Institute of Mathematics and Image Computing, Lübeck

MA4030: Optimierung, Sommersemester 2025

Übungsblatt 1

Abgabe von: *Fynn-Ole Claussen, 770712 und Johann Fischer, 779072 und Mika Kohlhammer, 779098*

A1\code\P1.py

```python
## TODO: Aufgabenteil 1a. Alle Zahlen zwischen 1 und 10 ausgeben
print('### 1a ###')
# BEGIN SOLUTION
for i in range(10):
    print(i+1)
# END SOLUTION


## TODO: Aufgabenteil 1b. Alle geraden Zahlen zwischen 1 und 10 ausgeben, auf 2 Arten
print('\n### 1b ###')
# BEGIN SOLUTION
for i in range(1,11):
    if i%2==0:
        print(i)
for i in range(2,11,2):
    print(i)
# END SOLUTION


## TODO: Aufgabenteil 1c. Funktion, die alle geraden Zahlen zwischen 1 und n ausgibt
def alle_gerade_zahlen(n):
    # BEGIN SOLUTION
    for i in range(2,n+1,2):
        print(i)
    # END SOLUTION

# Funktion testen
print('\n### 1d ###')
alle_gerade_zahlen(10)
alle_gerade_zahlen(4)
alle_gerade_zahlen(0)

#TERMINAL OUTPUT:
```

1a

1

2

3

4

5

6

7

8

9

10

1b

2

4

6

8

10

2

4

6

8

10

1d

2

4

6

8

10

2

4

```
#PLOTS:
```

A1\code\P2.py

```
## TODO: Aufgabenteil 2a. Liste mit geraden Zahlen zwischen 1 und 10
print('### 2a ###')
# BEGIN SOLUTION
mylist=[2,4,6,8,10]
# END SOLUTION


## TODO: Aufgabenteil 2b. Werte ausgeben
print('\n### 2b ###')
# BEGIN SOLUTION
print(mylist[0], mylist[2], mylist[-1], mylist[1:3], mylist[1:], mylist[:4],mylist[:-2])
print("Die Indizierung  beginnt bei 0. Bei negativer Indizierung wird die Liste von hinten durchgezÃ¤hlt.
# END SOLUTION


## TODO: Aufgabenteil 2c. Letzten Eintrag Ã¤ndern
print('\n### 2c ###')
# BEGIN SOLUTION
mylist[-1]=100
print(mylist)
# END SOLUTION


## TODO: Aufgabenteil 2d. Numpy Array mit allen geraden Zahlen von 1 bis 10 auf 2 Arten erstellen
print('\n### 2d ###')
# Hinweis: Importieren Sie das Paket numpy ausnahmsweise hier
import numpy as np
# BEGIN SOLUTION
mylist[-1]=10 #Modifizieren das letzte Element erneut. Nach Aufgabe ist myList wieder eine Liste mit Hoech
array1 = np.array(mylist)
array2 = np.arange(2,11,2)
# END SOLUTION
print(array1,array2)

## TODO: Aufgabenteil 2e. Summiere EintrÃ¤ge
print('\n### 2e ###')
# BEGIN SOLUTION
minimum = np.min(array2)
maximum = np.max(array2)
summe = np.sum(array2)

print(minimum, maximum, summe)
# END SOLUTION

#TERMINAL OUTPUT:
```

2a

2b

2 6 10 [4, 6] [4, 6, 8, 10] [2, 4, 6, 8] [2, 4, 6]

Die Indizierung beginnt bei 0. Bei negativer Indizierung wird die Liste von hinten durchgezählt. 'x:y' gibt die Elemente von x bis exklusive y aus. Lässt man x oder y weg beginnt(endet) die Liste vom ersten Element(am

letzten Element).

2c

[2, 4, 6, 8, 100]

2d

[2 4 6 8 10] [2 4 6 8 10]

2e

2 10 30

```
#PLOTS:
```

A1\code\P3.py

```
## Hier werden sÃ¤mtliche in dieser Datei benÃ¶tigten Pakete und Module geladen
import numpy as np


## TODO: Aufgabenteil 3a. Vektoren als eindim. Arrays erstellen und Rechnungen ausfÃ¼hren
print('### 3a ###')

a = np.array([1,2,3,4])
b = np.array([2,3])
print(f'a = \n{a} \nb = \n{b}')  # \n = neue Zeile in Terminal

# BEGIN SOLUTION
print("shape a: ",np.shape(a))
print("shape b: ",np.shape(b))
print("a transponiert: ", a.T)
print("b transponiert =", b.T)
print("shape a transponiert: ",str(np.shape(a.T)))
print("shape b transponiert: ",str(np.shape(b.T)))
print('Die transponierten Arrays sind gleich den nicht transponierten Arrays.')

print('a+a = ',a+a)
print('a+a.T = ',a+a.T)
print('Ergebnis der letzten beiden Ausgaben erwartbarerweise identisch.')
#print('a+b = ',a+b)
#print('a+b.T = ',a+b.T)
print('a+b und a+b.T ist aufgrund der unterschiedlichen Dimensionen nicht mÃ¶glich.')

print('a*a =',a*a)
print('a*a.T =',a*a.T)
print('a.T*a =',a.T*a)
#print('a*b =',a*b)
#print('a*b.T =',a*b.T)
print('a*b und a*b.T sind nicht mÃ¶glich da die Dimensionen unterschiedlich sind.')

print('a@a =',a@a)
print('a@a.T =',a@a.T)
print('Berechnet das innere Produkt der Vektoren.')
#print('a@b =',a@b)
#print('a@b.T =',a@b.T)
print('a@b und a@b.T sind nicht mÃ¶glich da die Dimensionen unterschiedlich sind.')


# END SOLUTION


## TODO: Aufgabenteil 3b. Vektoren als zweidim. Arrays erstellen und Rechnungen ausfÃ¼hren
print('\n### 3b ###')

a2 = a.reshape(4,1)
b2 = b.reshape(2,1)
print(f'a2 = \n{a2} \nb2 = \n{b2}')

# BEGIN SOLUTION
```

```
print("shape a2: ",np.shape(a2))
print("shape b2: ",np.shape(b2))
print("a2 transponiert: ", a2.T)
print("b2 transponiert =", b2.T)
print("shape a2 transponiert: ",str(np.shape(a2.T)))
print("shape b2 transponiert: ",str(np.shape(b2.T)))
print('a2 und b2 werden als Spalte, a2.T und b2.T jedoch als Zeile ausgegeben.')

print('a2+a2 = ',a2+a2)
print('a2+a2.T = ',a2+a2.T)
print('Die letzte Ausgabe erzeugt eine Matrix, wobei die EintrÃ¤ge durch Komponentenweise Addition entsteh
#print('a2+b2 = ',a2+b2)
print('a2+b2.T = ',a2+b2.T)
print('a2+b2 ist aufgrund der unterschiedlichen Dimensionen nicht mÃ¶glich. a2+b2.T analog zu a+a.T')

print('a2*a2 =',a2*a2)
print('a2*a2.T =',a2*a2.T)
print('a2.T*a2 =',a2.T*a2)
#print('a2*b2 =',a2*b2)
print('a2 und b2 haben keine vertrÃ¤glichen Dimensionen')
print('a2*b2.T =',a2*b2.T)
print('Der * Operator funktioniert bei Vektoren gleicher shape wie zuvor bei den anderen wird jetzt jedoch

#print('a2@a2 =',a2@a2)
print('a2@a2.T =',a2@a2.T)
#print('a2@b2 =',a2@b2)
print('a2@b2.T =',a2@b2.T)
print('a2@a2 und a2@b2 sind nicht mÃ¶glich da die Dimensionen nicht kompatibel sind.')
# END SOLUTION


## TODO: Aufgabenteil 3c. Matrizen erstellen
print('\n### 3c ###')
# BEGIN SOLUTION
reihe1 = np.zeros((1,4),dtype=int)
zeile1 = np.ones((3,1),dtype=int)
restmatrixA=np.array([[4,8,12],[1,7,6],[3,4,9]],dtype=int) #erstellen (3,3)-Teilmatrix
A = np.concatenate((zeile1,restmatrixA),axis=1) #erstellen (3,4)-Teilmatrix
A = np.concatenate((reihe1,A),axis=0) #stellen A fertig
print(A)
# END SOLUTION


## TODO: Aufgabenteil 3d. Matrix-Vektor-Produkte berechnen
print('\n### 3d ###')
# BEGIN SOLUTION
print('A@a = ',A@a)
print('A@a2 = ',A@a2)
print('a.T@A@a = ',a.T@A@a)
print('a2.T@A@a2 = ',a2.T@A@a2)
# END SOLUTION


## TODO: Aufgabenteil 3e. Spur, Eigenwerte, Determinante
print('\n### 3d ###')
# BEGIN SOLUTION
```

```
print('Spur(A) = ',np.trace(A))
print('Summe der Diagonale von A : ',np.sum(np.diag(A)))
print('Erwartbarerweise ist Spur(A) = Summe der Diagonale von A')
print('Eigenwerte von A : ',np.linalg.eigvals(A))
print('Determinante von A : ',np.linalg.det(A))
print('Summe der Eigenwerte von A: ',np.sum(np.linalg.eigvals(A)))
print('Produkt der EIgenwerte von A: ',np.prod(np.linalg.eigvals(A)))
print('Summe der Eigenwerte stimmt mit der Spur und das Produkt der Eigenwerte mit der Determinante Ã¼bere
# END SOLUTION

#TERMINAL OUTPUT:
```

3a

a =

[1 2 3 4]

b =

[2 3]

shape a: (4,)

shape b: (2,)

a transponiert: [1 2 3 4]

b transponiert = [2 3]

shape a transponiert: (4,)

shape b transponiert: (2,)

Die transponierten Arrays sind gleich den nicht transponierten Arrays.

a+a = [2 4 6 8]

a+a.T = [2 4 6 8]

Ergebnis der letzten beiden Ausgaben erwartbarerweise identisch.

a+b und a+b.T ist aufgrund der unterschiedlichen Dimensionen nicht möglich.

a*a = [1 4 9 16]

a*a.T = [1 4 9 16]

a.T*a = [1 4 9 16]

a*b und a*b.T sind nicht möglich da die Dimensionen unterschiedlich sind.

a@a = 30

a@a.T = 30

Berechnet das innere Produkt der Vektoren.

a@b und a@b.T sind nicht möglich da die Dimensionen unterschiedlich sind.

3b

a2 =

[[1]

[2]

[3]

[4]]

b2 =

[[2]

[3]]

shape a2: (4, 1)

shape b2: (2, 1)

a2 transponiert: [[1 2 3 4]]

b2 transponiert = [[2 3]]

shape a2 transponiert: (1, 4)

shape b2 transponiert: (1, 2)

a2 und b2 werden als Spalte, a2.T und b2.T jedoch als Zeile ausgegeben.

a2+a2 = [[2]

[4]

[6]

[8]]

a2+a2.T = [[2 3 4 5]

[3 4 5 6]

[4 5 6 7]

[5 6 7 8]]

Die letzte Ausgabe erzeugt eine Matrix, wobei die Einträge durch Komponentenweise Addition entstehen.

a2+b2.T = [[3 4]

[4 5]

[5 6]

[6 7]]

a2+b2 ist aufgrund der unterschiedlichen Dimensionen nicht möglich. a2+b2.T analog zu a+a.T

a2*a2 = [[1]

[4]

[9]

[16]]

a2*a2.T = [[1 2 3 4]

[2 4 6 8]

[3 6 9 12]

[4 8 12 16]]

a2.T*a2 = [[1 2 3 4]

[2 4 6 8]

[3 6 9 12]

[4 8 12 16]]

a2 und b2 haben keine verträglichen Dimensionen

a2*b2.T = [[2 3]

[4 6]

[6 9]

[8 12]]

Der * Operator funktioniert bei Vektoren gleicher shape wie zuvor bei den anderen wird jetzt jedoch eine Matrix erzeugt.

a2@a2.T = [[1 2 3 4]

[2 4 6 8]

[3 6 9 12]

[4 8 12 16]]

a2@b2.T = [[2 3]

[4 6]

[6 9]

[8 12]]

a2@a2 und a2@b2 sind nicht möglich da die Dimensionen nicht kompatibel sind.

3c

[[0 0 0 0]

[1 4 8 12]

[1 1 7 6]

[1 3 4 9]]

3d

A@a = [0 81 48 55]

A@a2 = [[0]

[81]

[48]

[55]]

a.T@A@a = 526

a2.T@A@a2 = [[526]]

3d

Spur(A) = 20

Summe der Diagonale von A : 20

Erwartbarerweise ist Spur(A) = Summe der Diagonale von A

Eigenwerte von A : [16.51560977 0.48439023 3. 0.]

Determinante von A : 0.0

Summe der Eigenwerte von A: 20.0

Produkt der EIgenwerte von A: 0.0

Summe der Eigenwerte stimmt mit der Spur und das Produkt der Eigenwerte mit der Determinante überein.

```
#PLOTS:
```