תרגיל בית 1 בלמידת מכונה

We define $\sigma = 1, n = 1000, m = 2$ when $\beta = \begin{pmatrix}1\\2\end{pmatrix}$.

For these values, we got $\hat{\beta} \approx \begin{pmatrix}0.956\\2.082\end{pmatrix}$.

For $n = 100$, $\hat{\beta} \approx \begin{pmatrix}1.205\\1.983\end{pmatrix}$.

For $n = 10$, $\hat{\beta} \approx \begin{pmatrix}0.990\\0.841\end{pmatrix}$.

The more samples we have, the best is the estimation. Let us define $n = 1000$. We would like to see how the variations of σ are influencing the value of $\hat{\beta}$.

For $\sigma = 0.1$, $\hat{\beta} \approx \begin{pmatrix}0.982\\2.029\end{pmatrix}$.

For $\sigma = 0.01$, $\hat{\beta} \approx \begin{pmatrix}1.000\\2.004\end{pmatrix}$.

The smaller σ is, the better the result of the linear regression is.

```python
################################################################
# FILE : main.py
# WRITER : Raphael Haehnel
# DESCRIPTION: Introduction to Machine Learning (BIU), ex1
################################################################

import numpy as np


def generate_x(n: int, m: int):
    """
    Generate a random matrix of dimension n x m
    :param n: The number of samples
    :param m: The number of features
    :return: The matrix X
    """
    return np.random.uniform(low=0, high=1, size=(n, m))


def generate_beta(m: int):
    """
    Generate the weight matrix
    :param m: The number of features
    :return: The matrix beta
    """
    return np.array([np.arange(1, m+1, 1)]).T


def generate_epsilon(n: int, sigma: float = 1.0):
    """
    Generate a random vector
    :param n: Size of the vector
    :param sigma: The standard deviation of the random generated elements
    :return: A vector of random elements
    """
    return np.random.normal(0, sigma, size=(n, 1))


def compute_y(x: np.ndarray, beta: np.ndarray, epsilon: np.ndarray):
    """
    Compute the y values of the training example
    :param x: The x values of the training example
    :param beta: The vector of parameters
    :param epsilon: The vector of random values
    :return: A vector of values
    """
    return np.dot(x, beta) + epsilon


def solve_beta(x: np.ndarray, y: np.ndarray):
    """
    Find the beta parameters of the linear regression
    :param x: Training example
    :param y: Training example
    :return: The parameter vector beta of the regression model
```

```
    """
    a1 = np.linalg.inv(np.dot(x.T, x))
    a2 = np.dot(x.T, y)
    return np.dot(a1, a2)


if __name__ == '__main__':
    n = 40
    m = 2
    sigma = 1

    X = generate_x(n, m)
    beta = generate_beta(m)
    epsilon = generate_epsilon(n, sigma)
    Y = compute_y(X, beta, epsilon)
    new_beta = solve_beta(X, Y)


```