```
import numpy as np
```

Zad 1

```
A = np.array([[2,-3,1],[4,5,0],[2,-1,3]])
```

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

```
C = np.array([[2,4],[-2,1],[5,0]])
```

```
D = np.array([[3],[6],[8]])
```

```
B1= B.reshape(3,1) #najprawdopodobniej błąd w zadaniu
```

```
print(B1)
```

```
     [[ 3]
      [-4]
      [-2]]
```

```
print(A*B)
#ValueError: operands could not be broadcast together with shapes (3,3) (1,4)
```

```
     [[  6  12  -2]
      [ 12 -20   0]
      [  6   4  -6]]
```

```
print(A*B1)
#ValueError: operands could not be broadcast together with shapes (3,3) (4,1)
```

```
     [[  6  -9   3]
      [-16 -20   0]
      [ -4   2  -6]]
```

```
#@print(A*C)
#ValueError: operands could not be broadcast together with shapes (3,3) (3,2)
```

```
print(A*D)
```

```
     [[ 6 -9  3]
      [24 30  0]
      [16 -8 24]]
```

```
print(B*A)
#ValueError: operands could not be broadcast together with shapes (1,4) (3,3)
```

```
     [[  6  12  -2]
      [ 12 -20   0]
      [  6   4  -6]]
```

```
print(B*B1)
```

```
     [[  9 -12  -6]
      [-12  16   8]
      [ -6   8   4]]
```

```
#print(B*C)
#ValueError: operands could not be broadcast together with shapes (1,3) (3,2)
```

```
print(B*D)
```

```
     [[  9 -12  -6]
      [ 18 -24 -12]
      [ 24 -32 -16]]
```

```
#print(C*A)
#ValueError: operands could not be broadcast together with shapes (3,2) (3,3)
```

```
#print(C*B)
#ValueError: operands could not be broadcast together with shapes (3,2) (1,3)
```

```
print(C*B1)
#ValueError: operands could not be broadcast together with shapes (3,2) (4,1)
```

```
[[  6  12]
 [  8  -4]
 [-10   0]]
```

```
print(C*D)
```

```
[[  6  12]
 [-12   6]
 [ 40   0]]
```

```
print(D*A)
```

```
[[ 6 -9  3]
 [24 30  0]
 [16 -8 24]]
```

```
print(D*B)
```

```
[[  9 -12  -6]
 [ 18 -24 -12]
 [ 24 -32 -16]]
```

```
print(D*B1)
#alueError: operands could not be broadcast together with shapes (3,2) (4,1)
```

```
[[  9]
 [-24]
 [-16]]
```

```
print(D*C)
```

```
[[  6  12]
 [-12   6]
 [ 40   0]]
```

Np.matmul

```
#print(np.matmul(A,B))
#ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 1 is diff
```

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

```
[[16]
 [-8]
 [ 4]]
```

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

```
[[15  5]
 [-2 21]
 [21  7]]
```

```
print(np.matmul(A,D))
```

```
[[-4]
 [42]
 [24]]
```

```
print(np.matmul(B,A))
```

```
[[-14 -27  -3]]
```

```
print(np.matmul(B,B1))
```

```
[[29]]
```

```
print(np.matmul(B,C))
```

```
[[4 8]]
```

```
print(np.matmul(B,D))
```

```
[[-31]]
```

```
#print(np.matmul(C,A))
#ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is diff
```

```
#print(np.matmul(C,B1))
#ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is diff
```

```
#print(np.matmul(C,D))
#ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is diff
```

```
#print(np.matmul(D,A))
#ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is diff
```

```
print(np.matmul(D,B))
```

```
[[  9 -12  -6]
 [ 18 -24 -12]
 [ 24 -32 -16]]
```

```
#print(np.matmul(D,B1))
#ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is diff
```

```
#print(np.matmul(D,C))
#ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is diff
```

Sprawdź czy operacja np.dot(x,y) jest tożsama z operacją np.dot np.matmul

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

```
[[15  5]
 [-2 21]
 [21  7]]
```

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

```
[[15  5]
 [-2 21]
 [21  7]]
```

W przypadku której macierzy możliwe jest znalezienie macierzy odwrotnej? Znajdź tę wartość wykorzystując odpowiednią operację z numpy.linalg

```
np.linalg.inv(A)
```

```
array([[ 0.28846154,  0.15384615, -0.09615385],
       [-0.23076923,  0.07692308,  0.07692308],
       [-0.26923077, -0.07692308,  0.42307692]])
```

```
try:
  if(np.linalg.inv(B)):
   print("Da sie znalezc odwrotnej B")
   print(np.linalg.inv(B))
except:
    print("nie da się")
```

```
nie da się
```

```
try:
  if(np.linalg.inv(B1)):
   print("Da sie znalezc odwrotnej B1")
   print(np.linalg.inv(B1))
except:
    print("nie da się")
```

```
nie da się
```

```
try:
  if(np.linalg.inv(C)):
   print("Da sie znalezc odwrotnej c")
   print(np.linalg.inv(C))
except:
    print("nie da się")
```

```
nie da się
```

```
try:
  if(np.linalg.inv(D)):
   print("Da sie znalezc odwrotnej D")
   print(np.linalg.inv(D))
except:
    print("nie da się")
```

```
nie da się
```

```
np.sum(A,axis=0)
```

```
array([8, 1, 4])
```

```
np.sum(A,axis=1)
```

```
array([0, 9, 4])
```

```
np.sum(B,axis=0)
```

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

```
np.sum(B,axis=1)
```

```
array([-3])
```

```
np.sum(B1,axis=0)
```

```
array([-3])
```

```
np.sum(B1,axis=1)
```

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

```
np.sum(C,axis=0)
```

```
array([5, 5])
```

```
np.sum(C,axis=1)
```

```
array([ 6, -1,  5])
```

```
np.sum(D,axis=0)
```

```
array([17])
```

```
np.sum(D,axis=1)
```

```
array([3, 6, 8])
```

```
np.sum(A)
```

```
13
```

```
np.sum(B)
```

```
-3
```

```
np.sum(B1)
```

```
-3
```

```
np.sum(C)

     10
```

```
np.sum(D)

     17
```

Zadanie 2

```
D = np.array(np.arange(0,12))
D

     array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
D2=D.reshape(3,2,2)
print(D2)

     [[[ 0  1]
       [ 2  3]]

      [[ 4  5]
       [ 6  7]]

      [[ 8  9]
       [10 11]]]
```

```
D3=D.reshape(6,2,1)
print(D3)

     [[[ 0]
       [ 1]]

      [[ 2]
       [ 3]]

      [[ 4]
       [ 5]]

      [[ 6]
       [ 7]]

      [[ 8]
       [ 9]]

      [[10]
       [11]]]
```

```
D4=D.reshape(3, 4, 1)
print(D4)



     [[[ 0]
       [ 1]
       [ 2]
       [ 3]]

      [[ 4]
       [ 5]
       [ 6]
       [ 7]]

      [[ 8]
       [ 9]
       [10]
       [11]]]
```

```
#np.matmul(D,D2)
#ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 2 is diff
```

```
#np.matmul(D,D3)
#ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 2 is diff
```

```
#np.matmul(D,D4)
#ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 4 is diff
```

```
np.matmul(D,D3.reshape(12,1))
```

```
array([506])
```

Zadanie 3

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

```
s_copy = data.copy()
print(s_copy)
```

```
   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
```

```
S1=pd.DataFrame(data=s_copy, index=[1],copy=True)
print(S1)
```

```
   X   B  C  D  E
1  2  11 -4  7 -2
```

```
S2=pd.DataFrame(data=s_copy, index= [1,2],copy=True)
print(S2)
```

```
   X   B  C  D  E
1  2  11 -4  7 -2
2  3  21  8 -2  9
```

```
S3=pd.DataFrame(data=s_copy, index= [2,3],columns=['B','C','D'],copy=True)
print(S3)
```

```
    B   C  D
2  21   8 -2
3   4  12  1
```

```
S4=pd.DataFrame(data=s_copy,columns=['B','D'],copy=True)
print(S4)
```

```
    B  D
0  12  5
1  11  7
2  21 -2
3   4  1
```

Zadanie 4

```
data2 = pd.read_csv('president_heights.csv')
print(data2)
```

```
    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
```

```
P1=pd.DataFrame(data=data2,columns=['height(cm)'],copy=True)
print(P1)
```

```
        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
```

```
M=P1.mean()
print("Mean height")
print(M)
```

```
    Mean height
    height(cm)    179.738095
    dtype: float64
```

```
Std=P1.std()
print("Std")
print(Std)
```

```
    Std
    height(cm)    7.015869
    dtype: float64
```

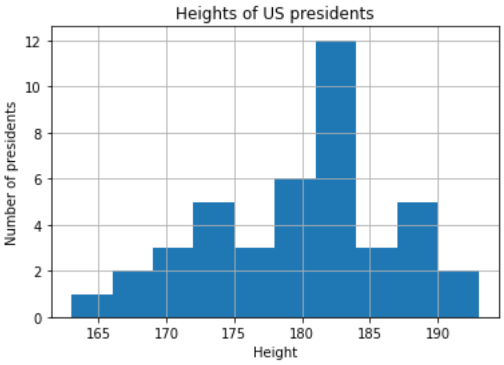
```
Min=P1.min()
Max=P1.max()
print("Min")
print(Min)
print("Max")
print(Max)
```

```
Min
height(cm)    163
dtype: int64
Max
height(cm)    193
dtype: int64
```

```
Median=P1.median()
print("median")
print(Median)
```

```
median
height(cm)    182.0
dtype: float64
```

```
import matplotlib.pyplot as plt
heights=P1.hist(xlabelsize=10)
#plt.hist(heights,10,color='red')
plt.title('Heights of US presidents')
plt.xlabel('Height')
plt.ylabel('Number of presidents')
plt.show()
```



Zadanie 5

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

```
tab=np.random.normal(size=(2,1000))
```

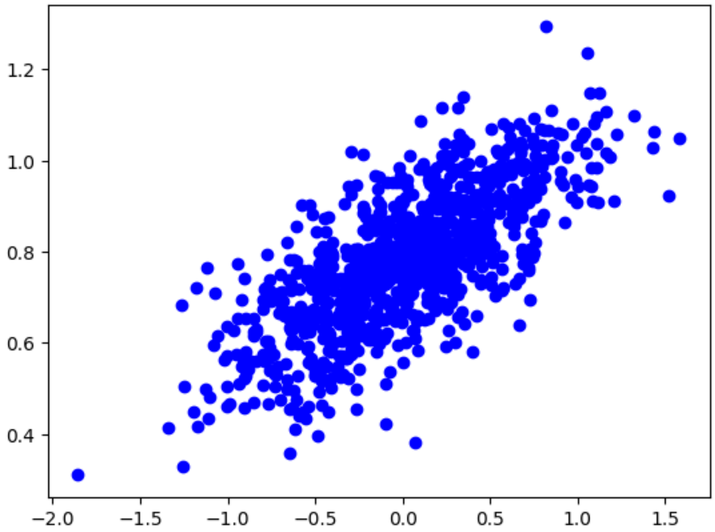
```
plt.scatter(tab[0],tab[1])
plt.show()
```

```
number_of_points=1000
x_point = []
y_point = []
x_1=[]
y_1=[]
```

```
a=0.22
b=0.78
```

```
for i in range(number_of_points):
  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)
  x_1.append(1)
  y_1.append(1)


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



Zadanie 6

```
x_s=np.array(x_point)
y_s=np.array(y_point)


x2=np.sum(x_s*x_s)
x1=np.sum(x_s)
x1sum=np.sum(x_1)
y1sum=np.sum(y_1)
xy=np.sum(x_s*y_s)
y1=np.sum(y_s)



print(x1)

     12.474799844036689


M = np.array([[x2,x1],[x1,x1sum]])
print(M)

     [[ 243.61496026   12.47479984]
      [  12.47479984 1000.        ]]


M_1=np.linalg.inv(M)
print(M_1)

     [[ 4.10746206e-03 -5.12397670e-05]
      [-5.12397670e-05  1.00063921e-03]]
```

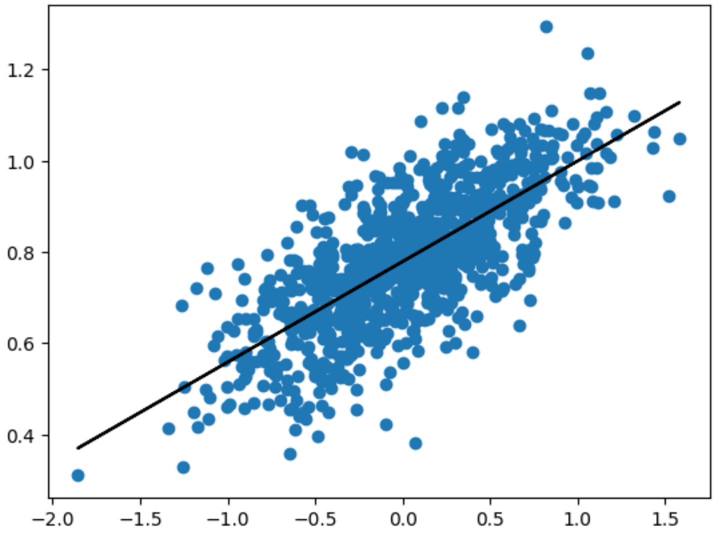
```
N=np.array([[xy],[y1]])
print(N)
```

```
[[ 63.38886088]
 [781.75156167]]
```

```
a,b = np.matmul(M_1,N)
print(a,b)
```

```
[0.22031057] [0.77900323]
```

```
plt.scatter(x_s,y_s)
plt.plot(x_s,a*x_s+b,"black")
plt.show()
```



Zadanie 7

```
import pandas as pd
```

```
california_cities = pd.read_csv('california_cities.csv')
```

```
california_cities.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 482 entries, 0 to 481
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         482 non-null    int64  
 1   city               482 non-null    object 
 2   latd               482 non-null    float64
 3   longd              482 non-null    float64
 4   elevation_m        434 non-null    float64
 5   elevation_ft       470 non-null    float64
 6   population_total   482 non-null    int64  
 7   area_total_sq_mi   480 non-null    float64
 8   area_land_sq_mi    482 non-null    float64
 9   area_water_sq_mi   481 non-null    float64
 10  area_total_km2     477 non-null    float64
 11  area_land_km2      478 non-null    float64
 12  area_water_km2     478 non-null    float64
 13  area_water_percent  477 non-null    float64
dtypes: float64(11), int64(2), object(1)
memory usage: 52.8+ KB
```

```
print(california_cities)
```

```
     Unnamed: 0         city       latd       longd  elevation_m  \
0             0     Adelanto  34.576111 -117.432778        875.0   
1             1  AgouraHills  34.153333 -118.761667        281.0   
2             2      Alameda  37.756111 -122.274444          NaN   
3             3       Albany  37.886944 -122.297778          NaN   
4             4     Alhambra  34.081944 -118.135000        150.0   
..          ...          ...        ...         ...          ...   
477         477   Yountville  38.403056 -122.362222         30.0   
478         478        Yreka  41.726667 -122.637500        787.0   
```

```
479        479     YubaCity  39.134722 -121.626111         18.0
480        480      Yucaipa  34.030278 -117.048611        798.0
481        481  YuccaValley  34.133333 -116.416667       1027.0

     elevation_ft  population_total  area_total_sq_mi  area_land_sq_mi  \
0          2871.0             31765            56.027           56.009
1           922.0             20330             7.822            7.793
2            33.0             75467            22.960           10.611
3            43.0             18969             5.465            1.788
4           492.0             83089             7.632            7.631
..            ...               ...               ...              ...
477          98.0              2933             1.531            1.531
478        2582.0              7765            10.053            9.980
479          59.0             64925            14.656           14.578
480        2618.0             51367            27.893           27.888
481        3369.0             20700            40.015           40.015

     area_water_sq_mi  area_total_km2  area_land_km2  area_water_km2  \
0               0.018         145.107        145.062           0.046
1               0.029          20.260         20.184           0.076
2              12.349          59.465         27.482          31.983
3               3.677          14.155          4.632           9.524
4               0.001          19.766         19.763           0.003
..                ...             ...            ...             ...
477             0.000           3.966          3.966           0.000
478             0.073          26.036         25.847           0.188
479             0.078          37.959         37.758           0.201
480             0.005          72.244         72.231           0.013
481             0.000         103.639        103.639           0.000

     area_water_percent
0                  0.03
1                  0.37
2                 53.79
3                 67.28
4                  0.01
..                  ...
477                0.00
478                0.72
479                0.53
480                0.02
481                0.00

[482 rows x 14 columns]
```

```
PT =pd.DataFrame(california_cities,columns=['population_total'])
print(PT)
```

```
     population_total
0               31765
1               20330
2               75467
3               18969
4               83089
..                ...
477              2933
478              7765
479             64925
480             51367
481             20700

[482 rows x 1 columns]
```

```
ATSQM =pd.DataFrame(california_cities,columns=['area_total_sq_mi'])
print(ATSQM)
```

```
     area_total_sq_mi
0              56.027
1               7.822
2              22.960
3               5.465
4               7.632
..                ...
477             1.531
478            10.053
479            14.656
480            27.893
481            40.015

[482 rows x 1 columns]
```

```
ALSQM =pd.DataFrame(california_cities,columns=['area_land_sq_mi'])
print(ALSQM)
```

```
     area_land_sq_mi
0             56.009
1              7.793
```

```
2             10.611
3              1.788
4              7.631
..               ...
477            1.531
478            9.980
479           14.578
480           27.888
481           40.015

[482 rows x 1 columns]
```

```
import numpy.ma as ma
```

```
np.corrcoef(PT,ATSQM)
```

```
array([[nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       ...,
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan]])
```

```
np.corrcoef(ATSQM,rowvar = False)
```

```
nan
```

```
np.corrcoef(ALSQM,rowvar = False)
```

```
1.0
```

```
corr_matrix = california_cities.corr()
```

```
<ipython-input-228-cc55c4f78f74>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  corr_matrix = california_cities.corr()
```

```
corr_matrix["area_land_sq_mi"].sort_values(ascending=False)
```

```
area_land_sq_mi       1.000000
area_land_km2         0.999993
area_total_km2        0.993643
area_total_sq_mi      0.967577
population_total      0.849758
area_water_km2        0.466967
area_water_sq_mi      0.256239
longd                 0.177034
elevation_m           0.079316
elevation_ft          0.074864
Unnamed: 0            0.038934
area_water_percent   -0.017031
latd                 -0.152656
Name: area_land_sq_mi, dtype: float64
```

```
corr_matrix["population_total"].sort_values(ascending=False)
```

```
population_total      1.000000
area_total_sq_mi      0.864089
area_total_km2        0.861592
area_land_km2         0.856184
area_land_sq_mi       0.849758
area_water_km2        0.485096
area_water_sq_mi      0.377493
longd                 0.081605
area_water_percent    0.046492
Unnamed: 0            0.041386
elevation_m          -0.058003
elevation_ft         -0.067929
latd                 -0.109800
Name: population_total, dtype: float64
```

```
corr_matrix["area_land_sq_mi"].sort_values(ascending=False)
```

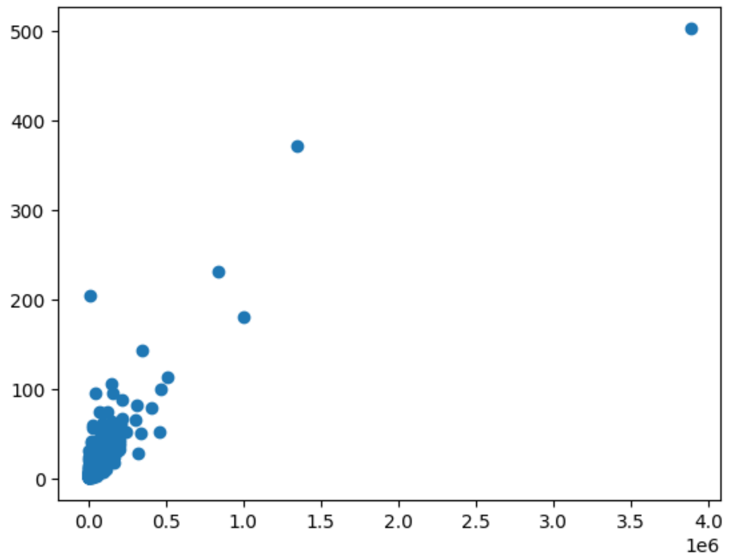
```
area_land_sq_mi       1.000000
area_land_km2         0.999993
area_total_km2        0.993643
area_total_sq_mi      0.967577
population_total      0.849758
area_water_km2        0.466967
area_water_sq_mi      0.256239
longd                 0.177034
```

```
    elevation_m           0.079316
    elevation_ft          0.074864
    Unnamed: 0            0.038934
    area_water_percent   -0.017031
    latd                 -0.152656
    Name: area_land_sq_mi, dtype: float64
```

```
print(california_cities.corr()["area_land_sq_mi"])
```

```
    Unnamed: 0            0.038934
    latd                 -0.152656
    longd                 0.177034
    elevation_m           0.079316
    elevation_ft          0.074864
    population_total      0.849758
    area_total_sq_mi      0.967577
    area_land_sq_mi       1.000000
    area_water_sq_mi      0.256239
    area_total_km2        0.993643
    area_land_km2         0.999993
    area_water_km2        0.466967
    area_water_percent   -0.017031
    Name: area_land_sq_mi, dtype: float64
    <ipython-input-237-38a412272fb0>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
      print(california_cities.corr()["area_land_sq_mi"])
```

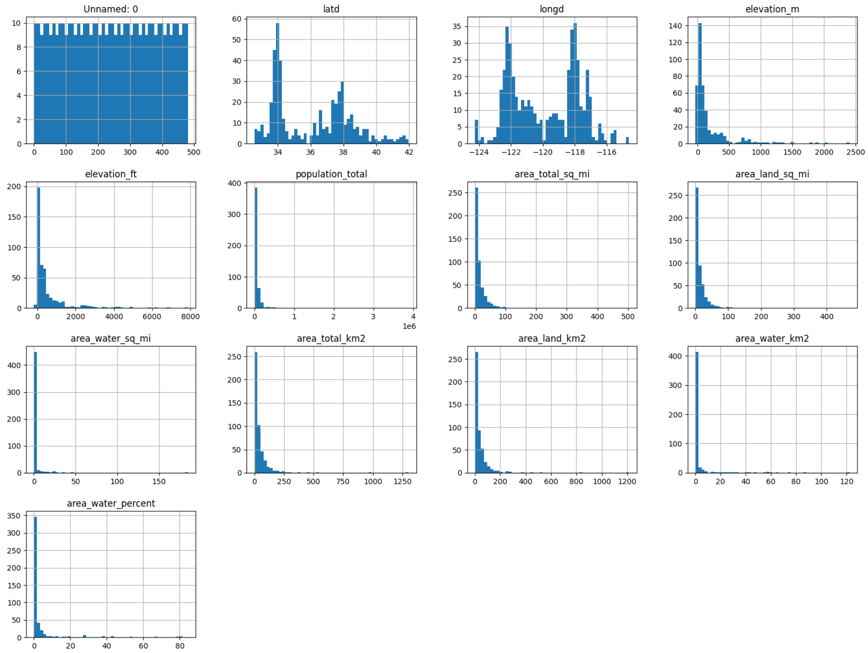
```
plt.scatter(PT,ATSQM)
plt.show()
```



```
%matplotlib inline # Wyłącznie w notatniku Jupyter
```

```
    UsageError: unrecognized arguments: # Wyłącznie w notatniku Jupyter
```

```
import matplotlib.pyplot as plt
california_cities.hist(bins=50, figsize=(20,15))
plt.show()
```


Unnamed: 0
0 2 4 6 8 10
0 100 200 300 400 500
latd
0 10 20 30 40 50 60
34 36 38 40 42
longd
0 5 10 15 20 25 30 35
−124 −122 −120 −118 −116
elevation_m
0 20 40 60 80 100 120 140
0 500 1000 1500 2000 2500
elevation_ft
0 50 100 150 200
0 2000 4000 6000 8000
population_total
0 100 200 300 400
0 1 2 3 4
1e6
area_total_sq_mi
0 50 100 150 200 250
0 100 200 300 400 500
area_land_sq_mi
0 50 100 150 200 250
0 100 200 300 400
area_water_sq_mi
0 100 200 300 400
0 50 100 150
area_total_km2
0 50 100 150 200 250
0 250 500 750 1000 1250
area_land_km2
0 50 100 150 200 250
0 200 400 600 800 1000 1200
area_water_km2
0 100 200 300 400
0 20 40 60 80 100 120
area_water_percent
0 50 100 150 200 250 300 350
0 20 40 60 80