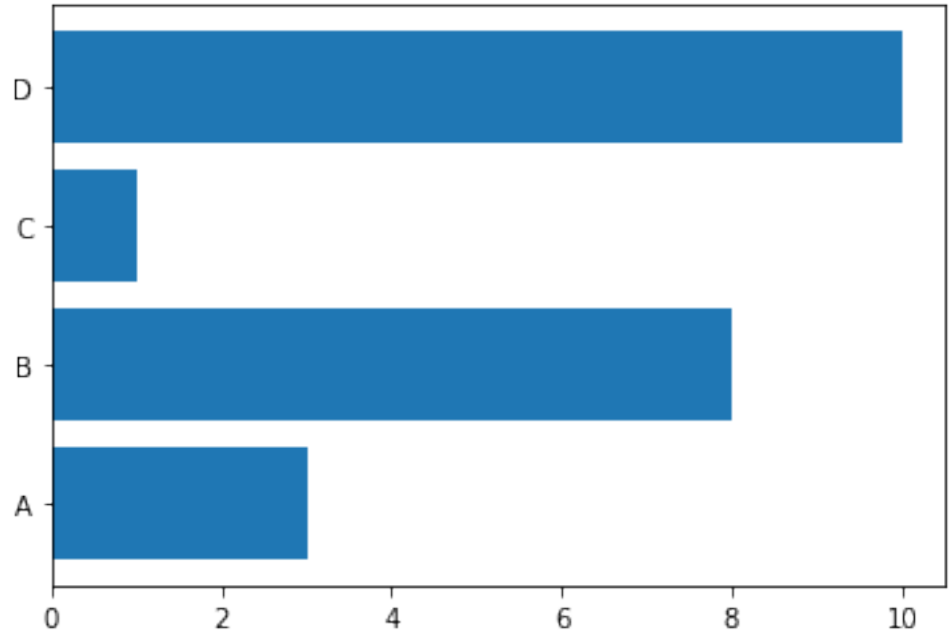
[43]:
```
# Transparenz einfügen "alpha"

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='nipy_spectral', alpha = 0.3)

plt.colorbar()

plt.show()
```



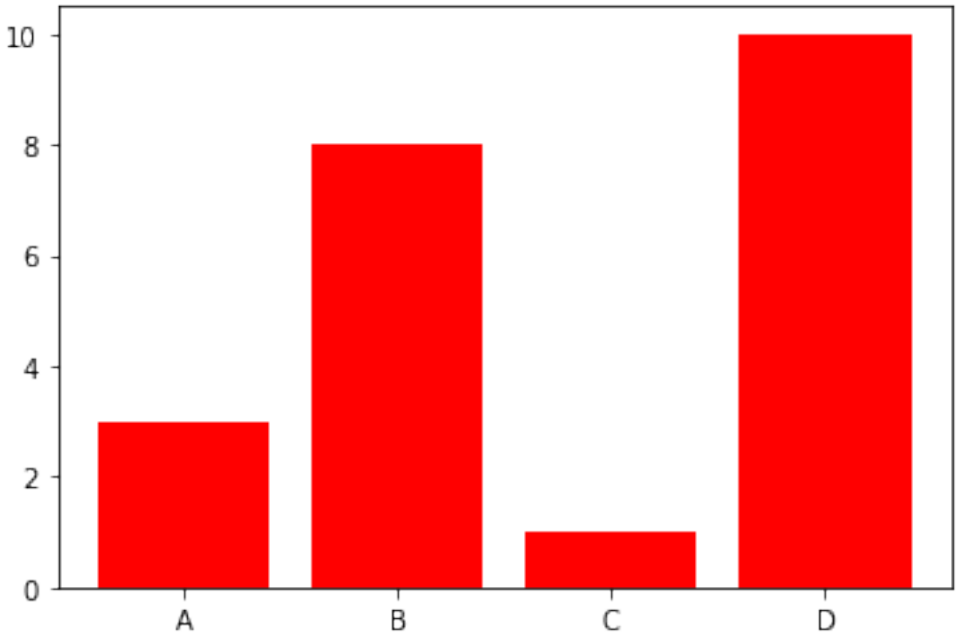
[44]:
```
# Balkendiagramme können mit "bar()" dargestellt werden

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

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



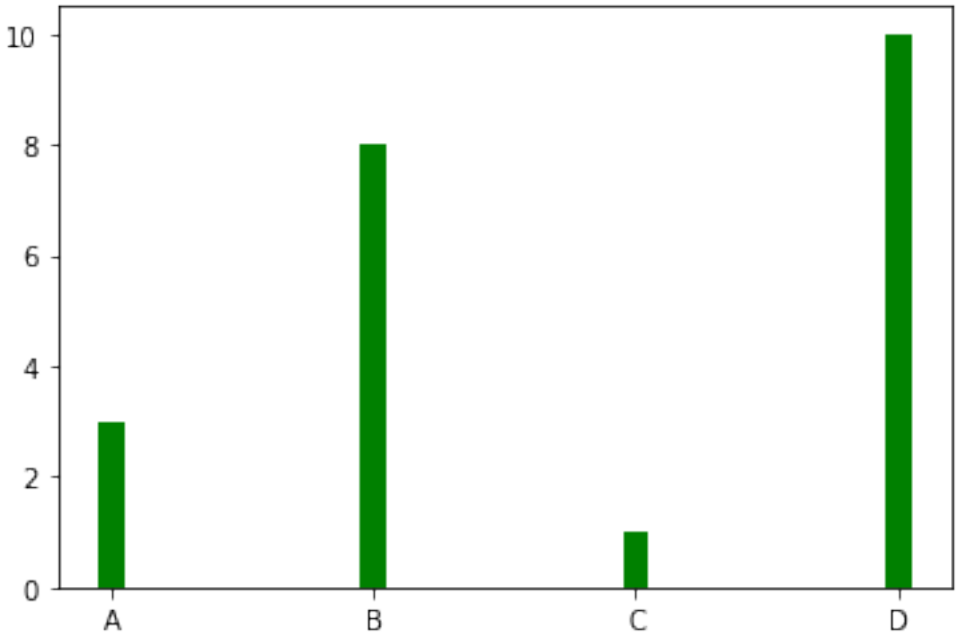
[45]:
```
# quer Balkendiagramme darstellen

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



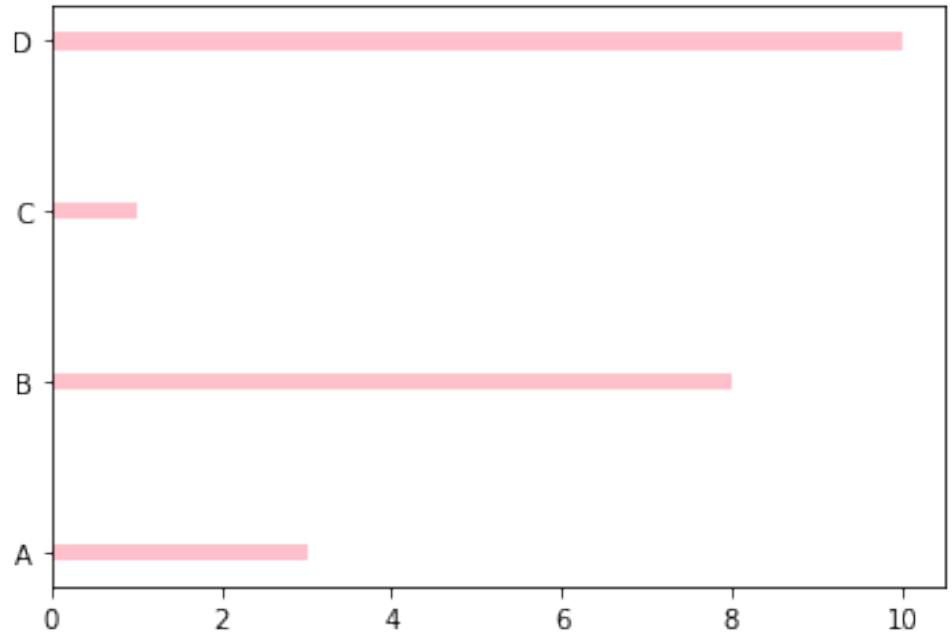
[46]:
```
# die Farbe kann angepasst werden

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

plt.bar(x,y, color='red')
plt.show()
```



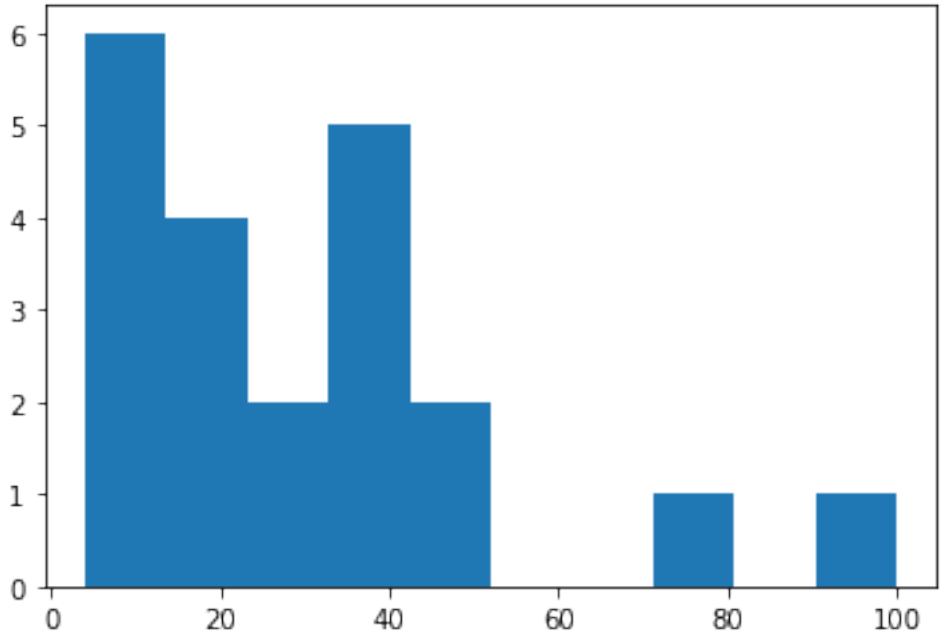
[47]:
```
# Balkenbreite kann angepasst werden

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()
```



[49]:
```
# 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()
```

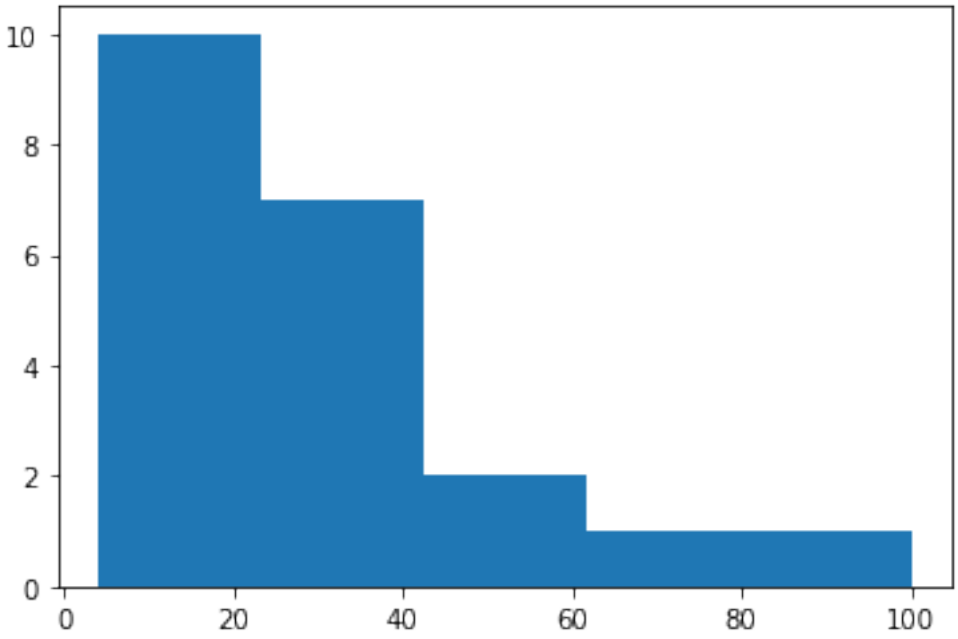


[53]:
```
# Histogramme (!)

# Ein Histogramm 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 [10,20] 4 datensätze gibt, usw.
```



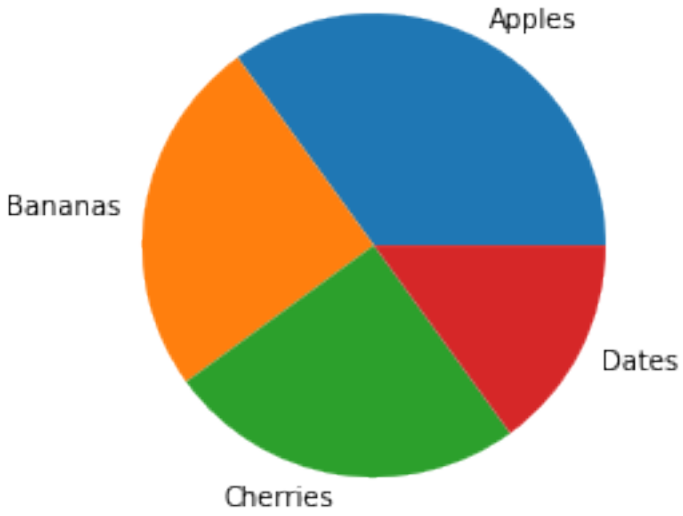
[52]:
```
# anzahl Intervalle kann angepasst werden "num_bins"

num_bins = 5

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.
```

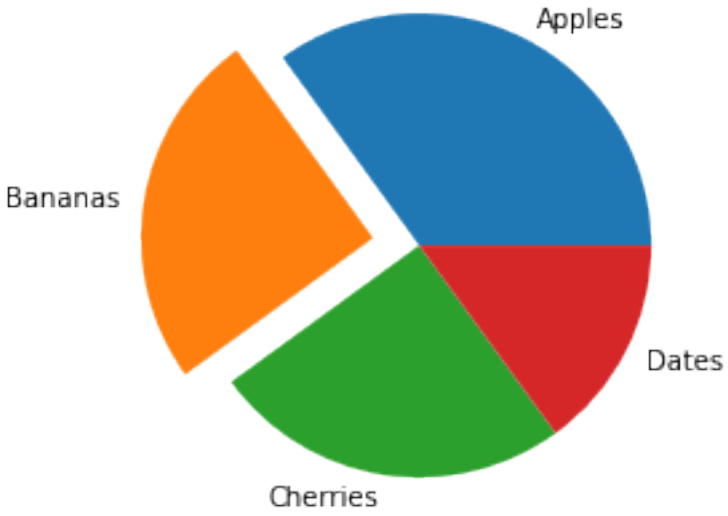


[54]:
```
# Kreisdiagramme erstelle

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

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

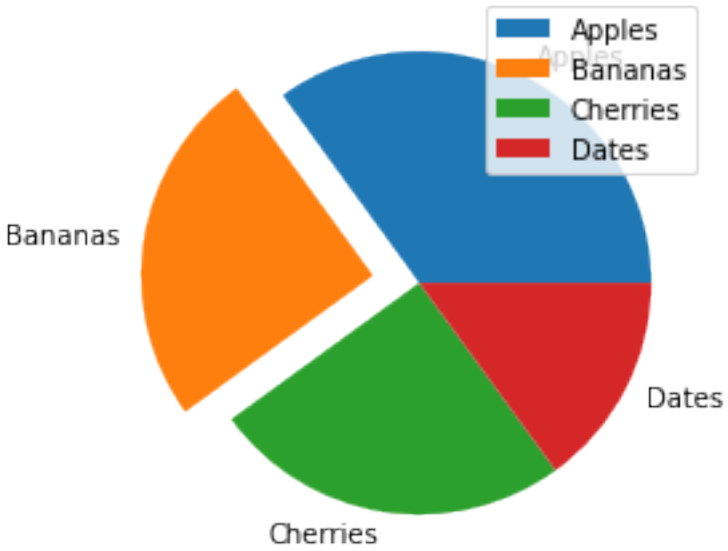
[55]:
```
# Labels auf den Daten

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

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



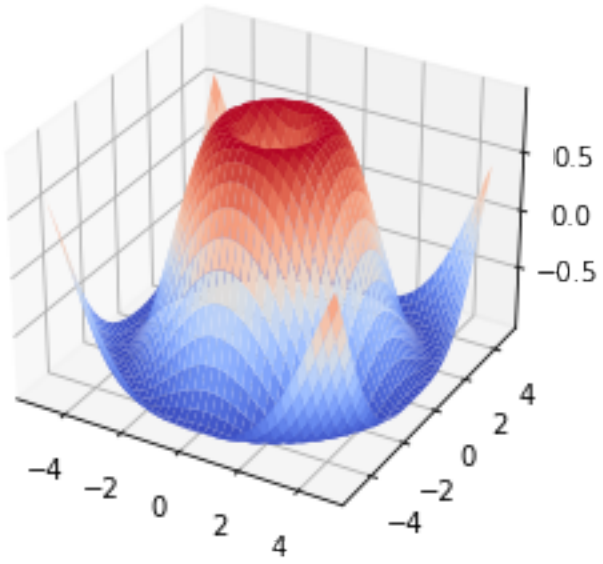
[56]:
```
# Vielleicht möchtet ihr, dass einer der Keile auffällt?
# Der "explode" Parameter ermöglicht sowas

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

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



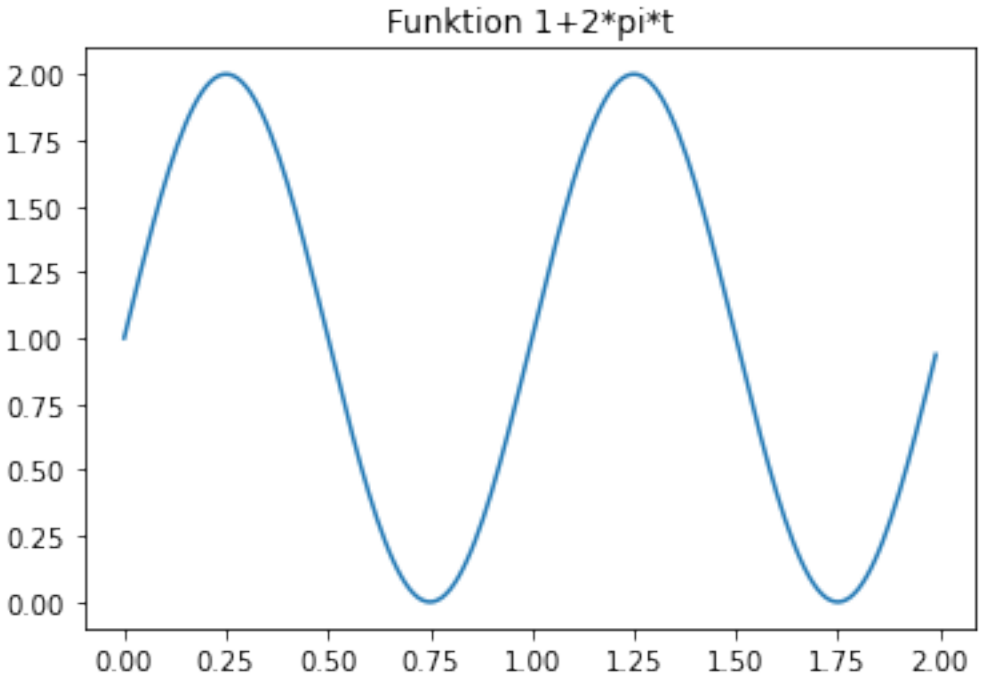
[57]:
```
# Legende

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

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



[60]:
```
# 3 dimensionen

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)

plt.show()
```



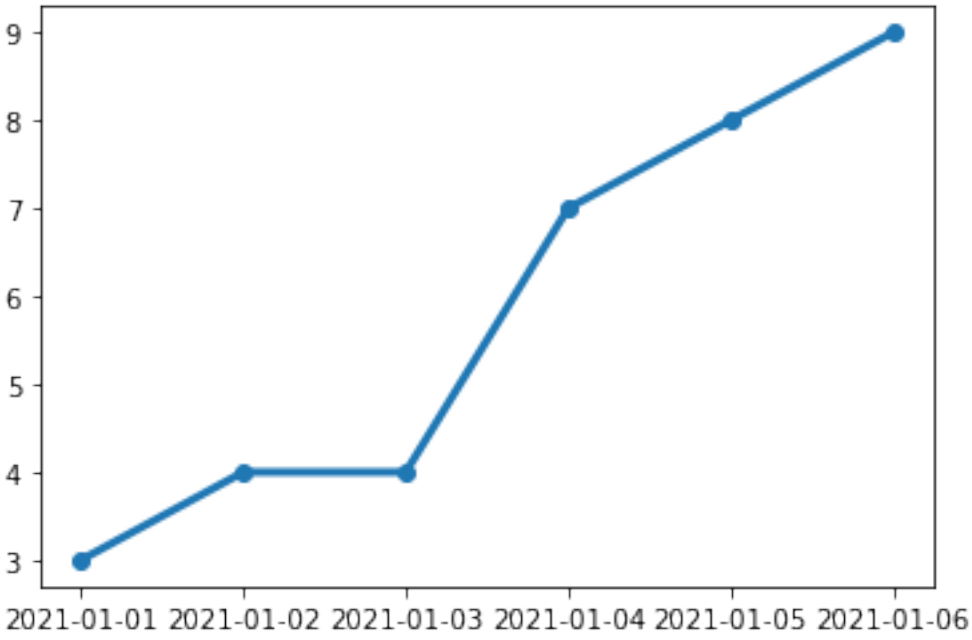
[67]:
```
# 2D Darstellung von einer Funktion

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()
```



[71]:
```
# Time series

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')
```

[71]: [<matplotlib.lines.Line2D at 0x7fe4807b1790>]



[ ]: