```
import numpy as np
from mnist import load_mnist
import singlelayer as sn #SingleayerNetwork
from PIL import Image

np.random.seed(1)

def img_show(img): # (784) 혹은 (28, 28)의 mnist 배열을 이미지로 보여주는 작업
   if img.ndim==1:
      img = img.reshape(28, 28)
      img *= 255.0
      pil_img = Image.fromarray(np.uint8(img))
      pil_img.show()


def get_data(): # mnist 데이터 받아오기(one_hot_label로)
   (x_train, y_train), (x_test, y_test) = \
   load_mnist(normalize=True, flatten=True, one_hot_label=True)
   return x_train, y_train, x_test, y_test

def is_number(s):
   try:
      int(s)
      return True
   except ValueError:
      return False

def TestSN(input_i, x_train, y_train, x_test, y_test, W, Bias): #test 이미지를 하나 뽑고 singleNN.py의 trainingAndResult를 돌려서 학습전 결과와 학습 후 결과가 어떻게 다른지 확인하는 것
   if is_number(input_i):
      i = int(input_i)
      Test = x_train[i]
      label = np.argmax(y_train[i])
      img_show(Test)
      print("이 이미지의 실제 값 : ", label) #그림의 숫자와 동일
      SN = sn.singleLayer(W, Bias)
      y_predict = SN.ScoreFunction(x_train[i])
      print("이 이미지의 학습 전 이미지의 추론 값 : ", np.argmax(y_predict)) #추론값의 결과가 그림의 숫자와 같을 수도 다를 수도 있음.
      SN.Optimization(x_train, y_train, x_test, y_test)
      y_predict = SN.ScoreFunction(x_train[i])
      print("학습이 완료되었습니다 \n이미지의 학습 후 추론 값: ", np.argmax(y_predict)) #트레이닝 후의 추론값 또한 결과값과 다를 수도 있다.(정확도가 87% 정도이기에)
      return SN

   else:
      print("잘못 입력하셨습니다. 학습을 하지 않습니다.")
      return False
```

```
x_train, y_train, x_test, y_test = get_data()

# W값과 Bias 값을 np.random.random 함수를 사용해 알맞게 넣어주십시오.(이 것만 빈칸 나머지는 제공)
# 3.1
# Training Dataset의 shape가 (60000, 784)이고 결괏값(class)이 10개(0~9의 숫자)이기 때문에
# W의 shape는 (784, 10)이다. b의 shape는 broadcast되기 때문에 (1, 10)으로 잡는다.
# 784 = 28 * 28 (input image의 해상도 pixel)
# Score = X*W+b
# Training Dataset Input 기준 : (60000, 784)*(784, 10) + (1, 10)
W = np.random.random((784, 10))
Bias = np.random.random((1, 10))

#i = input() # 자신이 원하는 숫자 넣기 가능
i = 5
print("train 데이터의 {} 번째의 값 추출".format(i))

Trainend = TestSN(i, x_train, y_train, x_test, y_test, W, Bias) # 위의 TestNN함수를 호출해 작업을 돌림.

#밑에 것은 심심하면 자신이 트레이닝한 것이 잘되는지 실험해보세요.
'''
if Trainend !=False:
   TrainNN =Trainend
   print("몇 번 추론하실 겁니까?")
   iterator = input()
   if(is_number(iterator)):
      for i in range(0, int(iterator)):
         print("x_train의 s번째 데이터를 뽑아주세요.\n")
         s = int(input())
         print("S : {}".format(s))
         check = x_train[s]
         img_show(check)
         Hypothesis = TrainNN.Forward(check)
         print("이 이미지의 추론 값 : {}".format(np.argmax(Hypothesis)))
   else:
      print("iterator로 숫자를 안넣었습니다. 종료합니다.")

'''
```