```python
import numpy as np
import math
import matplotlib.pyplot as plt
import sklearn
import sklearn.metrics as met
from sklearn.preprocessing import StandardScaler

iris = []

def zad1():
    print('\nzad1')
    with open('iris.data') as file:
        for line in file:
            iris.append(line.rstrip().split(','))
    print(len(iris))
    print(len(iris[0]))
zad1()


def zad2():
    print('\nzad2')
    print(iris[9])
    print(iris[74])
    a = float(iris[9][0])
    b = float(iris[74][0])
    c = float(iris[9][1])
    d = float(iris[74][1])
    print(math.sqrt((a-b)**2+(c-d)**2))
#zad2()


def mimasrkw(x):
    print(min(x), max(x), np.mean(np.asarray(x, dtype=float)),
np.std(np.asarray(x, dtype=float)))

irisArray = []
irisArrayTrans = []

def zad3():
    print('\nzad3')
    global irisArray
    irisArray = np.array(iris)
    global irisArrayTrans
    irisArrayTrans = np.transpose(irisArray)
    mimasrkw(irisArrayTrans[0])
    mimasrkw(irisArrayTrans[1])
    mimasrkw(irisArrayTrans[2])
    mimasrkw(irisArrayTrans[3])
zad3()


def zad4():
    print('\nzad4')
    plt.scatter(np.asarray(irisArrayTrans[0], dtype=float),
np.asarray(irisArrayTrans[1],dtype=float))
    plt.xlabel('sepal length')
    plt.ylabel('sepal width')
    plt.show()
#zad4()
```

```
def zad5():
    print('\nzad5')
    for dana in iris:
        if dana[4] == 'Iris-setosa':
            kolor = 'yellow'
        else:
            if dana[4] == 'Iris-versicolor':
                kolor = 'blue'
            else:
                if dana[4] == 'Iris-virginica':
                    kolor = 'green'
                else:
                    kolor = 'black'
        plt.scatter(dana[0], dana[2], color = kolor)
    plt.xlabel('sepal length')
    plt.ylabel('petal length')
    plt.show()
#zad5()


#zad6
print('\nzad6')
def means(x):
    print([np.mean(np.asarray(x[0], dtype=float)),
np.mean(np.asarray(x[1], dtype=float)),  np.mean(np.asarray(x[2],
dtype=float)),  np.mean(np.asarray(x[3], dtype=float))])
setosaArray = []
versicolorArray = []
for i in range(len(irisArray)):
    if (irisArray[i][4] == "Iris-setosa"):
        setosaArray.append(irisArray[i])
    if (irisArray[i][4] == "Iris-versicolor"):
        versicolorArray.append(irisArray[i])
setosaArrayTrans = np.array(setosaArray).T
versicolorArrayTrans = np.array(versicolorArray).T
print("Setosa")
means(setosaArrayTrans)
print("Versicolor")
means(versicolorArrayTrans)
# zad6


#zad7
print('\nzad7')
irisArray2 = np.delete(irisArray, np.s_[4], axis=1)
standardizedData = StandardScaler().fit_transform(irisArray2)
standardizedDataTrans = np.array(standardizedData).T
print("slength:")
mimasrkw(standardizedDataTrans[0])
print("swidth:")
mimasrkw(standardizedDataTrans[1])
print("plength:")
mimasrkw(standardizedDataTrans[2])
print("pwidth:")
mimasrkw(standardizedDataTrans[3])
#zad7


#zad8
print('\nzad8')
s0 = np.array(np.split(np.random.normal(-2, 1, 10), 1))
```

```python
print(s0)
s1 = np.array(np.split(np.random.uniform(0, 10, 10), 1))
print(s1)
array = np.concatenate((s0.T, s1.T), axis=1)
plt.scatter(np.asarray(array[0], dtype=float),  np.asarray(array[1],
dtype=float))
#plt.show()
#zad8


#zad9
print('\nzad9')
euclDist=met.pairwise.pairwise_distances(array, metric='euclidean')
print(euclDist)
mahDist=met.pairwise.pairwise_distances(array, metric='mahalanobis')
print(mahDist)
minkDist=met.pairwise.pairwise_distances(array, metric='minkowski')
print(minkDist)
#zad9


#zad10
print('\nzad10')
scaling = sklearn.preprocessing.MinMaxScaler((0,1))
scal = scaling.fit_transform(array)
euclDist=met.pairwise.pairwise_distances(scal, metric='euclidean')
print(euclDist)
mahDist=met.pairwise.pairwise_distances(scal, metric='mahalanobis')
print(mahDist)
minkDist=met.pairwise.pairwise_distances(scal, metric='minkowski')
print(minkDist)
#zad10


#zad11
print('\nzad11')
def f1(x):
    return -x[0] + x[1]
def f2(x):
    return x[0] - x[1]
# Powierzchnia decyzyjna: x[0]=-x[1]
#  Classificator
def classify(x):
    if f1(x) > f2(x):
        return 1
    else:
        return 2
#zad11


#zad12
print('\nzad12')
x1 = np.array(np.split(np.random.uniform(1, 10, 10), 1))
x2 = np.array(np.split(np.random.uniform(-10, -1, 10), 1))
data = np.array(np.concatenate((np.concatenate((x1.T, x2.T), axis=1),
np.concatenate((x2.T, x1.T), axis=1)), axis=0))
# randomize order in array
np.random.shuffle(data)
print(data)
labels = np.array([classify(data[i]) for i in range(len(data))])
print('Classifications for data:')
print(labels)
#zad12
```