MLET Assignment 1: Warm up

Names

Please provide all person names (family name, first name) working together on this assignment (MAX 2 in group).
Name1: Orzelek, Peer
Name2: Kostopoulos, Leon
Each one of the group has to submit this notebook.

Introduction

In this first exercise you make sure that your environment for solving the exercises is setup corrrectly. You also get already your first 30 points when you complete the assignments.

You should now already work within the Python virtual environment. When you are able to execute the below code from within the notebook in the browser you are good.

In [ ]:
```
# used for manipulating directory paths
import os

# Scientific and vector computation for python
import numpy as np

# Plotting library
from matplotlib import pyplot

import utils

grader=utils.Grader()

# SET YOUR Authentication Token. To get the token login to http://evalml.da.
AUTH_TOKEN = "cd4a9aa7782d5a61a41c6dd48746e426d1f9a09a"
grader.setToken(AUTH_TOKEN)
```

Submission and Grading

After completing each part of the assignment you are asked to submit your results

In this first intro exercise you will do just do some basic maths (linear algebra).

Required Exercises

| Section | Part |Submitted Function | Points |---------|:- |:- | :-:

| 1 | Identity Matrix | `identityMarix` | 5
| 2 | Determinant of Matrix | `determinantMatrix3x3` | 10
| 3 | Transpose of a Matrix | `transposeMatrix` | 10 | 4 | Matrix Multiplication | `multiplyMatrix` | 5
| | Total Points | | 30

You are allowed to submit your solutions multiple times. We will take only the Highscore into consideration.

> At the end of each section, you will find a code cell which contains code for submitting the solutions thus far to the grader. Execute the cell to see your score up to the current section. For all your work to be submitted properly, you must execute those cells at least once. They must also be re-executed everytime the submitted function is updated. If you once have submitted a correct result, this is saved on the server.

Debugging

Here are some things to keep in mind throughout this exercise:

- Python array indices start from zero, not one (contrary to OCTAVE/MATLAB).
- There is an important distinction between python arrays (called `list` or `tuple`) and `numpy` arrays. You should use `numpy` arrays in all your computations. Vector/matrix operations work only with `numpy` arrays. Python lists do not support vector operations (you need to use for loops).
- If you are seeing many errors at runtime, inspect your matrix operations to make sure that you are adding and multiplying matrices of compatible dimensions. Printing the dimensions of `numpy` arrays using the `shape` property will help you debug.
- By default, `numpy` interprets math operators to be element-wise operators. If you want to do matrix multiplication, you need to use the `dot` function in `numpy`. For, example if `A` and `B` are two `numpy` matrices, then the matrix operation AB is `np.dot(A, B)`. Note that for 2-dimensional matrices or vectors (1-dimensional), this is also equivalent to `A@B` (requires python >= 3.5).

1 Identity Matrix

In this first part you are asked to provide a function that returns a NxN identity matrix. The input paramter n specifies the number of rows and cols. Hint: You could use `numpy` for this task.

In [ ]:
```
def identityMatrix(n):
    """
    Computing a NxN identity matrix

    Returns
    -------
    A : array_like
        The NxN identity matrix.

    Instructions
    ------------
    Return the NxN identity matrix.
    """
    A = np.zeros(shape=(3, 3))

    # ======== WRITE YOUR CODE HERE ======
    A = np.identity(n) # creates an identity matrix of size n
    print(A)
    # ====================================
    return A
```

The previous cell only defines the function `identityMarix`. We can now run it by executing the following cell to see its output. You should see output similar to the following:

```
array([[ 1.,  0.,  0.],
       [ 0.,  1., 0.],
       [ 0.,  0.,  1.]])
```

In [ ]:
```
idMatFunc = identityMatrix
print(idMatFunc)
identityMatrix(3)
```

```
<function identityMatrix at 0x10ca34360>
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Out[ ]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

1.1 Submitting solutions

Here you can now submit your solution to get a grade.

To use this function you need to enter your your personal SUBMISSION_TOKEN in the first code block. If you have done that you can now execute the next code block to submit your results.

You can now submit your solutions.

In [ ]:
```
# appends the implemented function in part 1 to the grader object
grader.setFunc("identityMatrix", identityMatrix)

# send the added functions to the grader for getting a grade on this part
grader.grade()
```

```
Submitting Solutions | Programming Exercise
                            Identity Matrix |   5 /   5 | Nice work!
                          Determinant Matrix |   0 /  10 | wrong
                            Transpose Matrix |   0 /  10 | wrong
                             Multiply Matrix |   0 /   5 | wrong
                                   ---------------------------------
                                             |    5 / 30 |
```

2 Determinant of a Matrix

In mathematics, the determinant is a scalar value that is a function of the entries of a square matrix. It allows characterizing some properties of the matrix and the linear map represented by the matrix. In particular, the determinant is nonzero if and only if the matrix is invertible and the linear map represented by the matrix is an isomorphism. The determinant of a product of matrices is the product of their determinants (the preceding property is a corollary of this one). The determinant of a matrix A is denoted det(A), det A, or |A|. We will use the determinant later for e.g. eigenvalue decomposition. In the function determinantMatrix3x3 you are asked to compute the determinant for a 3x3 matrix. **Don't use the numpy function for that. Implement on your own.** You can use the Leibniz formula for that:

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$= aei + bfg + cdh - ceg - bdi - afh.$$

In [ ]:
```
def determinantMatrix3x3(M):
    """
    Computing the determinant of a 3x3 Matrix


    Returns
    -------
    det : float
    The determinant of the matrix M.
```

```
    Instructions
    ------------
    Return the determinant of M
    """
    mat = np.asarray(M.copy())
    det = 0

    # ======== WRITE YOUR CODE HERE (use variable mat, return a float) ===
    det = np.linalg.det(mat) # calculates the determinant of the matrix
    # or different version via formular with the matrix elements
    # det = mat[0,0]*mat[1,1]*mat[2,2] + mat[0,1]*mat[1,2]*mat[2,0] + mat[

    # ====================================
    return det
```

CHECK: The determinant of the given matrix below should be: 137180

In [ ]:
```
func = determinantMatrix3x3
print(func)
# creating a 3X3 Numpy matrix
n_array = np.array([[55, 25, 15],
                    [30, 44, 2],
                    [11, 45, 77]])
print(int(determinantMatrix3x3(n_array)))
```

```
<function determinantMatrix3x3 at 0x10d4d28e0>
137180
```

2.1 Submitting solutions

Make sure your code does the right thing and you have entered your SUBMISSION_TOKEN above. *You can now submit your solutions.*

In [ ]:
```
# appends the implemented function in part 2 to the grader object
grader.setFunc("determinantMatrix3x3", determinantMatrix3x3)

# getting a grade on this part
grader.grade()
```

```
Submitting Solutions | Programming Exercise
                            Identity Matrix |   5 /   5 | Nice work!
                       Determinant Matrix |  10 /  10 | Nice work!
                         Transpose Matrix |   0 /  10 | wrong
                          Multiply Matrix |   0 /   5 | wrong
                                ---------------------------------
                                          |   15 / 30 |
```

3 Transpose of a Matrix

We will also often use the transpose of a matrix. Exercise yourself and write the function transposeMatrix to transpose a Matrix of size 3x3. **Don't use the numpy function for**

that. Implement on your own.

In [ ]:
```
def transposeMatrix(M):
    """
    Computing the determinant of a NxN Matrix

    Returns
    -------
    A : array_like
        The NxN transposed matrix.

    Instructions
    ------------
    Return the transposed matrix of M
    """
    mat = np.asarray(M.copy())
    A = np.zeros(shape=(3, 3))

    # ======== WRITE YOUR CODE HERE (use variable mat) ======
    A = np.transpose(mat)


    # ====================================
    return A
```

CHECK: The transposed matrix of the below given Matrix should be:

```
array([[55, 30, 11],
       [25, 44, 45],
       [15,  2, 77]])
```

In [ ]:
```
func = transposeMatrix
print(func)
n_array = np.array([[55, 25, 15],
                    [30, 44, 2],
                    [11, 45, 77]])
transposeMatrix(n_array)
```

```
<function transposeMatrix at 0x10d4d2480>
```

Out[ ]:
```
array([[55, 30, 11],
       [25, 44, 45],
       [15,  2, 77]])
```

3.1 Submitting solutions

Make sure your code does the right thing and you have entered your SUBMISSION_TOKEN above. *You can now submit your solutions.*

In [ ]:
```
# appends the implemented function in part 3 to the grader object
grader.setFunc("transposeMatrix", transposeMatrix)

# getting a grade on this part
grader.grade()
```

```
Submitting Solutions | Programming Exercise
                            Identity Matrix |   5 /   5 | Nice work!
                          Determinant Matrix |  10 /  10 | Nice work!
                            Transpose Matrix |  10 /  10 | Nice work!
                             Multiply Matrix |   0 /   5 | wrong
                                   ------------------------------
                                             |   25 / 30 |
```

4 Matrix Multiplication

Mutliply to matrices.

In [ ]:
```python
def multiplyMatrix(A,B):
    """
    Computing the result of a matrix multiplication of A and B

    Returns
    -------
    A,B : array_like
          Input matrices for multiplication

    Instructions
    ------------
    Return the result of the matrix multiplication C = AB.
    """
    mat1 = np.asarray(A.copy())
    mat2 = np.asarray(B.copy())
    C = np.zeros(shape=(3, 3))

    # ======== WRITE YOUR CODE HERE ======
    C = np.dot(mat1, mat2)


    # ====================================
    return C
```

CHECK: The transposed matrix of the below given Matrix should be:

```
A = np.array([[50, 25, 10],
                    [30, 44, 2],
                    [11, 45, 77]])
B = np.array([[55, 25, 15],
                [30, 44, 2],
                    [11, 45, 77]])
```

should give:

```
array([[3610, 2800, 1570],
       [2992, 2776,  692],
       [2802, 5720, 6184]])
```

In [ ]:
```python
func = multiplyMatrix
```

```
print(func)
A = np.array([[50, 25, 10],
                     [30, 44, 2],
                     [11, 45, 77]])
B = np.array([[55, 25, 15],
                [30, 44, 2],
                     [11, 45, 77]])
multiplyMatrix(A,B)
```

```
<function multiplyMatrix at 0x10d4d2020>
```

Out[ ]:
```
array([[3610, 2800, 1570],
       [2992, 2776,  692],
       [2802, 5720, 6184]])
```

4.1 Submitting solutions

Make sure your code does the right thing and you have entered your SUBMISSION_TOKEN above. *You can now submit your solutions.*

In [ ]:
```
# appends the implemented function in part 4 to the grader object
grader.setFunc("multiplyMatrix", multiplyMatrix)

# getting a grade on this part
grader.grade()
```

```
Submitting Solutions | Programming Exercise
                           Identity Matrix |   5 /   5 | Nice work!
                        Determinant Matrix |  10 /  10 | Nice work!
                          Transpose Matrix |  10 /  10 | Nice work!
                           Multiply Matrix |   5 /   5 | Nice work!
                                 ---------------------------------
                                           |   30 / 30 |
```

5 Plotting the Data

Plot a dataset. In the next exercise you are often given a dataset and you need to apply certain ML techniques on it. In those exercises, as well in real life, it is always a good idea to look at the data first and understand the shape of the data. Here the matplotlib gives you powerful tools at hand to visualize data points. Additionally you are interested using plots to evaluate your ML algorithm on the dataset. Also for plotting evaluation figures matplotlib is a good first choice. In the following a dataset is loaded wich contains 100 2d data points. This toy dataset provides a relation between study hours and gained course grades. Your small exercise here is to plot this dataset. Note: This is simulated data and has nothing to do with ML grades and study hours ;-)

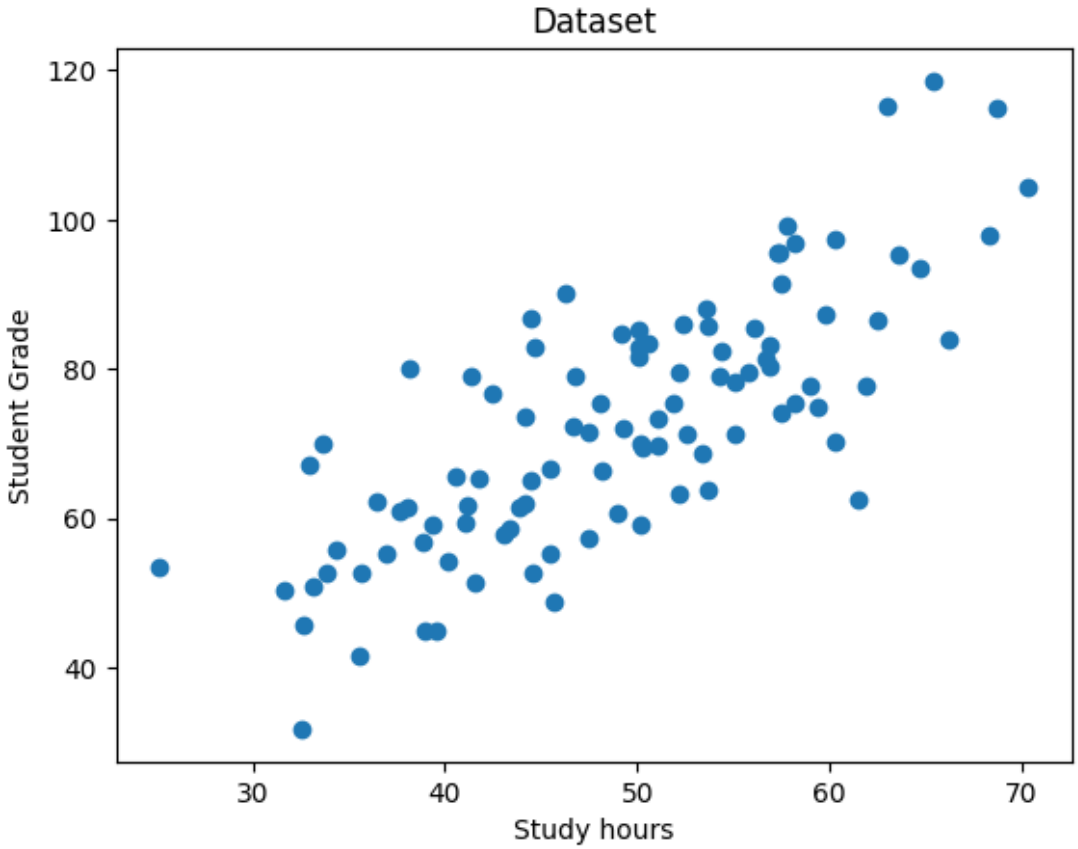
In [ ]:
```
points = np.genfromtxt('data/data.csv', delimiter=',')

#Extract columns
X = np.array(points[:,0])
```

```
y = np.array(points[:,1])
```

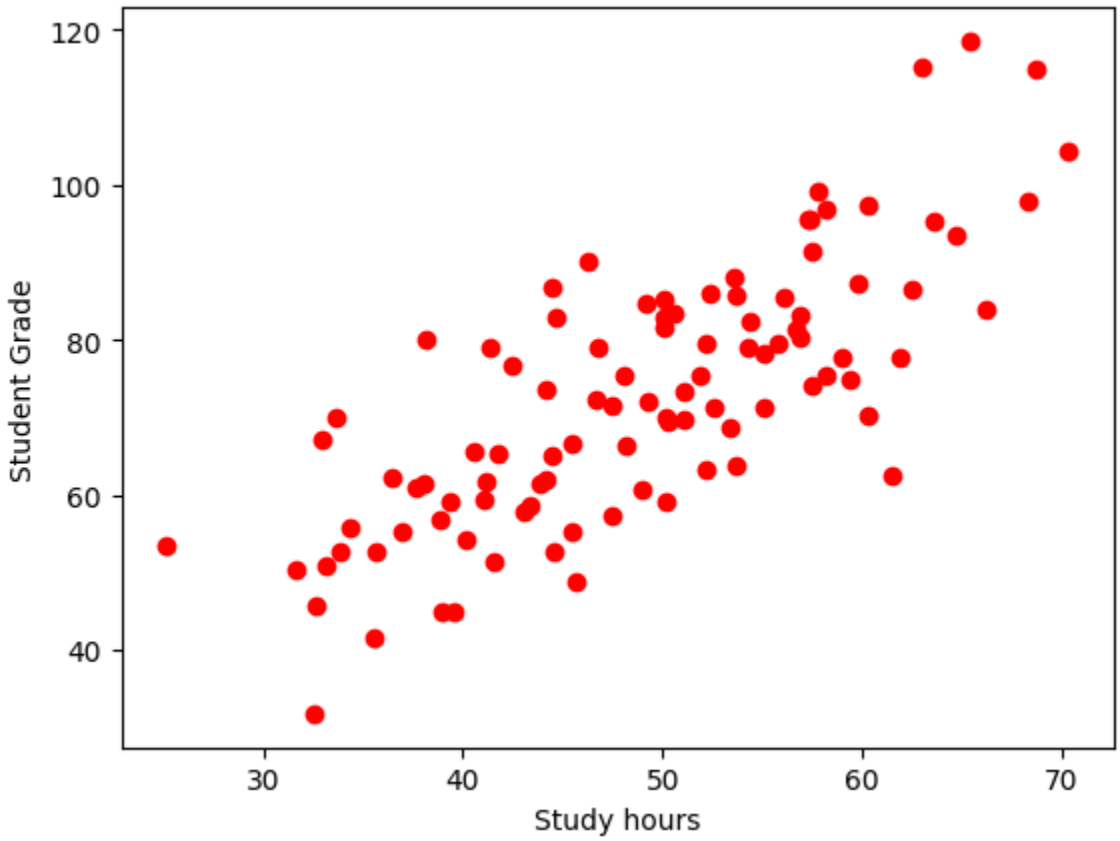
Plot the data with the function plotData. Use the matplotlib documentation to find out what functions you need to use. Also make sure axes are labeled correctly (X axis = 'Study hours', Y axis = "Student Grade")

In [ ]:
```
def plotData(x, y):
    """
    Plots ALL data points x and y into a 2D figure.

    Parameters
    ----------
    x : array_like
        Data point values for x-axis. Note x and y should have the same size

    y : array_like
        Data point values for y-axis. Note x and y should have the same size

    Instructions
    ------------
    Plot the training data into a figure using "figure" and "plot"
    functions. Set the axes labels using the "xlabel" and "ylabel" functions
    Optional: Change using red circles instead blue points.
    """
    fig = pyplot.figure()  # open a new figure

    # ====================== YOUR CODE HERE ======================
    pyplot.plot(x, y, 'ro')
    pyplot.xlabel('Study hours')
    pyplot.ylabel('Student Grade')


    # ============================================================
```

CHECK: By executing the code below your plot should look similar to this:



In [ ]:
```
plotData(X, y)
```



To learn more about the `matplotlib` plot function and what arguments you can

provide to it, you can type `?pyplot.plot` in a cell within the jupyter notebook.

Upload PDF form Notebook in Moodle

Well done. You are now ready with this assignment.

Please make sure that ALL code cells are executed and all results are visible.
Make sure submitting in time.
Create a PDF from the notebook. You can do this via Visual Code directly: Export as HTML / Nach HTML konvertieren (over command palette / Befelspalette) and then print PDF from HTML.
Submit the PDF in the correct task space on moodle:
"Ex1_[FIRST_NAME]_[LAST_NAME]". Submit only one PDF.