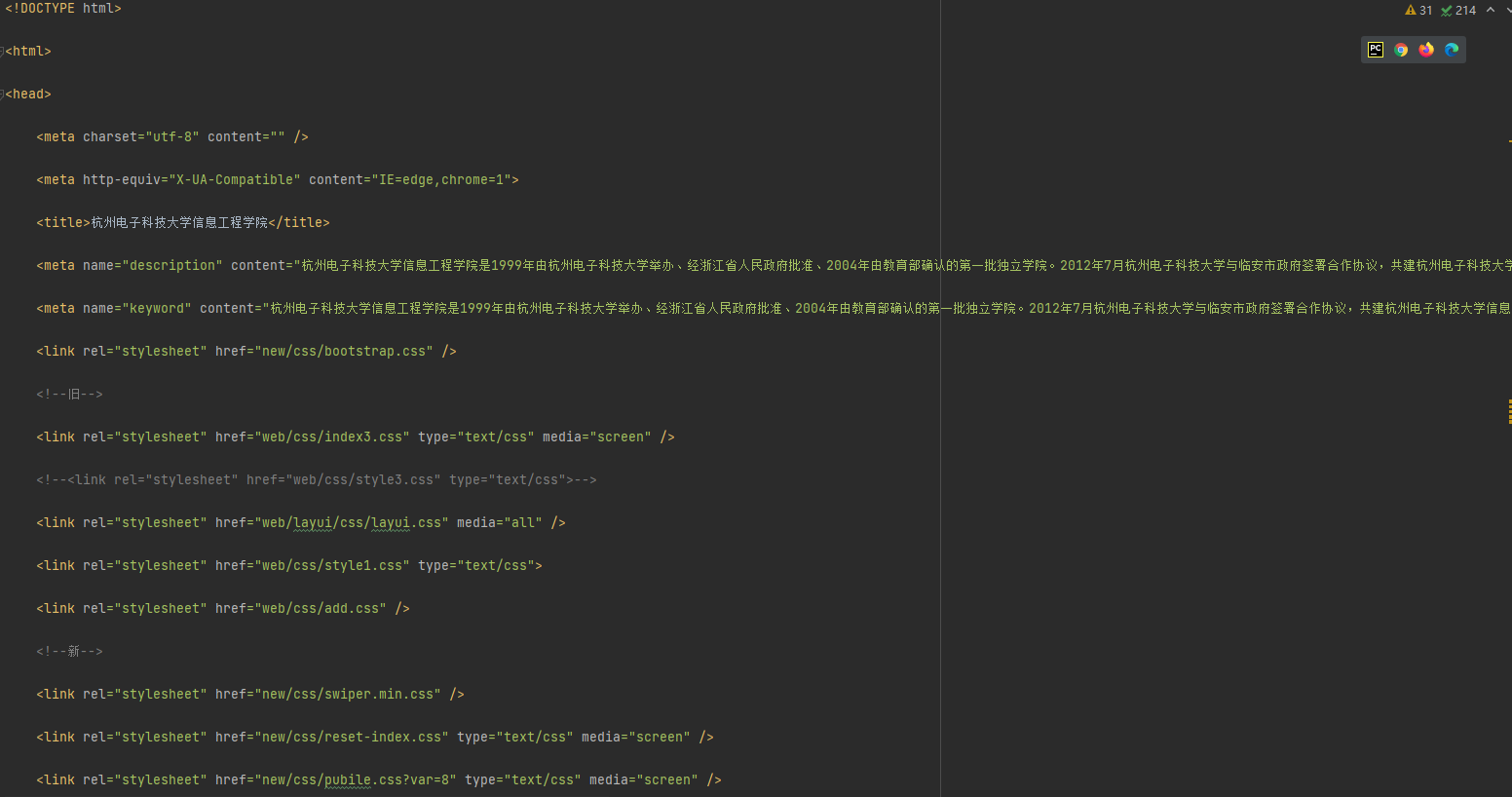
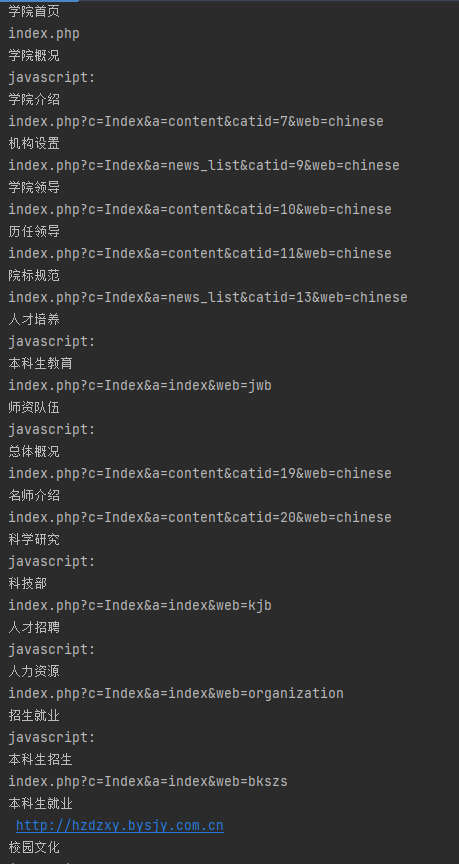
案例1界面

网页爬虫：爬取网页标题和链接

爬虫爬下来的网站内容







案例2界面

网页爬虫：抓取表格，做数据分析

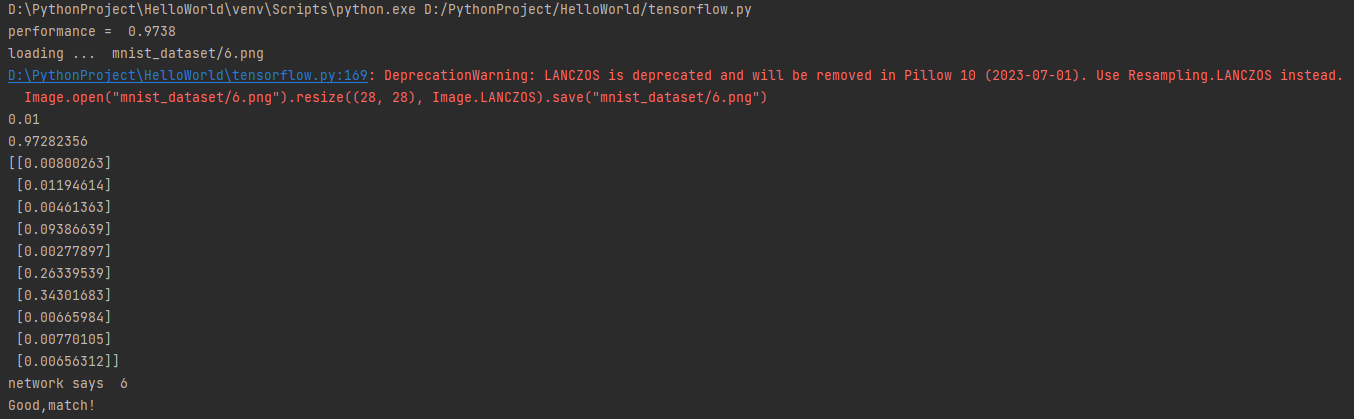


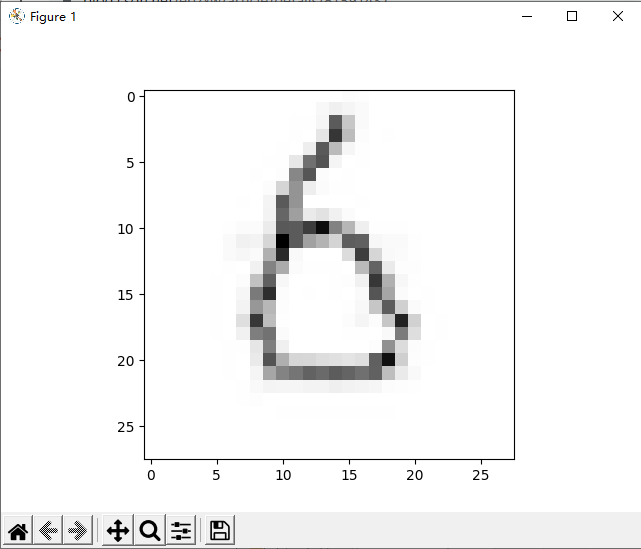


案例3界面

数字识别：

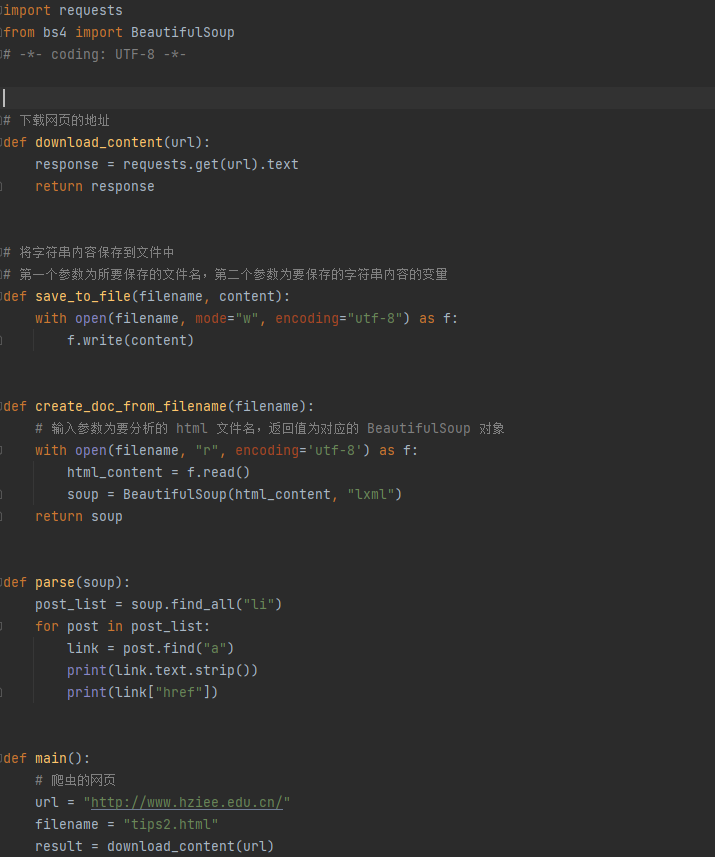
显示识别率以及数字识别的结果是否正确

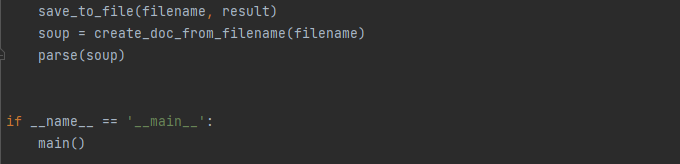




案例1代码

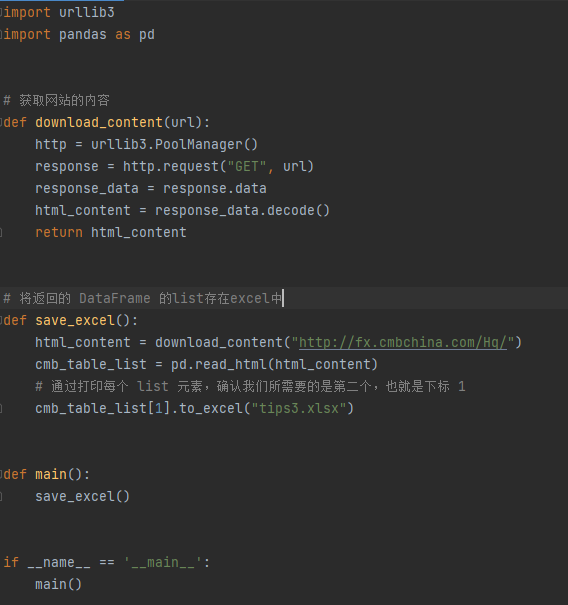
调用方法下载网页的内容并写入html文件，通过find\_all以及find方法获取想获取的标签名里的内容进行打印。





案例2代码

调用方法获取网页的内容，通过打印每个list元素，确认需要写入excel的表格内容。



案例3代码

使用mnist进行数字识别，通过mnist训练集进行数字识别训练，再通过mnist测试集测试数字识别的准确率，在测试手写数字的图片时，需要将手写数字的图片转换为28×28的像素，神经网络会显示输出值最大的数字。

# 数字识别  
import numpy  
import scipy.special  
import matplotlib.pyplot  
# from pasta.augment import inline  
# %matplotlib inline  
  
import imageio  
import glob  
from PIL import Image  
  
from matplotlib import pyplot as plt  
  
  
class neuralNetwork:  
 # initialise the neural network  
 def \_\_init\_\_(self, inputnodes, hiddennodes, outputnodes, learningrate):  
 # set number of nodes in each input,hidden,output layer  
 self.inodes = inputnodes  
 self.hnodes = hiddennodes  
 self.onodes = outputnodes  
 # learning rate  
 self.lr = learningrate  
 self.wih = (numpy.random.normal(0.0, pow(self.hnodes, -0.5), (self.hnodes, self.inodes)))  
 self.who = (numpy.random.normal(0.0, pow(self.onodes, -0.5), (self.onodes, self.hnodes)))  
  
 # activation function is the sigmoid function  
 self.activation\_function = lambda x: scipy.special.expit(x)  
  
 pass  
  
 # train the neural network  
 def train(self, inputs\_list, targets\_list):  
 # convert inputs list to 2d array  
 inputs = numpy.array(inputs\_list, ndmin=2).T  
 targets = numpy.array(targets\_list, ndmin=2).T  
  
 # calculate signals into hidden layer  
 hidden\_inputs = numpy.dot(self.wih, inputs)  
 # calculate the signals emerging from hidden layer  
 hidden\_outputs = self.activation\_function(hidden\_inputs)  
  
 # calculate signals into final output layer  
 final\_inputs = numpy.dot(self.who, hidden\_outputs)  
 # calculate the signals emerging from final output layer  
 final\_outputs = self.activation\_function(final\_inputs)  
  
 # output layer error is the (target-actual)  
 output\_errors = targets - final\_outputs  
 # hidden layer error is the output\_errors,split by weights,recombined at hidden nodes  
 hidden\_errors = numpy.dot(self.who.T, output\_errors)  
  
 # update the weights for the links between the hidden and output layers  
 self.who += self.lr \* numpy.dot((output\_errors \* final\_outputs \* (1.0 - final\_outputs)),  
 numpy.transpose(hidden\_outputs))  
  
 # update the weights for the links between the input and hidden layers  
 self.wih += self.lr \* numpy.dot((hidden\_errors \* hidden\_outputs \* (1.0 - hidden\_outputs)),  
 numpy.transpose(inputs))  
  
 pass  
  
 # query the neural network  
 def query(self, inputs\_list):  
 # convert inputs list to 2d array  
 inputs = numpy.array(inputs\_list, ndmin=2).T  
  
 # calculate signals into hidden layer  
 hidden\_inputs = numpy.dot(self.wih, inputs)  
 # calculate the signals emerging from hidden layer  
 hidden\_outputs = self.activation\_function(hidden\_inputs)  
  
 # calculate signals into final output layer  
 final\_inputs = numpy.dot(self.who, hidden\_outputs)  
 # calculate the signals emerging from final output layer  
 final\_outputs = self.activation\_function(final\_inputs)  
  
 return final\_outputs  
  
  
# number of input,hidden and output nodes  
# 28 \* 28 = 784  
input\_nodes = 784  
hidden\_nodes = 200  
output\_nodes = 10  
  
# learning rate is 0.3  
learning\_rate = 0.1  
  
# create instance of neural network  
n = neuralNetwork(input\_nodes, hidden\_nodes, output\_nodes, learning\_rate)  
  
# train the neural network  
  
# load the mnist training data csv file into a list  
training\_data\_file = open("mnist\_dataset/mnist\_train.csv", 'r')  
training\_data\_list = training\_data\_file.readlines()  
training\_data\_file.close()  
  
# epochs is the number of times the training data set is used for training  
epochs = 5  
for e in range(epochs):  
 # go through all records in the training data set  
 for record in training\_data\_list:  
 all\_values = record.split(',')  
 # scale and shift the inputs  
 inputs = (numpy.asfarray(all\_values[1:]) / 255.0 \* 0.99) + 0.01  
 # create the target output values (all 0.01, except the desired label which is 0.99)  
 targets = numpy.zeros(output\_nodes) + 0.01  
 # all\_values[0] is the target label for this record  
 targets[int(all\_values[0])] = 0.99  
 n.train(inputs, targets)  
 pass  
 pass  
  
# test the neural network  
  
# load the mnist test data csv file to a list  
test\_data\_file = open("mnist\_dataset/mnist\_test.csv", 'r')  
test\_data\_list = test\_data\_file.readlines()  
test\_data\_file.close()  
  
# scorecard for how well the network performs,initially empty  
scorecard = []  
# go through all records in the test data set  
for record in test\_data\_list:  
 all\_values = record.split(',')  
 # correct answer is first value  
 correct\_label = int(all\_values[0])  
 # scale and shift the inputs  
 inputs = (numpy.asfarray(all\_values[1:]) / 255.0 \* 0.99) + 0.01  
 # query the network  
 outputs = n.query(inputs)  
 # the index of the highest value corresponds to the label  
 label = numpy.argmax(outputs)  
 # print("Answer label is:",correct\_label," ; ",label," is network's answer")  
 # append correct or incorrect to list  
 if label == correct\_label:  
 # network's answer matches correct answer, add 1 to scorecard  
 scorecard.append(1)  
 else:  
 scorecard.append(0)  
 pass  
  
# calculate the performance score ,the fraction of correct answers  
scorecard\_array = numpy.asarray(scorecard)  
print("performance = ", scorecard\_array.sum() / scorecard\_array.size)  
  
# 测试神经网络是否能准确识别自己的手绘28\*28 png图像  
  
# our own image test data set  
our\_own\_dataset = []  
  
# load the png image data as test data set  
Image.open("mnist\_dataset/6.png").resize((28, 28), Image.LANCZOS).save("mnist\_dataset/6.png")  
for image\_file\_name in glob.glob('mnist\_dataset/6.png'):  
 # use the filename to set the correct label  
 label = int(image\_file\_name[-5:-4])  
  
 # load image data from png files into an array  
 print("loading ... ", image\_file\_name)  
 img\_array = imageio.imread(image\_file\_name, as\_gray=True)  
  
 # reshape from 28x28 to list of 784 values, invert values  
 img\_data = 255.0 - img\_array.reshape(784)  
  
 # then scale data to range from 0.01 to 1.0  
 img\_data = (img\_data / 255.0 \* 0.99) + 0.01  
 print(numpy.min(img\_data))  
 print(numpy.max(img\_data))  
  
 # append label and image data to test data set  
 record = numpy.append(label, img\_data)  
 our\_own\_dataset.append(record)  
  
 pass  
  
# test the neural network with our own images  
  
# record to test  
item = 0  
  
# plot image  
matplotlib.pyplot.imshow(our\_own\_dataset[item][1:].reshape(28, 28), cmap='Greys', interpolation='None')  
  
# correct answer is first value  
correct\_label = our\_own\_dataset[item][0]  
# data is remaining values  
inputs = our\_own\_dataset[item][1:]  
  
# query the network  
outputs = n.query(inputs)  
print(outputs)  
  
# the index of the highest value corresponds to the label  
label = numpy.argmax(outputs)  
print("network says ", label)  
# append correct or incorrect to list  
if label == correct\_label:  
 print("Good,match!")  
else:  
 print("no match!")  
 pass  
  
plt.show()

总结：

通过这门课的学习，我学会了网页爬虫、数字识别等相关知识，在今后的学习中可以通过网页爬虫来抓取网络上的数据，比如：文档、资料、图片等，可以对抓取的数据进行数据分析。在数字识别过程中，我遇到了

%matplotlib inline  
    ^  
SyntaxError: invalid [syntax](https://so.csdn.net/so/search?q=syntax&spm=1001.2101.3001.7020) 报错的问题，MatPlotlib图表不能在Pycharm中显示，需要删掉这行代码，用plt.show()来展示图表。