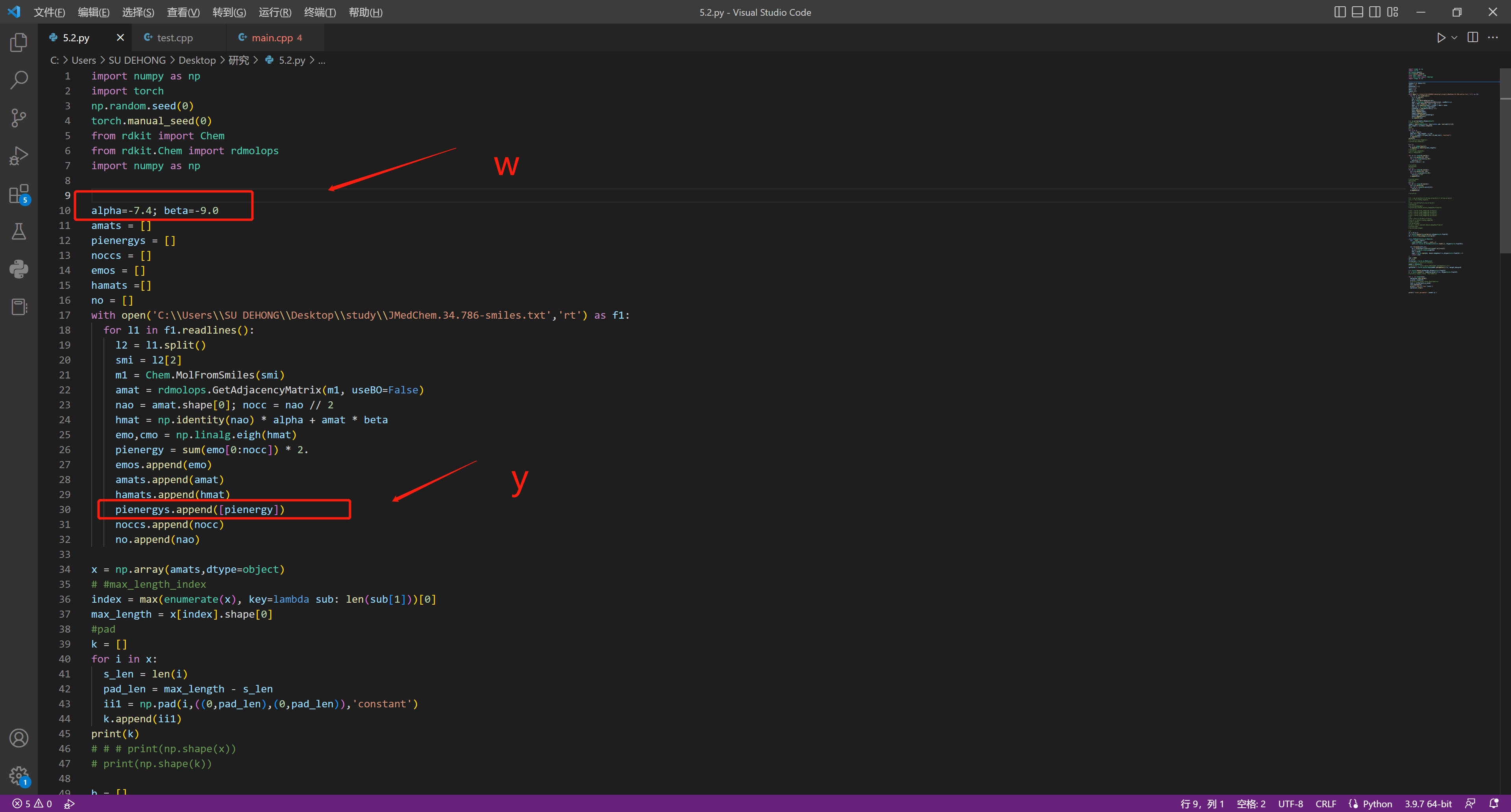
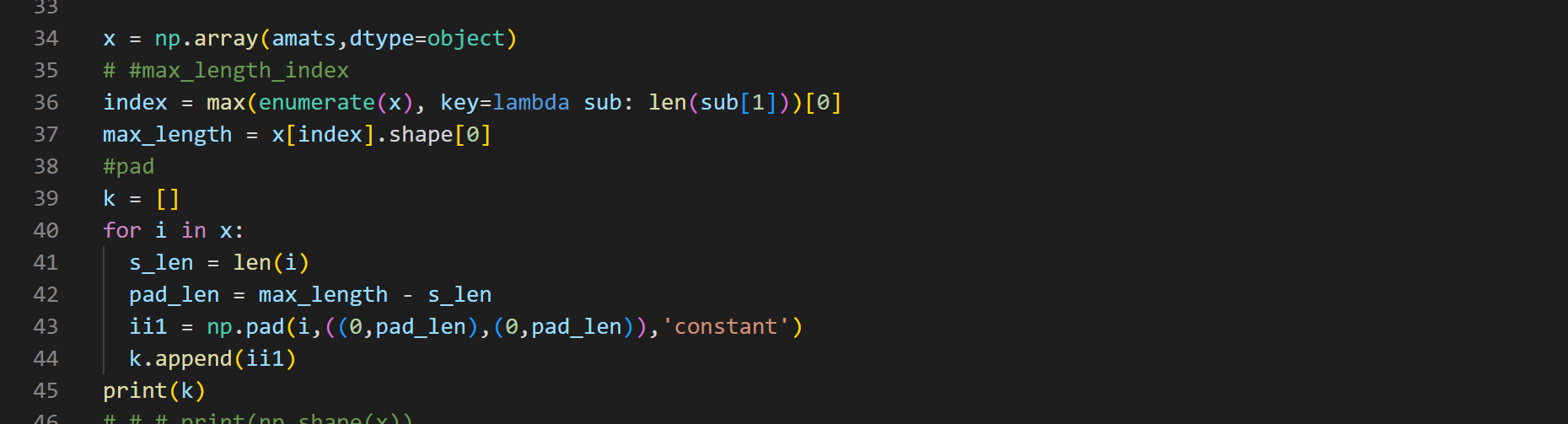
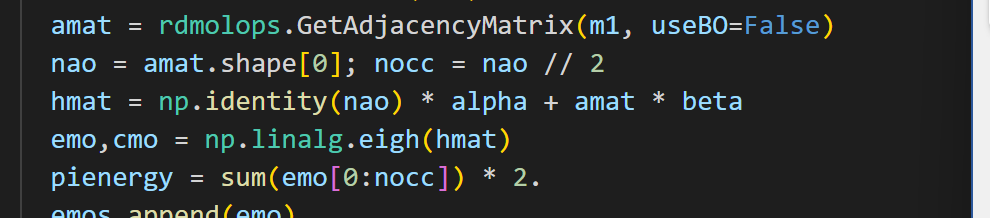
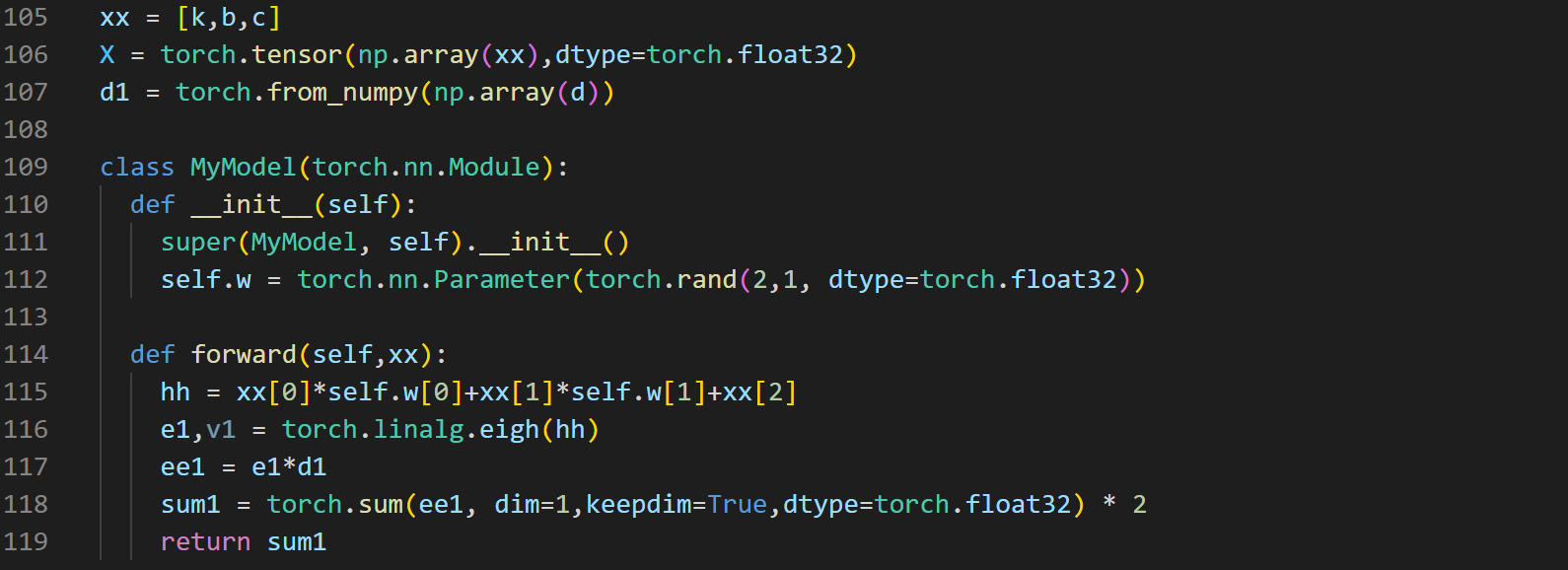
This is a simple linear regression prediction problem. In the given JMedChem.34.786-smiles file, I will extract the smile formula and generate the adjacency matrix of the molecule. Using the known w, calculate y.

Now we have smile and y. Write a simple linear regression model to predict w.

Since the length of the adjacency matrix generated by smile is different, it cannot be transformed into a tensor, so it has to be padded (padding), so the longest number of the adjacency matrix generated by 197 molecules is taken as the maximum number of padding (28), and the insufficient ones are padded with 0.







The function of Model is designed according to the original pienergy calculation process. There is another difficulty, the matrix after torch.linalg.eigh, the generated e (eigenvalue),v (corresponding eigenvector), e will be sorted by eigenvalue from smallest to largest. This is a problem in data processing, because the generated e will have a lot of zeros in the middle, such as [1,2,3,0,0,0,0,0,8], because it is filled with a lot of zeros. In this case, there will be problems in the subsequent summation by the number of individual numerators. So, the idea now is to write the fill matrix of amat, then write the same unit matrix of shape and amat's shape, then multiply it by the w parameter to be learned. Finally, where the fill is, change it to 777 on the diagonal (lucky number, as long as it is greater than the other e).

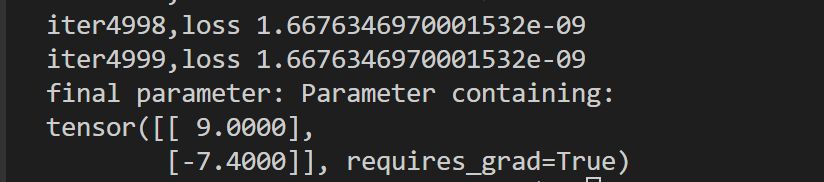
The last sum, according to the original 197 molecules of the number of individual molecules to extract the number of molecules of the general elements, do sum and then multiply by 2. In order not to use the for loop, I use the matrix corresponding elements multiplied, do masking, will be added after the 777 to mask out. The effect is the same as the original code.

hmat = np.identity(nao) \* alpha + amat \* beta

emo,cmo = np.linalg.eigh(hmat)

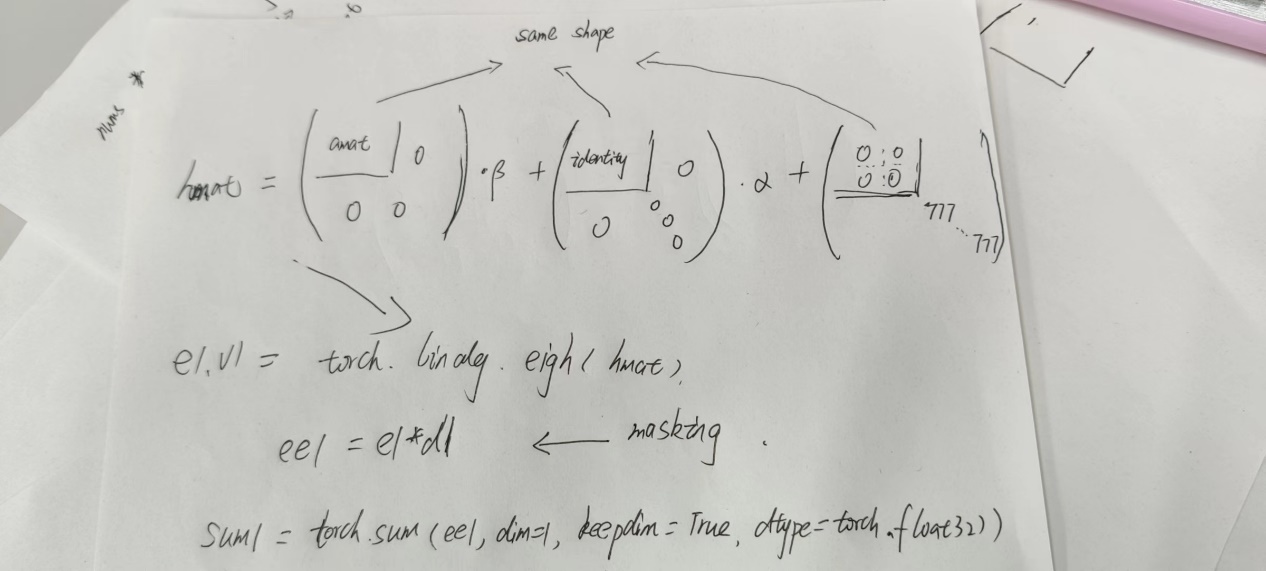
pienergy = sum(emo[0:nocc]) \* 2.

The final result, with loss finally converging to 0, results in



One of the parameters is the opposite of the original one. But the loss tends to 0, indicating that the current data processing is correct, and the opposite parameter is estimated to be an internal problem of the model, which can be ignored.

Data Core Ideas：



This code can be used as an introductory deep learning study.

Thanks