In [1]:
```
from keras.datasets import mnist;
(train_images, train_labels), (test_images, test_labels) = mnist.load_dat
print( train_images.shape )
print( len(train_labels) )
print( train_labels )
print( train_images[0])
```

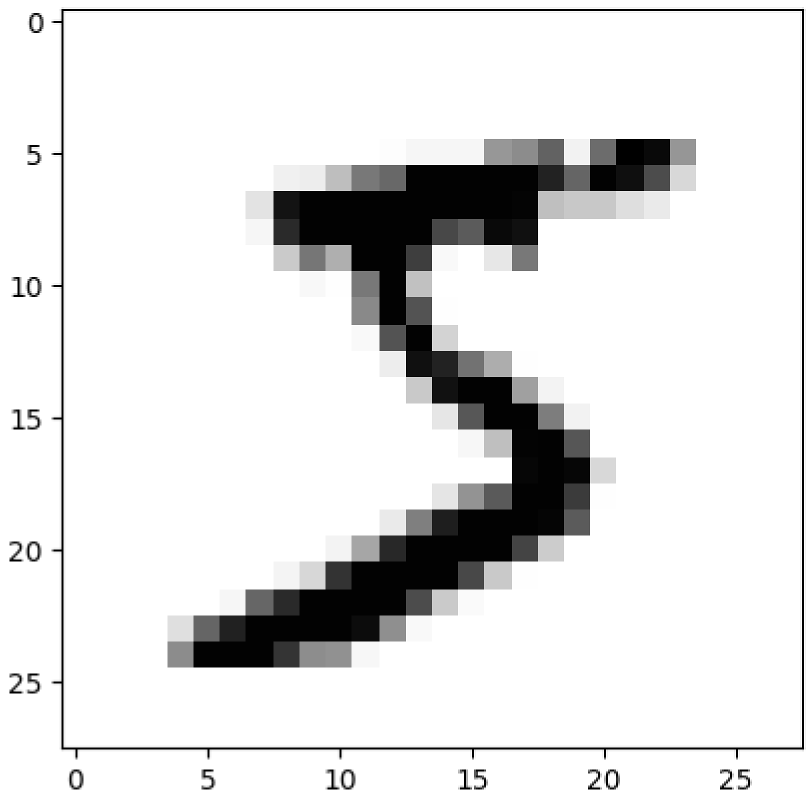
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
(60000, 28, 28)
60000
[5 0 4 ... 5 6 8]
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
```

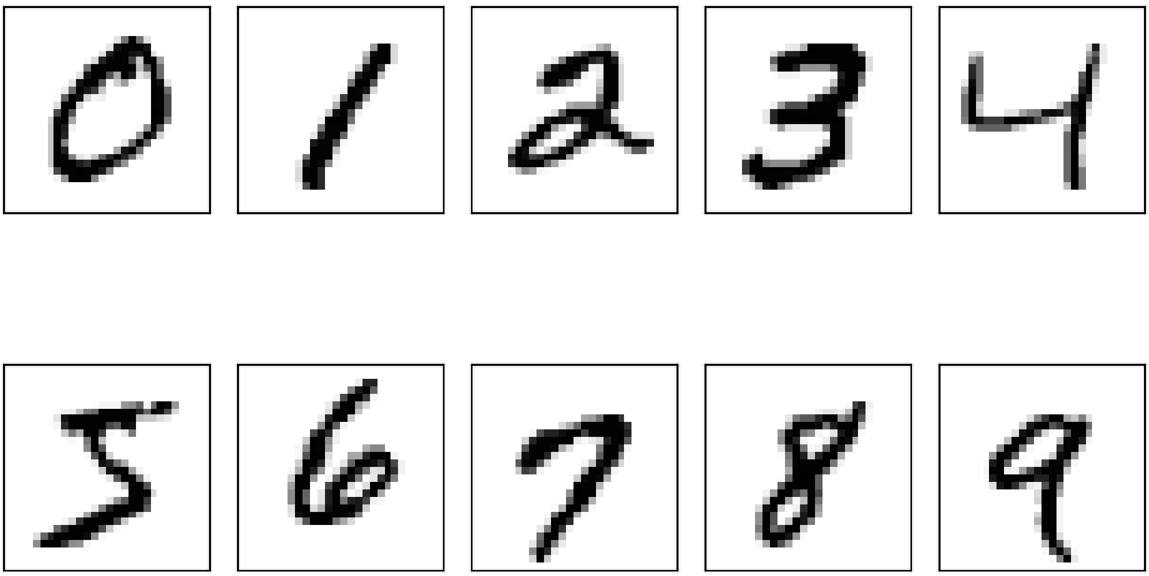
```
[  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0]]
```

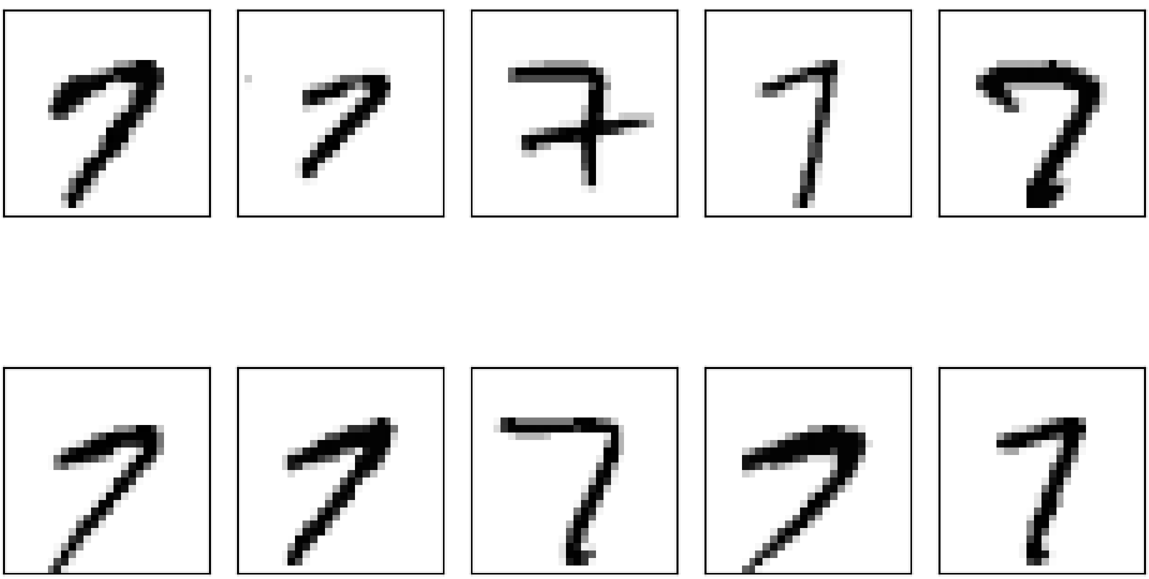
In [2]:

```
import matplotlib.pyplot as plt
img = train_images[0].reshape(28,28)
plt.imshow( img, cmap='Greys')
print("Liczba ", train_labels[0])


fig, ax = plt.subplots(nrows=2, ncols=5, sharex=True, sharey=True)
ax = ax.flatten()
for i in range(10):
 img = train_images[ train_labels==i ][0].reshape(28,28)
 ax[i].imshow( img, cmap='Greys')
ax[0].set_xticks([])
ax[0].set_yticks([])
plt.tight_layout()
plt.show()


fig, ax = plt.subplots(nrows=2, ncols=5, sharex=True, sharey=True)
ax = ax.flatten()
for i in range(10):
 img = train_images[ train_labels==7 ][i].reshape(28,28)
 ax[i].imshow( img, cmap='Greys')
ax[0].set_xticks([])
ax[0].set_yticks([])
plt.tight_layout()
plt.show()

x_train = train_images.reshape((60000, 28*28))
x_train = x_train.astype('float32') / 256
x_test = test_images.reshape((10000, 28*28))
x_test = x_test.astype('float32') / 256
print( train_images[0])
print( x_train[0])
```

```
Liczba  5
```


0
5
10
15
20
25
0
5
10
15
20
25





```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
```

```
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.01171875 0.0703125  0.0703125  0.0703125
0.4921875  0.53125    0.68359375 0.1015625  0.6484375  0.99609375
0.96484375 0.49609375 0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.1171875  0.140625   0.3671875  0.6015625
0.6640625  0.98828125 0.98828125 0.98828125 0.98828125 0.98828125
0.87890625 0.671875   0.98828125 0.9453125  0.76171875 0.25
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.19140625
0.9296875  0.98828125 0.98828125 0.98828125 0.98828125 0.98828125
0.98828125 0.98828125 0.98828125 0.98046875 0.36328125 0.3203125
0.3203125  0.21875    0.15234375 0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.0703125  0.85546875 0.98828125
0.98828125 0.98828125 0.98828125 0.98828125 0.7734375  0.7109375
0.96484375 0.94140625 0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.3125     0.609375   0.41796875 0.98828125
0.98828125 0.80078125 0.04296875 0.         0.16796875 0.6015625
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.0546875  0.00390625 0.6015625  0.98828125 0.3515625
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.54296875 0.98828125 0.7421875  0.0078125  0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.04296875
0.7421875  0.98828125 0.2734375  0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.13671875 0.94140625
0.87890625 0.625      0.421875   0.00390625 0.         0.
```

```
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.31640625 0.9375     0.98828125
0.98828125 0.46484375 0.09765625 0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.17578125 0.7265625  0.98828125 0.98828125
0.5859375  0.10546875 0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.0625     0.36328125 0.984375   0.98828125 0.73046875
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.97265625 0.98828125 0.97265625 0.25       0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.1796875  0.5078125  0.71484375 0.98828125
0.98828125 0.80859375 0.0078125  0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.15234375 0.578125
0.89453125 0.98828125 0.98828125 0.98828125 0.9765625  0.7109375
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.09375    0.4453125  0.86328125 0.98828125 0.98828125 0.98828125
0.98828125 0.78515625 0.3046875  0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.08984375 0.2578125  0.83203125 0.98828125
0.98828125 0.98828125 0.98828125 0.7734375  0.31640625 0.0078125
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.0703125  0.66796875
0.85546875 0.98828125 0.98828125 0.98828125 0.98828125 0.76171875
0.3125     0.03515625 0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.21484375 0.671875   0.8828125  0.98828125 0.98828125 0.98828125
0.98828125 0.953125   0.51953125 0.04296875 0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.53125    0.98828125
0.98828125 0.98828125 0.828125   0.52734375 0.515625   0.0625
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
```

```
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.        ]
```

In [3]:

```
from keras.utils import to_categorical
y_train = to_categorical(train_labels)
y_test = to_categorical(test_labels)

print( train_labels[0] )
print( y_train[0] )
```

```
5
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

# Zadanie nr 1:

In [4]:

```
from keras import models
from keras import layers
network = models.Sequential()
network.add(layers.Input(shape=(28*28,)))
#kolejne warstwy sieci Dense
#...
network.add(layers.Dense(10, activation="relu"))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
```

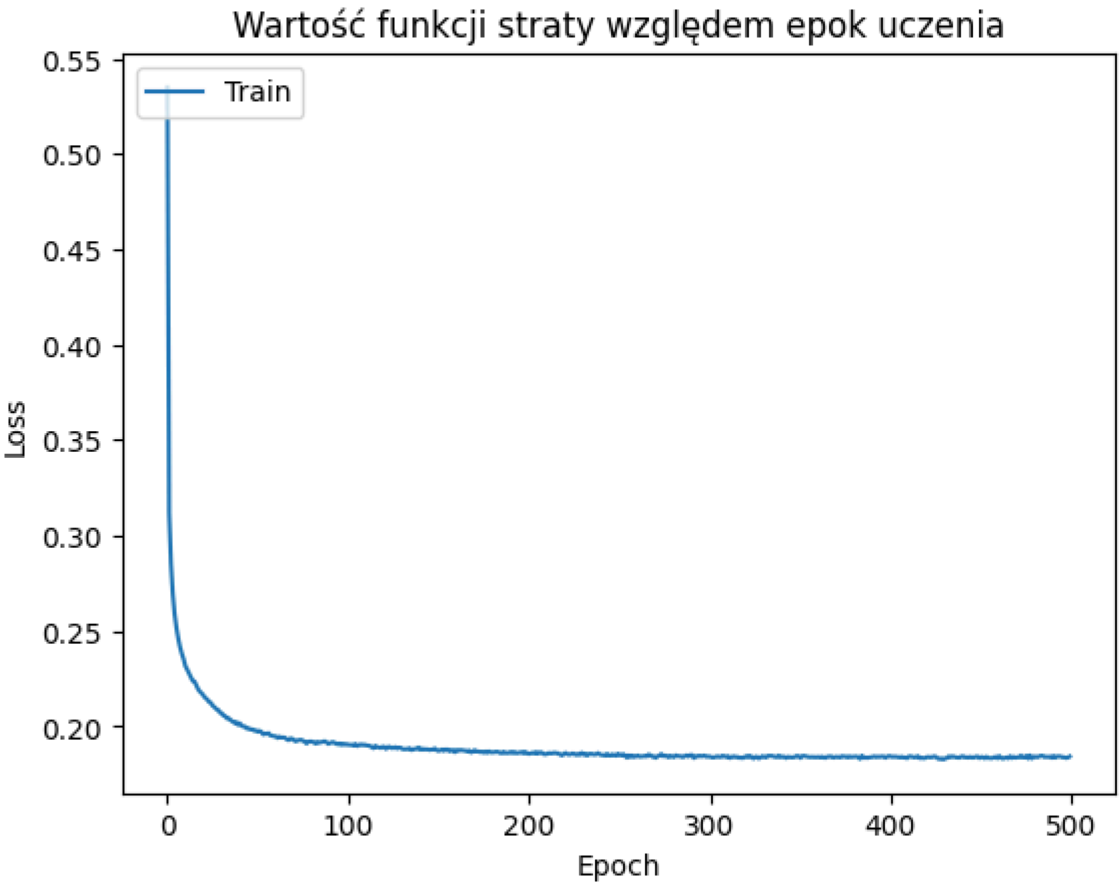
# Zadanie nr 2:

In [5]:

```
import copy
from sklearn.metrics import accuracy_score
def getIndexOfMax(arr):
  index = 0;
  for i in range(len(arr)):
    if arr[i] > arr[index]:
      index = i
    #elif (i != index) and (arr[i] == arr[index]):
      #raise Exception("Cannot get max element. Elements aren't uniqe.")
  return index
def maxto1restto0(_mat):
    mat = copy.deepcopy(_mat)
    for arr in mat:
      index = getIndexOfMax(arr)
      for i in range(len(arr)):
        if i == index:
          arr[i] = 1.0;
        else:
          arr[i] = 0.0;
    return mat

history = network.fit(x_train, y_train, epochs=500, batch_size=32, verbos
```

```
print("Ostatni błąd:", history.history['loss'][-1])
plt.plot(history.history['loss'])
plt.title('Wartość funkcji straty względem epok uczenia')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train'], loc='upper left')
plt.show()

#treningowe
y_result_train = network.predict(x_train)
y_result_train_rounded = maxto1restto0(y_result_train)
accuracy = accuracy_score(y_train, y_result_train_rounded)
print("Accuracy:",accuracy)
print("Wyniki obliczeń:\n", y_result_train[0],"\nZaokrąglone:\n",y_result
```

```
Ostatni błąd: 0.18411201238632202
```




```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step
Accuracy: 0.9544666666666667
Wyniki obliczeń:
 [1.1338093e-09 1.7666850e-12 4.8749328e-08 1.4026439e-02 5.2114424e-29
 9.8597360e-01 3.3910630e-12 1.3519860e-08 2.1042602e-10 1.2197202e-12]
Zaokrąglone:
 [[0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]]
```

# Zadanie nr 3

In [20]:
```
import numpy as np
def hitPercentage(_x, _y, model, label):
  predict_x = model.predict(_x)
  y_result = np.argmax(predict_x,axis=1)
  count = 0
  goodCount = 0
  for i in range(len(_y)):
    if(y_result[i] == _y[i]):
      goodCount += 1
    count += 1
  print(label,(goodCount/count)*100,"%")
hitPercentage(x_train, train_labels, network, "Zbiór treningowy:")
hitPercentage(x_test, test_labels, network, "Zbiór testowy:")
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step
Zbiór treningowy: 95.44666666666667 %
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Zbiór testowy: 92.88 %
```

# Zadanie nr 4

In [34]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score
import numpy as np
def metrics(_x, _y, model,label):
  # Przewidywanie wyników
  y_pred = model.predict(_x)
  # Zaokrąglenie wyników przewidywań
  y_pred_rounded = np.argmax(y_pred, axis=1)
  # Obliczenie metryk
  accuracy = accuracy_score(_y, y_pred_rounded)
  precision = precision_score(_y, y_pred_rounded, average="macro")
  recall = recall_score(_y, y_pred_rounded, average="macro")
  # Uzyskanie macierzy pomyłek
  conf_matrix = confusion_matrix(_y, y_pred_rounded)
  # Wyświetlanie wyników
  print(label)
  print("Accuracy:", accuracy)
  print("Precision:", precision)
  print("Recall:", precision)
  print("Confusion Matrix:\n", conf_matrix)

metrics(x_train, train_labels, network, "Metriki i macierz pomyłek dla zb
metrics(x_test, test_labels, network, "Metrik i macierz pomyłek dla zbior
```

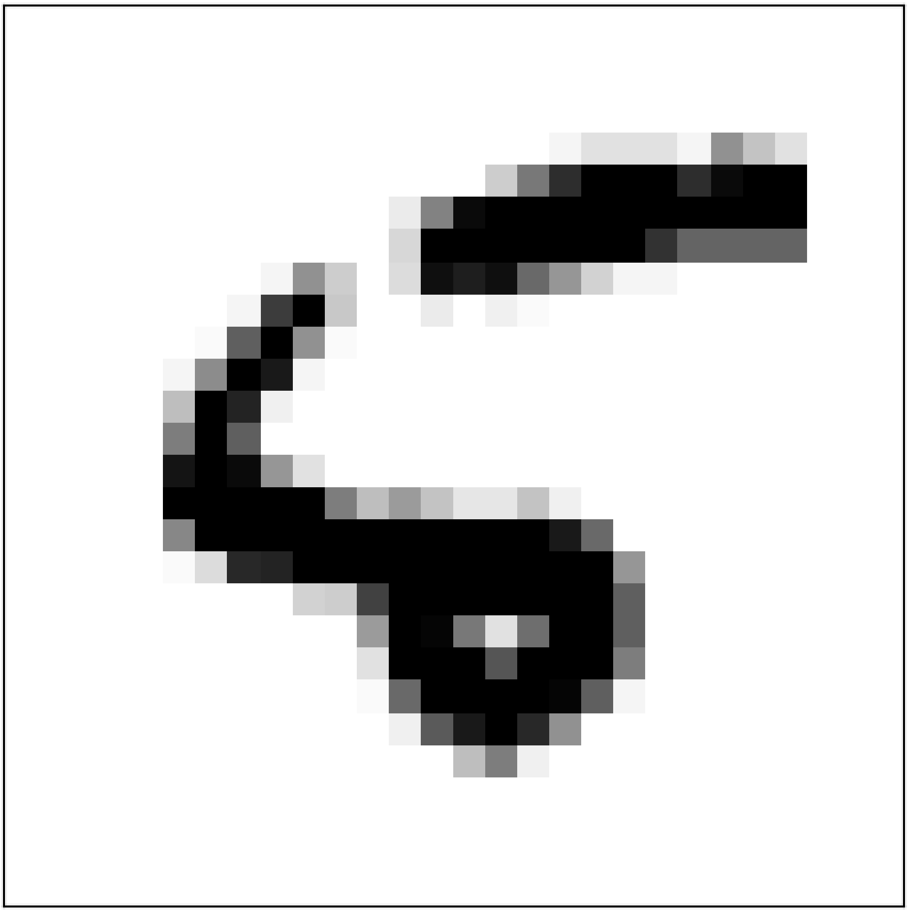
```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step
Metriki i macierz pomyłek dla zbioru treningowego:
Accuracy: 0.9544666666666667
Precision: 0.9542385739932907
Recall: 0.9542385739932907
Confusion Matrix:
 [[5812    1   12    4    1   13   31    5   41    3]
 [   1 6643   22    8    3    7    3   10   37    8]
 [  28   30 5632   64   28    8   50   31   79    8]
 [  20   23   87 5683    2  116   16   27  127   30]
 [  10   17   33    8 5553    6   67    8   39  101]
 [  31   31   15  102   13 5046   85    3   81   14]
 [  15    9    6    3   15   43 5805    3   18    1]
 [  21   19   20   23   30    5    5 6022   35   85]
 [  20   67   21   69   17   53   33    8 5549   14]
 [  22   11    6   55  108   35    7   96   86 5523]]
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Metrik i macierz pomyłek dla zbioru testowego:
Accuracy: 0.9288
Precision: 0.9282213884893157
Recall: 0.9282213884893157
Confusion Matrix:
 [[ 960    0    2    2    0    2    8    2    4    0]
 [   0 1110    4    2    2    2    5    0    9    1]
 [   9    9  930   15   12    7   16    8   26    0]
 [   2    4   16  920    2   26    3   10   24    3]
 [   1    2    9    7  898    4   17    7    3   34]
 [   5    3    2   24    5  810   17    1   18    7]
 [   9    4    7    1    3   10  918    2    4    0]
 [   2    9   14   14    5    2    0  947    8   27]
 [   5   13    7   14    9   14   10    9  888    5]
 [   8    6    0   11   26    8    0   25   18  907]]
```

# Zadanie nr 5

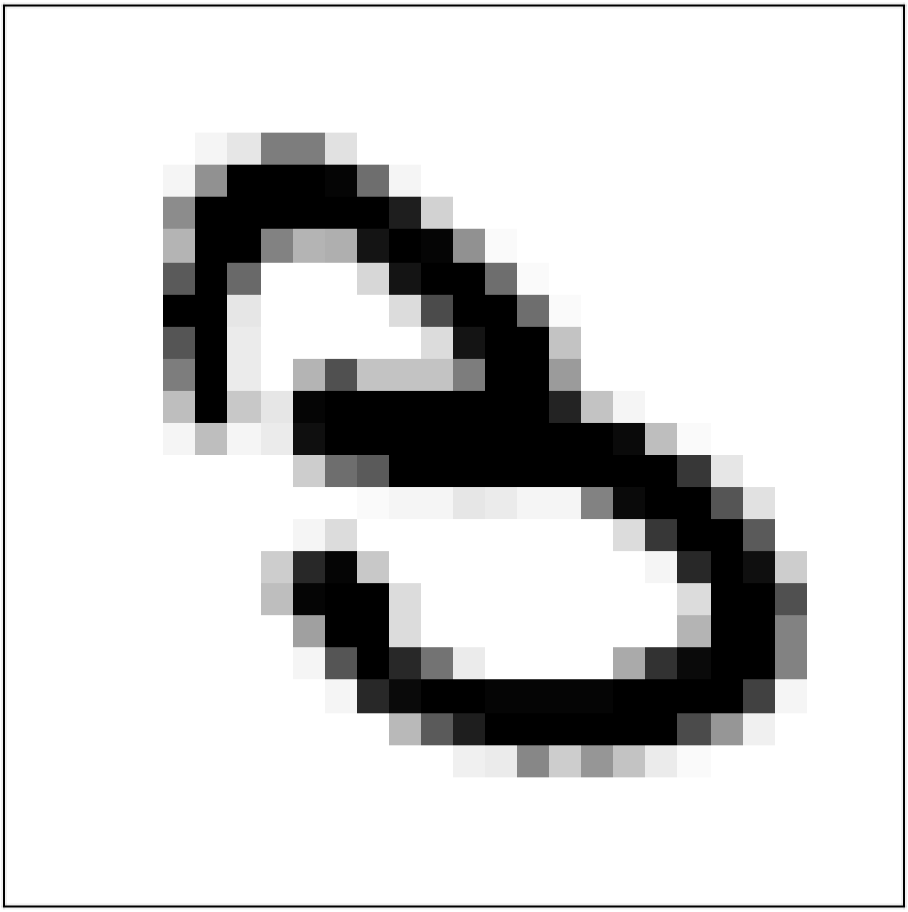
In [47]:

```python
badIndexes = []
y_pred = network.predict(x_test)
y_pred_rounded = np.argmax(y_pred, axis=1)
for i in range(len(y_pred_rounded)):
  if(y_pred_rounded[i] != test_labels[i]):
    badIndexes.append(i)
#otrzymaliśmy tablicę zawierającą indexy błednie sklasyfikowanych cyfr
#wyświetlenie błędnych 4 cyfr
len_badIndexes = len(badIndexes)
for i in range(4):
  if i < len_badIndexes:

    fig, ax = plt.subplots(nrows=1, ncols=1, sharex=True, sharey=True)
    img = test_images[badIndexes[i]].reshape(28,28)
    ax.imshow( img, cmap='Greys')
    ax.set_xticks([])
    ax.set_yticks([])
    plt.tight_layout()
    plt.show()
    print("Cyfra sklasyfikowana jako:",y_pred_rounded[badIndexes[i]])
    print("Poprawna klasyfikacja:", test_labels[badIndexes[i]])
```

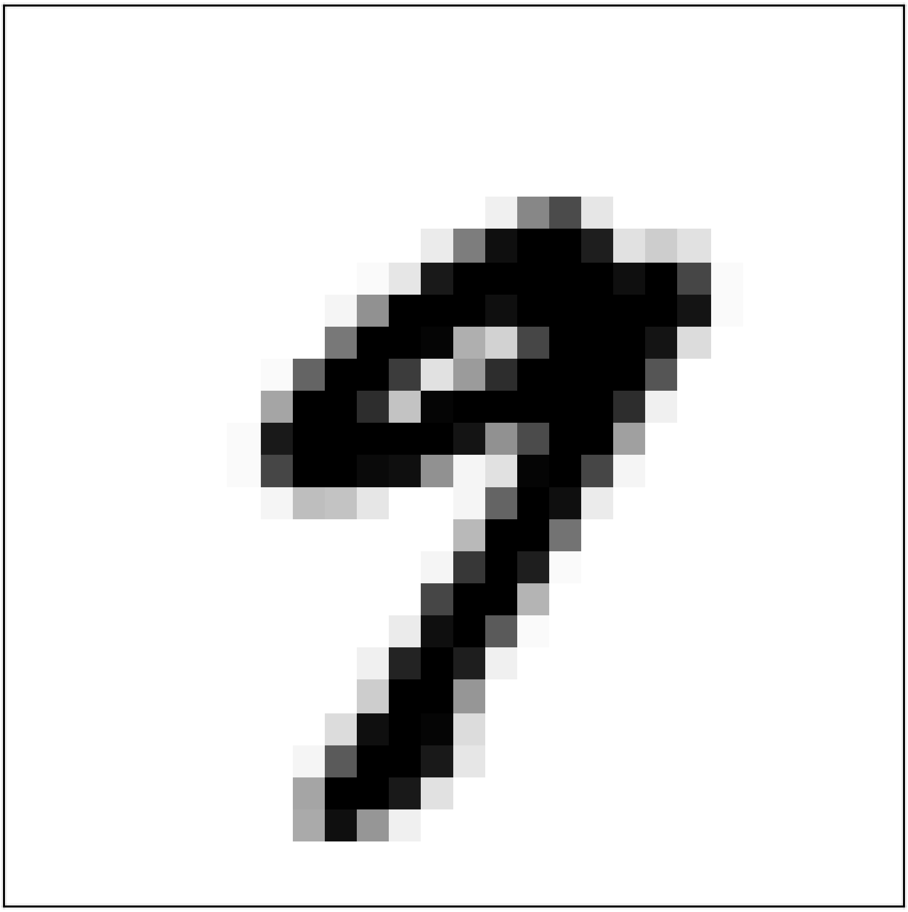
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```



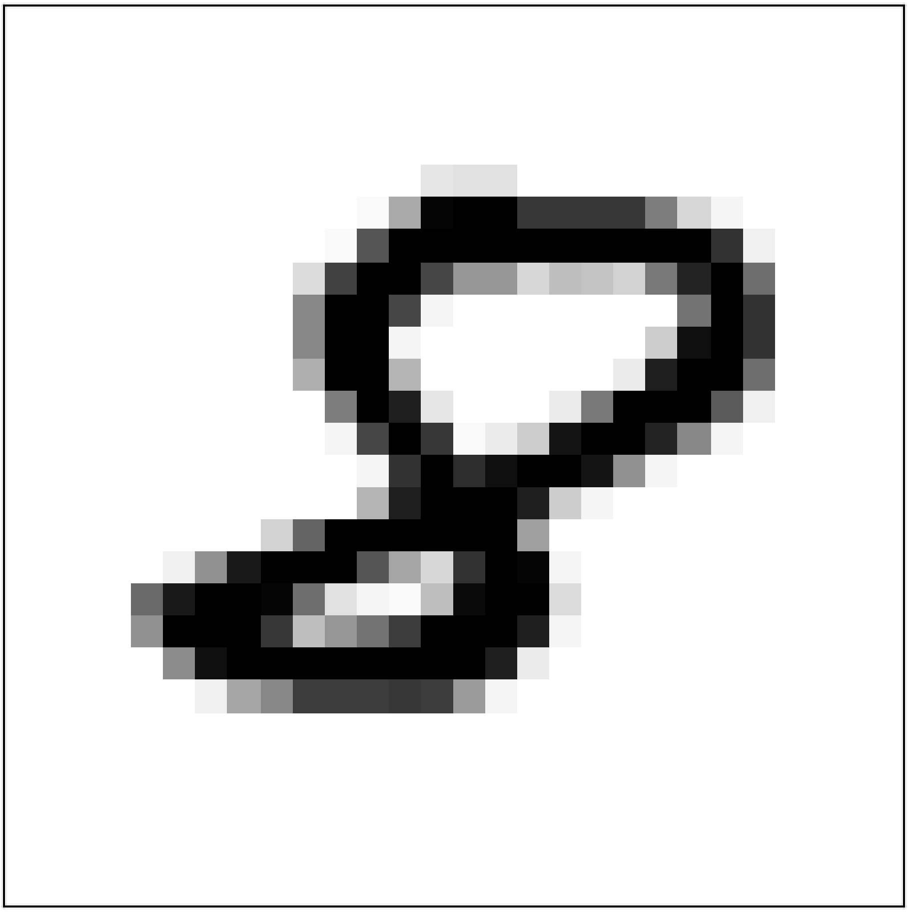
```
Cyfra sklasyfikowana jako: 6
Poprawna klasyfikacja: 5
```



```
Cyfra sklasyfikowana jako: 8
Poprawna klasyfikacja: 3
```



```
Cyfra sklasyfikowana jako: 7
Poprawna klasyfikacja: 9
```



```
Cyfra sklasyfikowana jako: 2
Poprawna klasyfikacja: 8
```

In [ ]:

```
```