03c_NumPy_Miniuebungen_Loesungen

0.1 Mini-Aufgaben zur Überprüfung des Verständnis: ndarrays

Legen Sie die folgenden Matrizen mittels arange() und reshape() an:

$$A = \begin{pmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 13 & 15 & 17 \end{pmatrix} \qquad B = \begin{pmatrix} 17 & 15 & 13 \\ 11 & 9 & 7 \\ 5 & 3 & 1 \end{pmatrix}$$

Hinweis: Dies ist jeweils mit einem Einzeiler möglich.

[4]:
```
import numpy as np

A = np.arange(1,18,2).reshape((3,3))
print(A)

B = np.arange(17,0,-2).reshape((3,3))
print(B)
```

```
[[ 1  3  5]
 [ 7  9 11]
 [13 15 17]]
[[17 15 13]
 [11  9  7]
 [ 5  3  1]]
```

Erreichen Sie dasselbe Ergebnis ohne Verwendung von reshape(). Hierzu sollten zwei Codezeilen pro Matrix ausreichen.

[13]:
```
import numpy as np

A = np.arange(1,18,2)
A.shape = (3,3)
print(A)

B = np.arange(17,0,-2)
B.shape = (3,3)
print(B)
```

```
[[ 1  3  5]
 [ 7  9 11]
```

```
 [13 15 17]]
[[17 15 13]
 [11  9  7]
 [ 5  3  1]]
```

Schreiben Sie Code, der mittels diag() für eine beliebige Länge der Hauptdiagonale Matrizen der folgenden Form erzeugt (nicht angegebene Einträge sind Nullen):

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

Hinweis: Matrizen können in NumPy mit + addiert werden.

[17]:
```
import numpy as np

n = 5

# Version mit list comprehension
diag1 = np.array([2 for i in range(5)])
# effizientere Alternative:
diag1 = 2 * np.ones(5)

diag2 = np.array([-1 for i in range(4)]) # Wichtig ist die 4: sonst funktioniert␣
 ↪die Addition weiter unten nicht
# effizientere Alternative:
diag2 = - np.ones(4)


A = np.diag(diag1) + np.diag(diag2,-1) + np.diag(diag2,1)
print(A)
```

```
[[ 2. -1.  0.  0.  0.]
 [-1.  2. -1.  0.  0.]
 [ 0. -1.  2. -1.  0.]
 [ 0.  0. -1.  2. -1.]
 [ 0.  0.  0. -1.  2.]]
```

Erzeugen Sie mit linspace() jeweils mit einem Einzeiler die folgenden Vektoren: * Zahlen von 1 bis 10 * 10 äquidistante Punkte im Intervall $[0, \pi]$

Hinweis: NumPy stellt die Konstante π natürlich auch bereit. Finden Sie heraus wie, anstatt das Paket math aus der Standardbibliothek zu verwenden.

[12]:
```
import numpy as np

a = np.linspace(1,10,10)
print(a)
```

```
b = np.linspace(0,np.pi,10)
print(b)
```

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
[0.         0.34906585 0.6981317  1.04719755 1.3962634  1.74532925
 2.0943951  2.44346095 2.7925268  3.14159265]
```

0.2 Impressum

0.2.1 Programmierkurs Python, Dominik Göddeke https://www.ians.uni-stuttgart.de, Universität Stuttgart

Version vom April 2023

Gefördert durch die Stiftung Innovation in der Hochschullehre. (alle Rechte am Logo vorbehalten)

Gefördert mit Mitteln der Deutschen Forschungsgemeinschaft (EXC 2075 - 390740016) im Rahmen der Exzellenzstrategie.

[ ]: