Lineare Gleichungssysteme (1)

- Ziel ist es, das folgende lineare Gleichungssystem zu lösen:

```
a * x1**3 + b * x1**2 + c * x1 = y1
a * x2**3 + b * x2**2 + c * x2 = y2
a * x3**3 + b * x3**2 + c * x3 = y3
```

  a, b, c sind die Unbekannten.

- Dazu stellen wir die Koeffizientenmatrix x und den Konstantenvektor y auf:

$$x = \begin{bmatrix} x1**3 & x1**2 & x1 \\ x2**3 & x2**2 & x2 \\ x3**3 & x3**2 & x3 \end{bmatrix} \qquad y = \begin{bmatrix} y1 \\ y2 \\ y3 \end{bmatrix}$$

Lineare Gleichungssysteme (2)

- In numpy gibt es dafür eine Funktion **array**, die ein Array zurückgibt:

```
x = np.array([[ x1**3, x1**2, x1 ],
              [ x2**3, x2**2, x2 ],
              [ x3**3, x3**2, x3 ]])

y = np.array([y1, y2, y3])
```

  Beachten Sie bei x die doppelten eckigen Klammern, da es sich um ein zweidimensionales Array handelt.

- Man kann die Matrixelemente auch einfach der Reihe nach aufzählen und sie mit Hilfe der Funktion **reshape** anordnen lassen. Dadurch spart man sich das Eintippen von eckigen Klammern:

```
x = np.array([x1**3, x1**2, x1, x2**3, x2**2, x2, x3**3, x3**2, x3]).reshape(3, 3)
```

Lineare Gleichungssysteme (3)

- y ein Zeilenvektor. Das geht auch als Matrix mit einer Zeile und drei Spalten:

```
y = np.array([[y1, y2, y3]])
```

- Wenn man einen Spaltenvektor möchte, muss man eine Matrix mit drei Zeilen und einer Spalte anlegen:

```
y = np.array([[y1], [y2], [y3]])
```

- Oder auch mit Hilfe der Funktion **T** zu Transponierung:

```
y = np.array([[y1, y2, y3]]).T
```

  Beachten Sie, dass nur Matrizen transponiert werden können.

- Zur Überprüfung sollten Sie print-Ausgaben einfügen:

```
print("coefficients:")
print(x)
print("constants:")
print(y)
```