```
import numpy as np
```

zad.1

```
A = np.array([[1, 1, 3, -1],
             [2, 3, 5, 9],
             [-2, 3 ,5, 7],
             [-7, 9, 2 ,1]])

B = np.array([3, 5, -4, -2])

C = np.array([[3, 2, 1],
             [3, 1, -4],
             [-2, 3, 5],
             [-1, 5, 7]])
```

a)

```
import tensorflow as tf

At = tf.constant(
[[1, 1, 3, -1],
[-2, 3, 5, 9],
[2, 3, 5, 7],
[-7, 9, 2, 1]])

Bt = tf.constant(
[[3, 5, -4, -2]])

Ct = tf.constant(
[[3, 2, 1],
[3, 1, -4],
[-2, 3, 5],
[-1, 5, 7]])
```

b)

```
B1 = B.reshape(4,1).copy()
B1
```

```
     array([[ 3],
            [ 5],
            [-4],
            [-2]])
```

c)

```
print(A*B)
print("")
print(A*B1)
```

```
     [[  3   5 -12   2]
      [  6  15 -20 -18]
      [ -6  15 -20 -14]
      [-21  45  -8  -2]]

     [[  3   3   9  -3]
      [ 10  15  25  45]
      [  8 -12 -20 -28]
      [ 14 -18  -4  -2]]
```

d)

```
print(np.dot(A,B1))
```

```
     [[ -2]
      [-17]
      [-25]
      [ 14]]
```

```
print(np.dot(A,C))
```

```
     [[  1   7   5]
      [ -4  67  78]
      [-14  49  60]
      [  1   6  26]]
```

e)

```
print(np.linalg.inv(A))
print("")
#print(np.linalg.inv(B))
#print("")
#print(np.linalg.inv(C))
#print("")
#print(np.linalg.inv(B1))
```

```
    [[ 0.08108108  0.21829522 -0.27027027  0.00831601]
     [ 0.02702703  0.18814969 -0.25675676  0.13097713]
     [ 0.24324324 -0.11434511  0.18918919 -0.05197505]
     [-0.16216216  0.06340956  0.04054054 -0.01663202]]
```

f)

```
print(np.sum(A, axis=0))
print("")
print(np.sum(A, axis=1))
print("")
print(np.sum(A))
```

```
    [-6 16 15 16]

    [ 4 19 13  5]

    41
```

zad.2

a) *Znajdź wartości i wektory własne macierzy* `(21 12)`

b) *Sprawdź czy wektory własne tworzą kąt prosty*

```
A = [[2, 1],
     [1, 2]]

"""
wartości własne macierzy (numpy.linalg.eigvals)
wartości i wektory własne macierzy (eig)
"""
z1 = np.linalg.eigvals(A)
print(z1)

print("")

z2 = np.linalg.eig(A)
print(z2)
```

```
    [3. 1.]

    (array([3., 1.]), array([[ 0.70710678, -0.70710678],
           [ 0.70710678,  0.70710678]]))
```

```
from numpy import array
import math

z21 = array([0.70710678, -0.70710678])
z22 = array([0.70710678,  0.70710678])

# iloczyn slalarny
iloczyn = (0.70710678 * 0.70710678) + (0.70710678 * 0.70710678)
print(iloczyn)

print("")
dlugosc = math.sqrt((0.70710678 * 0.70710678) + (0.70710678 * 0.70710678))
print(dlugosc)

print("")

fin = math.cos(iloczyn/(dlugosc*dlugosc))
print(fin)
```

```
    0.9999999966439369

    0.9999999983219684
    0.5403023058681395
```

2.a)

```
A = np.array([[2.0, 4.0, 5.0],
              [4.0, 5.0, 1.0],
              [5.0, 1.0, 3.0]])
```

```
from numpy import linalg as LA
w, v = LA.eig(A)
w, v
```

```
    (array([10.05548601, -3.25927992,  3.20379391]),
     array([[-0.6164593 , -0.76754779,  0.1756369 ],
            [-0.59073031,  0.30335923, -0.7476703 ],
            [-0.52059161,  0.56466235,  0.64042236]]))
```

```
print(np.dot(A, v[:,0]))
print("")
print(np.dot(w[0], v[:,0]))
print("")
print(np.dot(A, v[:,1]))
print("")
print(np.dot(w[1], v[:,1]))
# odpowiadają wektoram własnym
```

```
    [-6.19879791 -5.94008038 -5.23480167]

    [-6.19879791 -5.94008038 -5.23480167]

    [ 2.5016531  -0.98873265 -1.84039266]

    [ 2.5016531  -0.98873265 -1.84039266]
```

zad.3

```
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
```

```
    Mounted at /content/drive
```

```
import os
os.chdir('/content/drive/My Drive/DM')
```

```
import pandas as pd
import numpy as np
data = pd.read_csv('simple_dataset.csv')
#print(data)
data
```

	X	B	C	D	E
0	1	12	6	5	-4
1	2	11	-4	7	-2
2	3	21	8	-2	9
3	4	4	12	1	10

```
#loc - indeks wprost
S1 = data.loc[1,:]
S1
```

```
    X     2
    B    11
    C    -4
    D     7
    E    -2
    Name: 1, dtype: int64
```

```
S2 = data.loc[1:2,:]
S2
```

	X	B	C	D	E

```
S3 = data.loc[:,'B':'D']
S3
```

	B	C	D
0	12	6	5
1	11	-4	7
2	21	8	-2
3	4	12	1

```
#iloc - indeks domyślny (liczbowy)
S4 = data.iloc[:,1:4]
S4
```

	B	C	D
0	12	6	5
1	11	-4	7
2	21	8	-2
3	4	12	1

```
S4_1 = np.array(data.iloc[:,1:4])
S4_1
```

```
array([[12,  6,  5],
       [11, -4,  7],
       [21,  8, -2],
       [ 4, 12,  1]])
```

a-d)

```
S1 = np.array(data.loc[1,:])
S1
```

```
array([ 2, 11, -4,  7, -2])
```

```
S2 = np.array(data.loc[1:2,:])
S2
```

```
array([[ 2, 11, -4,  7, -2],
       [ 3, 21,  8, -2,  9]])
```

```
S3 = np.array(data.loc[2:3,'B':'D'])
S3
```

```
array([[21,  8, -2],
       [ 4, 12,  1]])
```

```
S4_1 = np.array(data.loc[:,'B'])
print(S4_1)
print("")

S4_2 = np.array(data.loc[:,'D'])
print(S4_2)
print("")

S4 = np.concatenate((S4_1, S4_2), axis = None)
print(S4)
print("")
arr = np.stack((S4_1, S4_2), axis=1)
arr
```

```
[12 11 21  4]

[ 5  7 -2  1]

[12 11 21  4  5  7 -2  1]

array([[12,  5],
       [11,  7],
       [21, -2],
       [ 4,  1]])
```

zad.4

```
import pandas as pd
import numpy as np
data = pd.read_csv('president_heights.csv')
print(data)
```

```
    order                    name  height(cm)
0       1       George Washington         189
1       2              John Adams         170
2       3        Thomas Jefferson         189
3       4           James Madison         163
4       5            James Monroe         183
5       6       John Quincy Adams         171
6       7          Andrew Jackson         185
7       8        Martin Van Buren         168
8       9  William Henry Harrison         173
9      10              John Tyler         183
10     11           James K. Polk         173
11     12          Zachary Taylor         173
12     13        Millard Fillmore         175
13     14         Franklin Pierce         178
14     15          James Buchanan         183
15     16         Abraham Lincoln         193
16     17          Andrew Johnson         178
17     18        Ulysses S. Grant         173
18     19     Rutherford B. Hayes         174
19     20       James A. Garfield         183
20     21       Chester A. Arthur         183
21     23       Benjamin Harrison         168
22     25        William McKinley         170
23     26      Theodore Roosevelt         178
24     27     William Howard Taft         182
25     28          Woodrow Wilson         180
26     29       Warren G. Harding         183
27     30         Calvin Coolidge         178
28     31          Herbert Hoover         182
29     32   Franklin D. Roosevelt         188
30     33         Harry S. Truman         175
31     34    Dwight D. Eisenhower         179
32     35         John F. Kennedy         183
33     36       Lyndon B. Johnson         193
34     37           Richard Nixon         182
35     38             Gerald Ford         183
36     39            Jimmy Carter         177
37     40           Ronald Reagan         185
38     41       George H. W. Bush         188
39     42            Bill Clinton         188
40     43          George W. Bush         182
41     44            Barack Obama         185
```

```
print(data['height(cm)'])
```

```
0     189
1     170
2     189
3     163
4     183
5     171
6     185
7     168
8     173
9     183
10    173
11    173
12    175
13    178
14    183
15    193
16    178
17    173
18    174
19    183
20    183
21    168
22    170
23    178
24    182
25    180
26    183
27    178
28    182
29    188
30    175
31    179
32    183
33    193
34    182
35    183
36    177
37    185
```

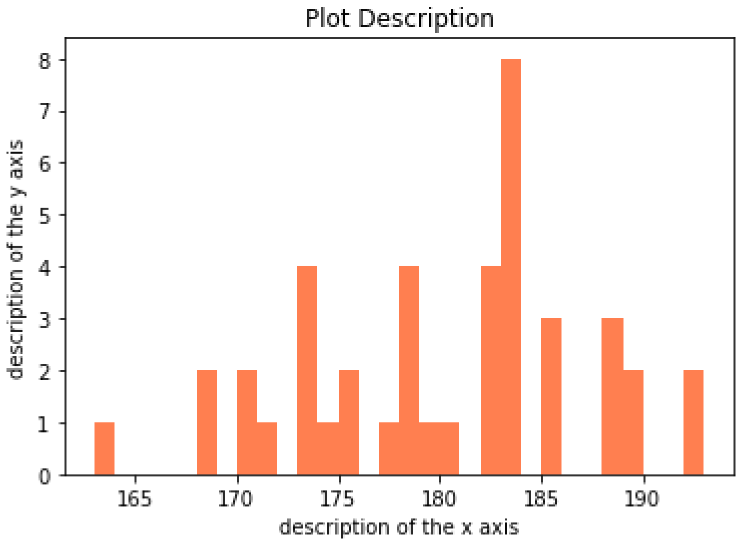
```
    38    188
    39    188
    40    182
    41    185
    Name: height(cm), dtype: int64
```

```
print('wartość średnia: ', data['height(cm)'].mean())
print('odchylenie standardowe: ', data['height(cm)'].std())
print('mediana: ', data['height(cm)'].median())
print('minimum: ', data['height(cm)'].min())
print('maximum: ', data['height(cm)'].max())
```

```
    wartość średnia:  179.73809523809524
    odchylenie standardowe:  7.015868855358296
    mediana:  182.0
    minimum:  163
    maximum:  193
```

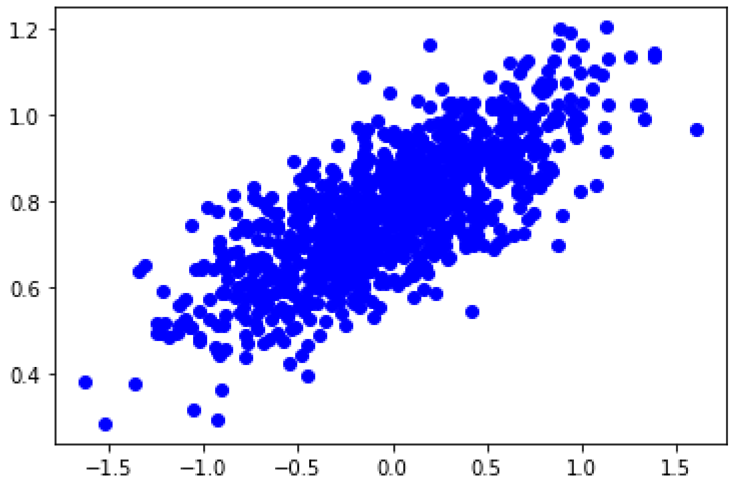
```
import matplotlib.pyplot as plt
```

```
heights = data['height(cm)']
plt.hist(heights, 30, color = 'coral')
plt.title('Plot Description')
plt.xlabel('description of the x axis')
plt.ylabel('description of the y axis')
plt.show()
```



zad.5

```
#Generacja zbioru
number_of_points = 1000
x_point = []
y_point = []

a = 0.22
b = 0.78


for i in range(number_of_points):
  #np.random.normal(środek rozkładu normal, odchylenie standardowe)
  x = np.random.normal(0.0, 0.5)
  y = (a*x + b) + np.random.normal(0.0, 0.1)
  x_point.append(x)
  y_point.append(y)


plt.scatter(x_point, y_point, c = 'b')
plt.show()
```



```
x = np.array(x_point)
y = np.array(y_point)
```

```
#
data = np.array([x_point, y_point])
df = pd.DataFrame(data, columns = ["x", "y"])
df.to_csv('data.csv')
#

print(np.sum(x))
print(np.sum(y))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/usr/local/lib/python3.6/dist-packages/pandas/core/internals/managers.py in create_block_manager_from_blocks(blocks, axes)
   1670                 blocks = [
-> 1671                     make_block(values=blocks[0], placement=slice(0, len(axes[0])))
   1672                 ]

                                5 frames
ValueError: Wrong number of items passed 1000, placement implies 2

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
/usr/local/lib/python3.6/dist-packages/pandas/core/internals/managers.py in create_block_manager_from_blocks(blocks, axes)
   1679         blocks = [getattr(b, "values", b) for b in blocks]
   1680         tot_items = sum(b.shape[0] for b in blocks)
-> 1681         raise construction_error(tot_items, blocks[0].shape[1:], axes, e)
   1682 
   1683 

ValueError: Shape of passed values is (2, 1000), indices imply (2, 2)
```

SEARCH STACK OVERFLOW

```
x = np.sum(x*x)
x
```

```
262.84634319976504
```

```
data = np.array([x_point, y_point])
data = data.reshape(-1, 2)
print(data)
```

```
[[-0.78481817  0.38948041]
 [ 0.77505964  0.67257364]
 [-0.01431895 -0.35930979]
 ...
 [ 0.77003532  0.70588316]
 [ 0.56271199  0.58978019]
 [ 0.88082487  0.66810146]]
```

```
df = pd.DataFrame(data, columns = ["X", "Y"])
print(df)
```

```
            X         Y
0   -0.784818  0.389480
1    0.775060  0.672574
2   -0.014319 -0.359310
3   -0.020112 -1.034988
4    1.248690  0.451209
..        ...       ...
995  0.677250  1.013516
996  0.605540  1.071027
997  0.770035  0.705883
998  0.562712  0.589780
999  0.880825  0.668101

[1000 rows x 2 columns]
```

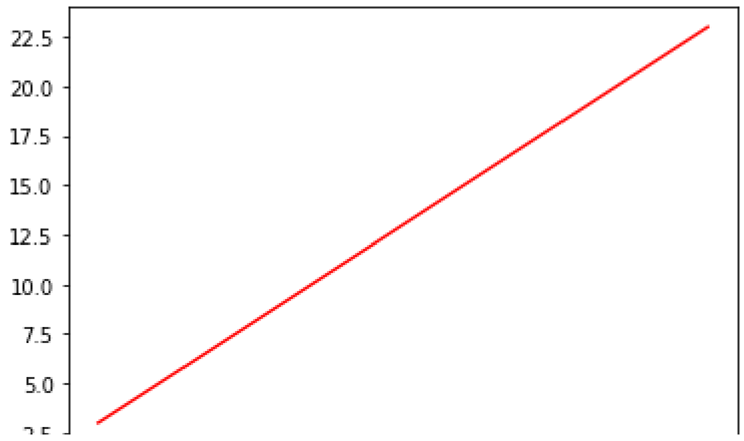
```
df.to_csv('dane.csv')
```

zad.6

```
X = np.linspace(0, 10, num = 10)
X
```

```
array([ 0.        ,  1.11111111,  2.22222222,  3.33333333,  4.44444444,
        5.55555556,  6.66666667,  7.77777778,  8.88888889, 10.        ])
```

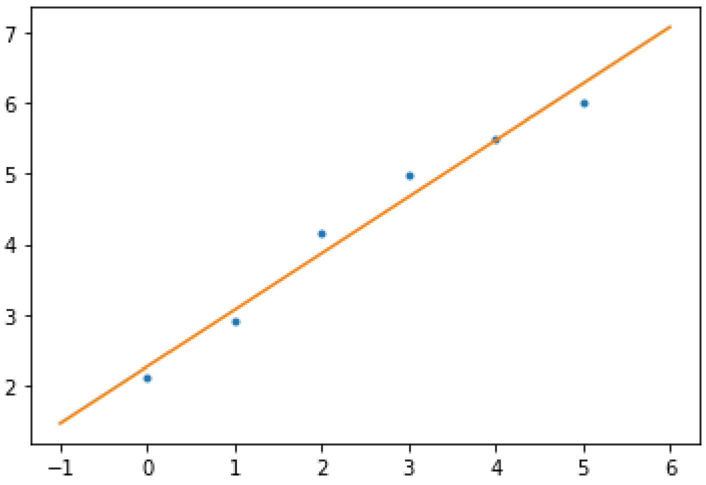
```
plt.plot(X, 2*X+3, C='R')
plt.show()
```



```
print("")
```

MNK Example:

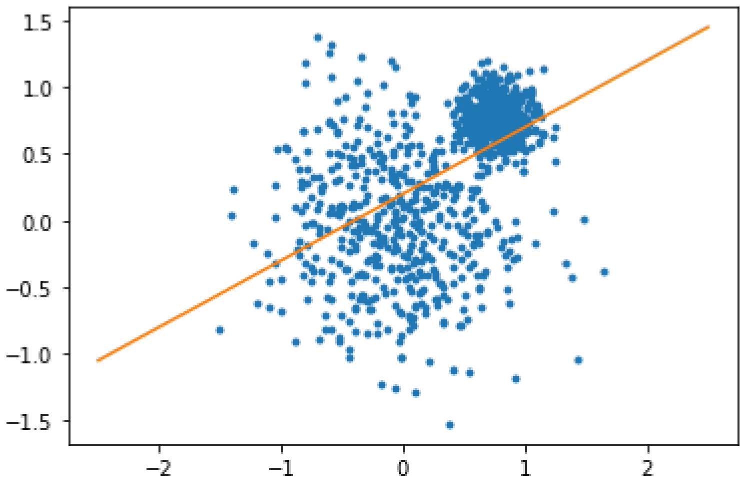
```
import pandas as pd
import numpy as np
data = pd.read_csv('dane.csv')


print(data.X)

     0     -0.784818
     1      0.775060
     2     -0.014319
     3     -0.020112
     4      1.248690
              ...
     995    0.677250
     996    0.605540
     997    0.770035
     998    0.562712
     999    0.880825
     Name: X, Length: 1000, dtype: float64


x_test = np.array(data.X)
#print(x_test)


#EXAMPLE
x = np.array([0,1,2,3,4,5])
y = np.array([2.1, 2.9, 4.15, 4.98, 5.5, 6])

z = np.polyfit(x, y, 1)
p = np.poly1d(z)

#plotting
import matplotlib.pyplot as plt
xp = np.linspace(-1, 6, 100)
plt.plot(x, y, '.', xp, p(xp))
plt.show()
```



```
     0     -0.784818
     1      0.775060
     2     -0.014319
     3     -0.020112
     4      1.248690
              ...
     995    0.677250
     996    0.605540
     997    0.770035
     998    0.562712
     999    0.880825
     Name: X, Length: 1000, dtype: float64
```

MNK (short):

```
data = pd.read_csv('dane.csv')

x = np.array(data.X)
y = np.array(data.Y)

z = np.polyfit(x, y, 1)
p = np.poly1d(z)

#plotting
import matplotlib.pyplot as plt
xp = np.linspace(-2.5, 2.5, 200)
plt.plot(x, y, '.', xp, p(xp))
plt.show()
```



MNK:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive',force_remount=True)

import os
os.chdir('/content/drive/My Drive/DM')
```

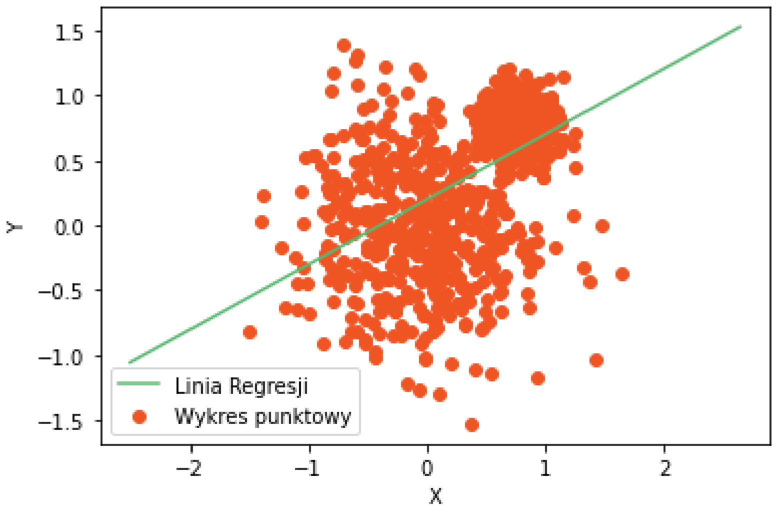
```
     Mounted at /content/drive
```

```
data = pd.read_csv('dane.csv')

# Wczytywanie liczb z kolumn X i Y do tablicy
X = data['X'].values
Y = data['Y'].values

# Wartoci średnie X i Y
mean_x = np.mean(X)
mean_y = np.mean(Y)

# Całkowita liczba wartości
n = len(X)

# Obliczenie "a" i "b"
numer = 0
denom = 0
for i in range(n):
  numer = numer + (X[i] - mean_x) * (Y[i] - mean_y)
  denom = denom + (X[i] - mean_x) ** 2
a = numer / denom
b = mean_y - (a * mean_x)

# Wyświetlanie współczynników
print("Współczynniki:")
print('a: ', a, '\nb: ', b)
```

```
     Współczynniki:
     a:  0.501013503612495
     b:  0.19956768562938146
```

```
# Wykres z wartościami i linią regresji
max_x = np.max(X) + 1
min_x = np.min(X) - 1

# Obliczanie wartości linii x i y
x = np.linspace(min_x, max_x, 100)
y = a * x + b
```

```
# Linia
plt.plot(x, y, color='#58b970', label='Linia Regresji')
# Punkty rozproszenia
plt.scatter(X, Y, c='#ef5423', label='Wykres punktowy')

plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```



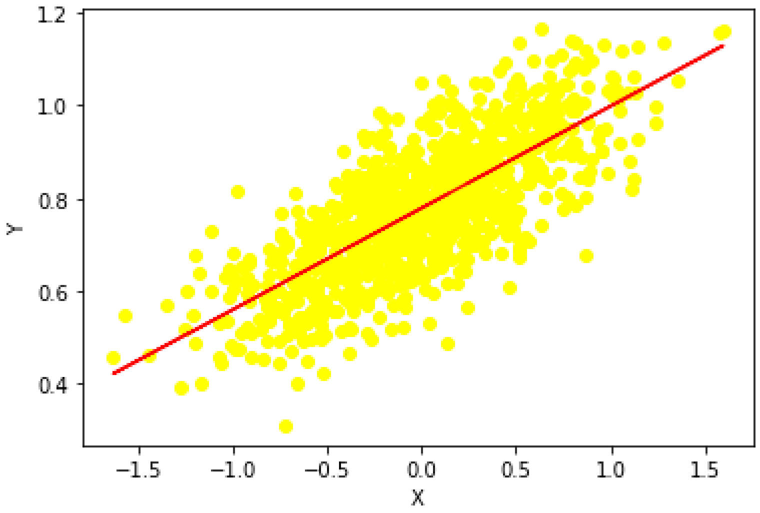
zad.6

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


"""
#Generacja zbioru
number_of_points = 1000
x_arr = []
y_arr = []

A = 0.22
B = 0.78

for i in range(number_of_points):
  #np.random.normal(środek rozkładu normal, odchylenie standardowe)
  x_local = np.random.normal(0.0, 0.5)
  y_local = (A*x_local + B) + np.random.normal(0.0, 0.1)
  x_arr.append(x_local)
  y_arr.append(y_local)

data = np.array([x_arr, y_arr])
data = data.reshape(-1, 2)
print(data)
df = pd.DataFrame(data, columns = ["x", "y"])
print(df)

df.to_csv('data.csv')
```

```
[[-0.03980865  0.07432214]
 [-0.13512663  0.61916132]
 [-0.273509    1.03295717]
 ...
 [ 1.11410718  0.96268307]
 [ 0.59842613  0.94993351]
 [ 0.71461461  0.60880334]]
            x         y
0   -0.039809  0.074322
1   -0.135127  0.619161
2   -0.273509  1.032957
3    1.002881  0.181030
4   -0.566037  0.045430
..        ...       ...
995  1.023641  0.689182
996  0.687501  0.868513
997  1.114107  0.962683
998  0.598426  0.949934
999  0.714615  0.608803

[1000 rows x 2 columns]
```

```
#Generacja zbioru
number_of_points = 1000
x_arr = []
y_arr = []
```

```
a = 0.22
b = 0.78

for i in range(number_of_points):
  x_local = np.random.normal(0.0, 0.5)
  y_local = (a*x_local + b) + np.random.normal(0.0, 0.1)
  x_arr.append(x_local)
  y_arr.append(y_local)

x = np.array(x_arr)
y = np.array(y_arr)

m1 = np.sum(x*x)
m2 = np.sum(x)
ms = x.size
m3 = np.sum(x*y)
m4 = np.sum(y)

mx = np.array(((m1,  m2), (m2, ms)))
mod = np.linalg.inv(mx)
mx2 = np.array((m3, m4))
mx2.shape = (2, 1)
m = mod.dot(mx2)
a = m[0][0]
b = m[1][0]

plt.plot(x, a*x + b, color = "red")
plt.scatter(x, y, c = "yellow")
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```


1.2
1.0
0.8
0.6
0.4
Y
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
X