Deep Learning
Exercise "Introduction to NumPy"

Prof. Dr. Jürgen Brauer

NumPy (Numerical Python Extensions) is a Python library that is widely used whenever we need a data structure for storing n-dimensional arrays (including vectors, matrices, but also 3D arrays, etc.) and also supports linear algebra functions that can be applied onto these arrays. Its long history goes back to the year 1995 and NumPy's predecessors were the libraries Numeric and Numarray. NumPy 1.0 was released in 2006.

You can install NumPy directly with the help of the *Anaconda Navigator*. However, I recommend to install *TensorFlow* directly, which has dependencies to *NumPy* and therefore NumPy will be installed automatically as well, since we will need TensorFlow in the next exercises.

Exercise 1 - Version check

How can you output the version of your NumPy installation?

```
Your NumPy version is 1.13.1
```

Exercise 2 - Generating arrays

a) Generate an 1D array a1 that consists of 5 integers. Then output the data type used to store the entries of this array.

```
a1= [ 2  4  6  8 10]
Numbers in a1 are stored using data type int32
```

b) Change the data type that is used to store the numbers in a1 to float with 32 bit and make sure you did it right by outputting the data type used for storing the numbers of a1 again:

```
Numbers in a1 are now stored using data type float32
```

c) Generate a 2D array a2 of size 3x2 (rows x columns) with the following numbers and output it. Also output the number of dimensions of this array, the number of rows and number of columns.

```
a2= [[1 2]
 [3 4]
 [5 6]]
number of dimensions of a2:  2
number of rows    of a2: 3
number of columns of a2: 2
```

d) Output how much bytes are used to store a single array element of a2 and how many bytes are used in total:

```
nr of bytes needed to store one a2 array element : 4
nr of bytes needed to store all a2 array elements: 24
```

e) Generate the following 3D array a3. Also output the number of dimensions of a3 and the size of each dimension

```
a3= [[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
number of dimensions of a3:  3
number of slices  of a3: 2
number of rows    of a3: 3
number of columns of a3: 4
```

Exercise 3 - Accessing array elements

a) Get and output the value of the element of a3 in the second slice, third row, fourth column:

```
Value of that element is  24
```

b) Change the value of that element to 42 and output the value again by retrieving the value at that position from array a3:

```
Value of that element is now 42
```

c) Store the first slice from the 3D array a3 in a new 2D array a4 and output it:

```
a4= [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

d) Now retrieve from a4 the third column as a 1D array a5 and output it:

```
a5= [ 3  7 11]
```

e) Retrieve from a4 the second row as a 1D array a6 and output it:

```
a6= [5 6 7 8]
```

f) Retrieve from a4 the following 2x2 sub-matrix:

```
a7= [[ 6  7]
 [10 11]]
```

Exercise 4 - Reshaping arrays

a) Generate the following 1D array A. Then reshape it to a 2D array B with 2 rows and 5 columns.

```
A= [ 1  2  3  4  5  6  7  8  9 10]
B= [[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
```

b) Reshape the 2D array B to a 2D array C with 5 rows and 2 columns.

```
C= [[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]]
```

Exercise 5 - Linear algebra with arrays

a) Define the following two matrices A and B in your code. Then compute the sum of the two matrices, the element wise multiplication result and the matrix multiplication result:

```
A=
 [[1 1]
 [0 1]]
B=
 [[2 0]
 [3 4]]
A+B=
 [[3 1]
 [3 5]]
element wise multiplication A*B=
 [[2 0]
 [0 4]]
matrix multiplication A*B=
 [[5 4]
 [3 4]]
```

b) Define the following matrix A. Then compute its inverse A_inv. Check whether the matrix product of both matrices A and A_inv really gives you the 2x2 identity matrix.

```
A=
 [[ 1.  2.]
 [ 3.  4.]]
A_inv=
 [[-2.   1. ]
 [ 1.5 -0.5]]
A * A_inv=
 [[  1.00000000e+00   1.11022302e-16]
 [  0.00000000e+00   1.00000000e+00]]
```

c) What does the function `numpy.eye()`?

d) How can we automatically check whether $A * A\_inv$ gives us the 2x2 identity matrix?

Exercise 6 - Random arrays

a) How can you generate a random matrix of 5 rows and 3 columns with random float values drawn from a uniform distribution with values in the range -1 and +1? Generate such a matrix `rndA`, then output it:

```
rndA=
 [[ 0.67630242 -0.49098576 -0.18706128]
 [ 0.61222022  0.38307423  0.74869381]
 [ 0.16949814  0.16301043 -0.77961425]
 [-0.99861878  0.9521788  -0.55123554]
 [ 0.92839569  0.45590548  0.09234368]]
```

b) How to do the same but with random int values?

```
rndB=
 [[-1  0 -1]
 [ 0 -1  0]
 [-1  0 -1]
 [ 0 -1  0]
 [ 0 -1  0]]
```