1 Python list creation

Create lists with the following properties, choose names like $ex1\_1, 2, 3$ for them:

1. numbers from 10 to 20(included) that are odd
2. numbers from 100 to 0(included) that can be divided by 10 (use %, the modulus operator and list comprehension)
3. numbers from 15 to 1 (included) that can be divided by 3
4. string like "x", "xx", "xxx" repeated 10 times. Use the fact that, in Python, a string s multiplied by an integer n results in s repeated n times.
5. the string "stringX" repeated 5 times, where X goes from 5 to 0(excluded). Use the builtin function str() to convert numbers to strings and the fact that strings can be concatenated using the "+" operator
6. a list with the items "1", 1, 1.0, "one"
7. all the numbers from 0 to 99 that contain the digit "5". You may use the method find() that all strings possess to look for a substring. If it is found, the start index is returned, otherwise -1.

2 Array creation

You may need to combine Python list comprehension and numpy array creation!
a) Create a 1D array with entries from -100 to 0(included) in steps of 2
b) Create a 2D with 3 rows and 2 columns, with row entries 1,1..., 2,2,..., 3,3,...
c) Create a 2D with 3 rows and 2 columns that has the value -1 everywhere
d) Create a 3D tensor with shape (5,4,3) with random normal entries, with mean 0 and standard deviation 1.

3 Numpy basics and slicing

Assume that the 3D array 'traind' contains 2000 images of dimension 20x20. You can, e.g., generate it using traind = np.random.uniform(0,255,[2000,20,20]). Give code snippets for the following operations:
a) Slice out the 1st sample into an array x and print it!
b) Set the 2 lowermost columns of sample 1000 to -1
c) Print the mean pixel value in the 10th data sample
d) Generate the following variations of the 10th sample and store them in a new variable z:
- just keep every 3rd row
- just keep every 3rd column
- inverse all rows but not columns

- invert rows but not colums, just keeping every 2th row
e) Apply the in-place transform

$$1 + x$$

to all samples.

4 Reduction

Take the array 'traind' from the previous exercise and ...
a) Compute the pixel variance for pixel 0,0 over all samples
b) Compute the pixel argmax for pixel 0,0 over all samples
c) Compute the "standard deviation image" over all samples
d) Compute the row-wise mean over all samples
e) Compute the column-wise mean over all samples

5 Broadcasting

Using 'traind' ...
a) create a 20-element row vector with entries from 1 to 20, and subtract it from all rows of all samples using broadcasting
b) create a 20-element column vector with entries from 1 to 20, and multiply it with all columns of all samples using broadcasting
c) computet he mean imnage over all samples, and subtract it from all samples via broadcasting

6 Fancy indexing and mask indexing

a) create a 20-element vector with entries from 1 to 20, and copy out all elements that are even using mask indexing!
b) create a 20-element vector with entries from 1 to 20, and in-place multiply elements 0,1,2 and 19 by 2 using fancy indexing!

7 Matplotlib

a) plot the function $1/x$ between 1 and 5 using 100 support points!
b) generate a scatter plot of the same data as in a)!
c) generate a bar plot of the same data as in a)!
d) plot $1/x$ and $\sqrt{x}$ together in a single plot, same range as before
e) generate 100 numbers distributed according to a uniform distribution between 0 and 1, and display their histogram!

8 MNIST

'traind' contains the samples, 'trainl' the target values (labels) in one-hot format.
a) Create a scatter plot of the image-wise pixel variance for all samples!
b) Copy out all the samples whose variance over pixels is > 0.3 and display 3 of them
c) Compute the "variance image" and display it! **d)** Compute the "variance image" for samples of class 5 and display it!

9 Softmax theory

Show that the softmax function $\vec{S}(\vec{x})$ is normalized, i.e., that it satisfies $\sum_i S_i(\vec{x}) = 1$

10 Cross-entropy theory

Show that the cross-entropy is always positive!

11 Implementing softmax in TF

Write a python function S(x) which takes an 1D TF tensor and returns the softmax, also as a tf tensor! Print out results for $\vec{x} = [-1, -1, 5]$ and $\vec{x} = [1, 1, 2]$!

12 Implementing cross-entropy in TF

Write a python function CE(y,t) which takes an 1D TFD tensor and returns the its cross-entropy as a TF scalar! Print out results for $\vec{t} = [0, 0, 1]$ in the three cases of $\vec{y} = [0.1, 0.1, 0.8]$ and $\vec{y} = [0.3, 0.3, 0.4]$ and $\vec{y} = [0.8, 0.1, 0.1]$!

13 1D derivatives

Compute the derivative $h'(x)$. Always give rule and decomposition!!
a) $h(x) = \sin(\sin(x))$
b) $h(x) = 2\sin(x) + 5\cos(x)$
c) $h(x) = \alpha\sin(x) + \beta\cos(x)$
d) $h(x) = \ln(\sin(\cos(5.2)))$
e) $h(x) = \exp(x)\ln(x)$
f) $h(x) = x\cos(x)$
g) $h(x) = e^{\cos(x)}$

14 nD derivatives

Compute the following derivatives!
a) $\frac{\partial}{\partial x_1}\sum_{i=1}^{5} x_i$
b) $\frac{\partial}{\partial x_1 6}\sum_{i=1}^{5} x_i$
c) $\frac{\partial}{\partial x_2}\sum_{i=1}^{5}(x_i-5)^2$
d) $\frac{\partial}{\partial x_3}\sum_{i=1}^{5}(x_i^2-x_i)^2$
e) $\frac{\partial}{\partial x_2}\cos(x_3)x_3$
f) $\frac{\partial}{\partial x_3}\cos(x_3)x_4$
g) $\frac{\partial}{\partial x_3}\cos(\exp(x_3))$

15 Vector-vector chain rule

Given the function $\vec{g}(\vec{x})$ with $g_i(\vec{x}) = \exp(x_i)$, and $f(\vec{x}) = \sum_i x_i^2$: compute the partial derivative $\frac{\partial f(g(\vec{x}))}{\partial x_1}$!

16 Gradient descent (on paper)

Consider the function $f(\vec{x}) = x_1^2 + 2x_2$. Assuming a starting point of $\vec{x}(0) = [1, 3]$ and a step size of $\epsilon = 0.1$: perform 2 steps of gradient ascent, that is, compute the values of $\vec{x}(1)$, $\vec{x}(2)$ by iteration. Verify that the value of f is decreasing!

17 Gradient theory

a) Prove that the derivative of the function $f(x) = x$ is 1, in analogy to the example for the function $f(x) = x^2$ that was demonstrated!
b) What is the basic approximation that lies behind gradient computations?
c) Consider the function $f(\vec{x}) = x_1^2 + x_2^2$. Write down the approximation of that function around the point $[0, 0]^T$ and the point $[1, 1]^T$.
d) What are the directions of strongest increase for these two points? Plot both points and direction in a 2D plot (by hand!)

18 Vector-vector chain rule

Given the matrix-matrix function $A(B) = BW$ and $\mathcal{L}(A) = \sum_{l,m} A_{lm}^2$: compute the partial derivative $\frac{\partial \mathcal{L}}{\partial B_{11}}$!

19 Linear regression

Assume a machine learning model given by $Y = f(X, W, \vec{b}) = XW + \vec{b}^T$. The loss is $\mathcal{L}(Y) = N^{-1} \sum_n \sum_k (T_{nk} - Y_{nk})^2$. This is the mean-squared-error loss for linear regression, by the way. Since this is a totally flat model, back-propagation is not required but we can still practise gradient computations. Compute:
a) The derivative $\frac{\partial \mathcal{L}}{\partial Y_{ij}}$
b) The derivative $\frac{\partial \mathcal{L}}{\partial W_{ab}}$
c) The derivative $\frac{\partial \mathcal{L}}{\partial b_a}$

20 A back-propagation step with numbers

Assuming we are faced with a transfer function layer X, $g(x) = \text{relu}(x)$, and we know that $\frac{\partial \mathcal{L}}{\partial A^{(X)}} = \begin{pmatrix} 4 & 1 \\ 0 & 4 \end{pmatrix}$, as well as $A^{(X-1)} = \begin{pmatrix} -2 & -1 \\ 0 & 2 \end{pmatrix}$.

Compute all entries of the matrix $\frac{\partial \mathcal{L}}{\partial A^{(X-1)}_{ij}}$!

21 A vector-scalar function in TF

Let $f(\vec{x}) = \sum_{i=1}^{3} x_i$.
a) Implement this function in TF and compute its output for the inputs $\vec{x}_1 = (1, 2, 3)^T$ and $\vec{x}_2 = (2, 0, 2)^T$. Hint: use a tf function to compute the sum!
b) Use TF to compute and display the value of $\vec{\nabla} f$, evaluated for $\vec{x} = \vec{x}_1$
c) Use TF to compute and display the value of $\frac{\partial f}{\partial x_1}$, evaluated for $\vec{x} = \vec{x}_1$