# 04_numpy_exercise

October 18, 2020

## 1 Übung numpy - Lösung

### 1.1 Import

[2]:
```
# Importiere numpy als np
# ...
```

### 1.2 Einfache Initialisierungen

[4]:
```
noten = [1, 2, 3, 4, 5, 6]
# Erstelle aus der Liste einen Numpy Vektor
noten_as_vector = #...
```

[5]:
```
# Erstelle nun selbigen Vektor direkt mit np.arange
# ...
```

[5]:
```
array([1, 2, 3, 4, 5, 6])
```

[7]:
```
# Erstelle nun selbigen Vektor direkt mit np.linspace
# ...
```

[7]:
```
array([1., 2., 3., 4., 5., 6.])
```

[10]:
```
# Erstelle eine Einheitsmatrix (1 auf der Diagonale, sonst überall0 0) der␣
 ↪Dimension (6,6)
einheitsmatrix = #...
# Lasse dir die Einheitsmatrix ausgeben
print(einheitsmatrix)
```

```
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]]
```

[11]:
```
# Erstelle nun eine Matrix mit 1,2,...,6 auf der Diagonalen und sonst überall 0␣
↪indem du die Matrix Einheitsmatrix mit dem Vektor Noten multiplizierst.
# ...
```

[11]:
```
array([[1., 0., 0., 0., 0., 0.],
       [0., 2., 0., 0., 0., 0.],
       [0., 0., 3., 0., 0., 0.],
       [0., 0., 0., 4., 0., 0.],
       [0., 0., 0., 0., 5., 0.],
       [0., 0., 0., 0., 0., 6.]])
```

### 1.3 Zufallszahlen und einfache Operationen

[13]:
```
# Initialisiere eine Zufallszahlengenerator mit np.random.default_rng()
zufallszahlengenerator = # ...
```

[16]:
```
# Lasse dir eine 6x6-Matrix ausgeben, mit normalverteilten Werten mit␣
↪Mittelwert 2.5 und Varianz 2
normalmatrix = #...
print(normalmatrix)
```

```
[[-1.3039024   2.50980685  2.33246813  3.99902598  5.6763469   0.40504878]
 [ 3.4860015  -0.39550172  3.7485764   3.44680994  2.23258314  1.41745698]
 [-0.72904811  4.8349993   5.7016118  -2.17527762  5.26903784  2.88529045]
 [ 3.84383992  4.65938939  6.14326728  3.17871095  1.30081294  0.19693054]
 [ 5.89501237  3.19392031  3.04827936  0.69635245  3.93599965  1.71493758]
 [ 3.34551799  2.54869148  1.83685982  3.62210049  6.05018422  2.27957831]]
```

[17]:
```
# Multipliziere nun die Matrix mit der Exponentialfunktion. Dann sind deine␣
↪erzeugten Zufallsvariablen Log-normalvereilt
# ...
```

[17]:
```
array([[2.71470337e-01, 1.23025536e+01, 1.03033402e+01, 5.45449962e+01,
        2.91881210e+02, 1.49937563e+00],
       [3.26551149e+01, 6.73342128e-01, 4.24605919e+01, 3.14000641e+01,
        9.32391999e+00, 4.12661304e+00],
       [4.82367933e-01, 1.25838495e+02, 2.99349503e+02, 1.13576618e-01,
        1.94228993e+02, 1.79087684e+01],
       [4.67044720e+01, 1.05571599e+02, 4.65572244e+02, 2.40157760e+01,
        3.67228081e+00, 1.21765946e+00],
       [3.63221329e+02, 2.43838324e+01, 2.10790438e+01, 2.00642083e+00,
        5.12133199e+01, 5.55632869e+00],
       [2.83752700e+01, 1.27903564e+01, 6.27679699e+00, 3.74160775e+01,
        4.24191166e+02, 9.77255856e+00]])
```

[20]:
```
# Lasse dir die 6. Zeile und dort die 2.-3. Spalte von normalmatrix ausgeben
print(#...)
```

```
[2.54869148 1.83685982]
```

### 1.4 Und zum Abschluss noch eine etwas schwierigere Aufgabe

[24]:
```
# Lasse dir alle Einträge von Normalmatrix ausgeben, die über dem empirischen␣
 ↪Mittelwert aller Beobachtungen sind
normalmatrix_larger = # ...BlockingIOError
print(normalmatrix_larger)
```

```
[3.99902598 5.6763469  3.4860015  3.7485764  3.44680994 4.8349993
 5.7016118  5.26903784 2.88529045 3.84383992 4.65938939 6.14326728
 3.17871095 5.89501237 3.19392031 3.04827936 3.93599965 3.34551799
 3.62210049 6.05018422]
4.298196101292535
3.8899197857477077
```

[ ]:
```
# Berechne nun den Mittelwert und den Median der davor ausgewählten Elemente
print(#...)
print(#...)
```