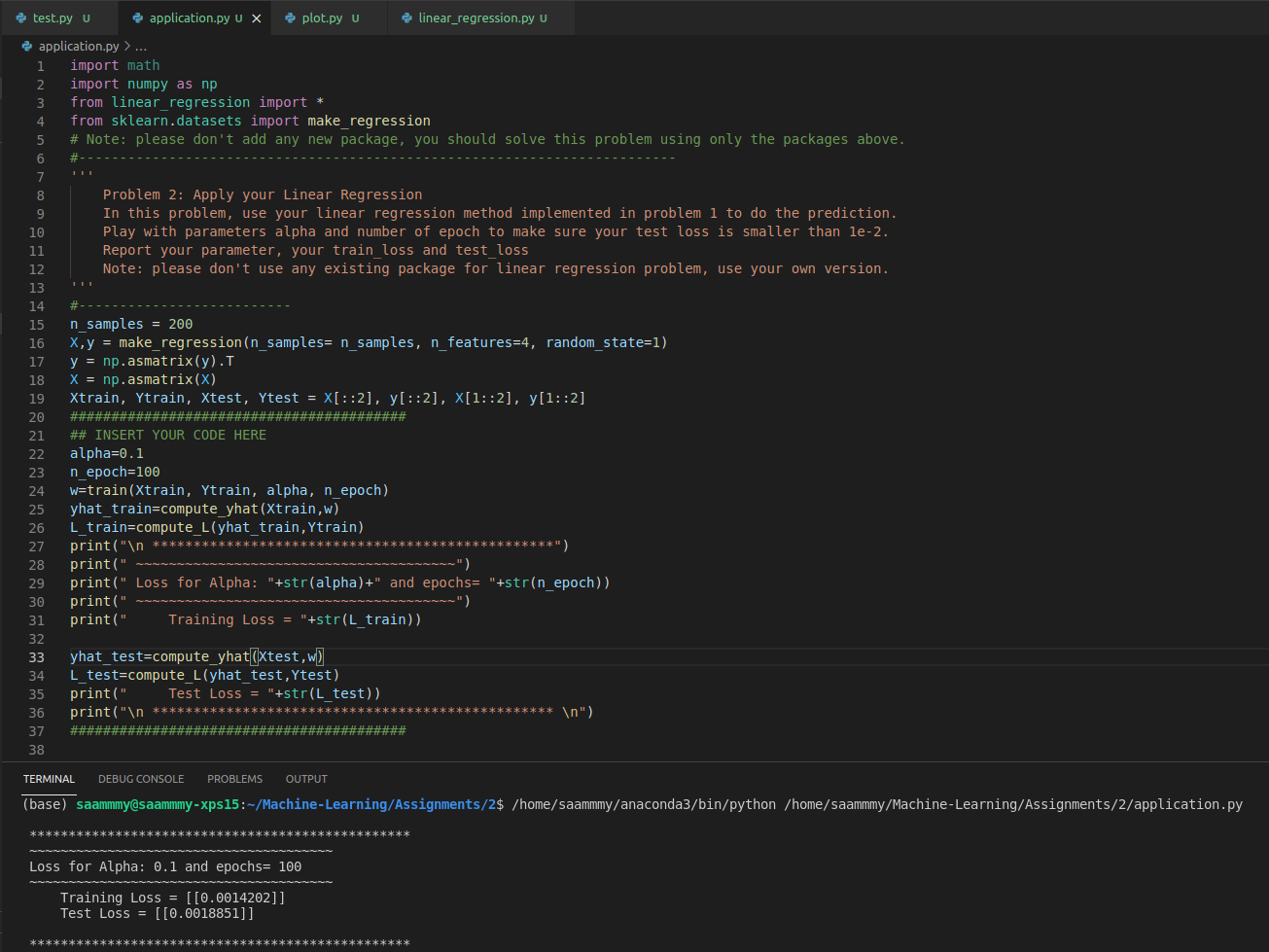
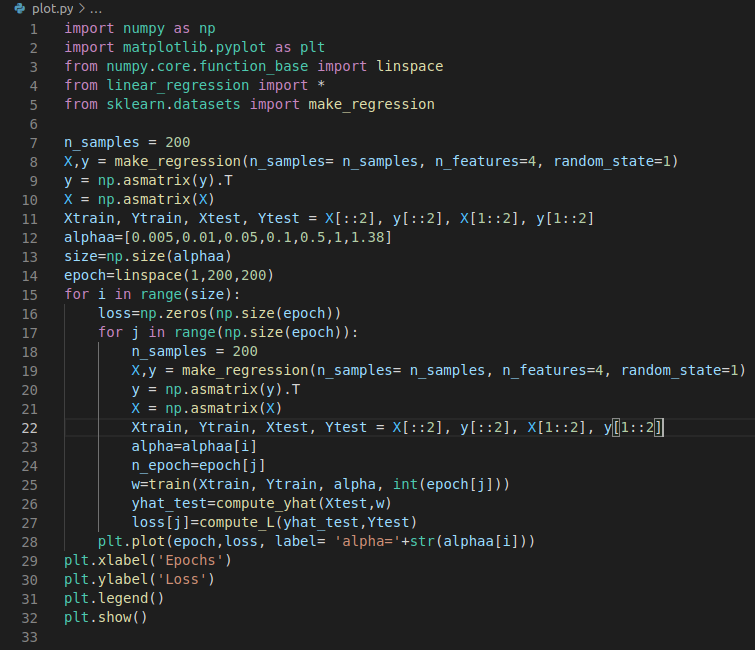
Homework #2

**Part 2:** Make Predictions by using your implementation.

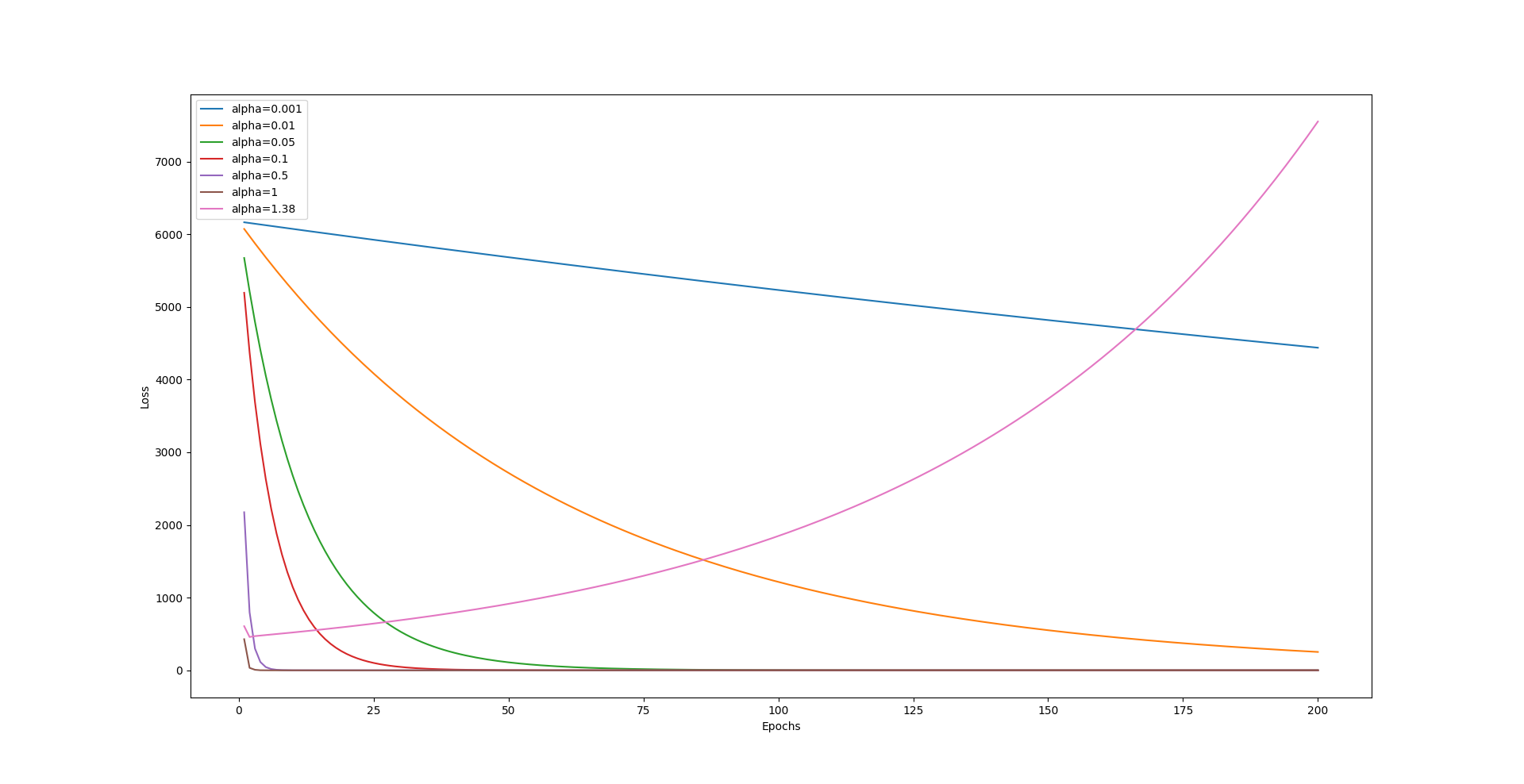
* First, I inserted my code into the application.py script.
* Below is my code where I trained the data and used the test data to find training loss.
* The o/p below shows both the training and test loss.



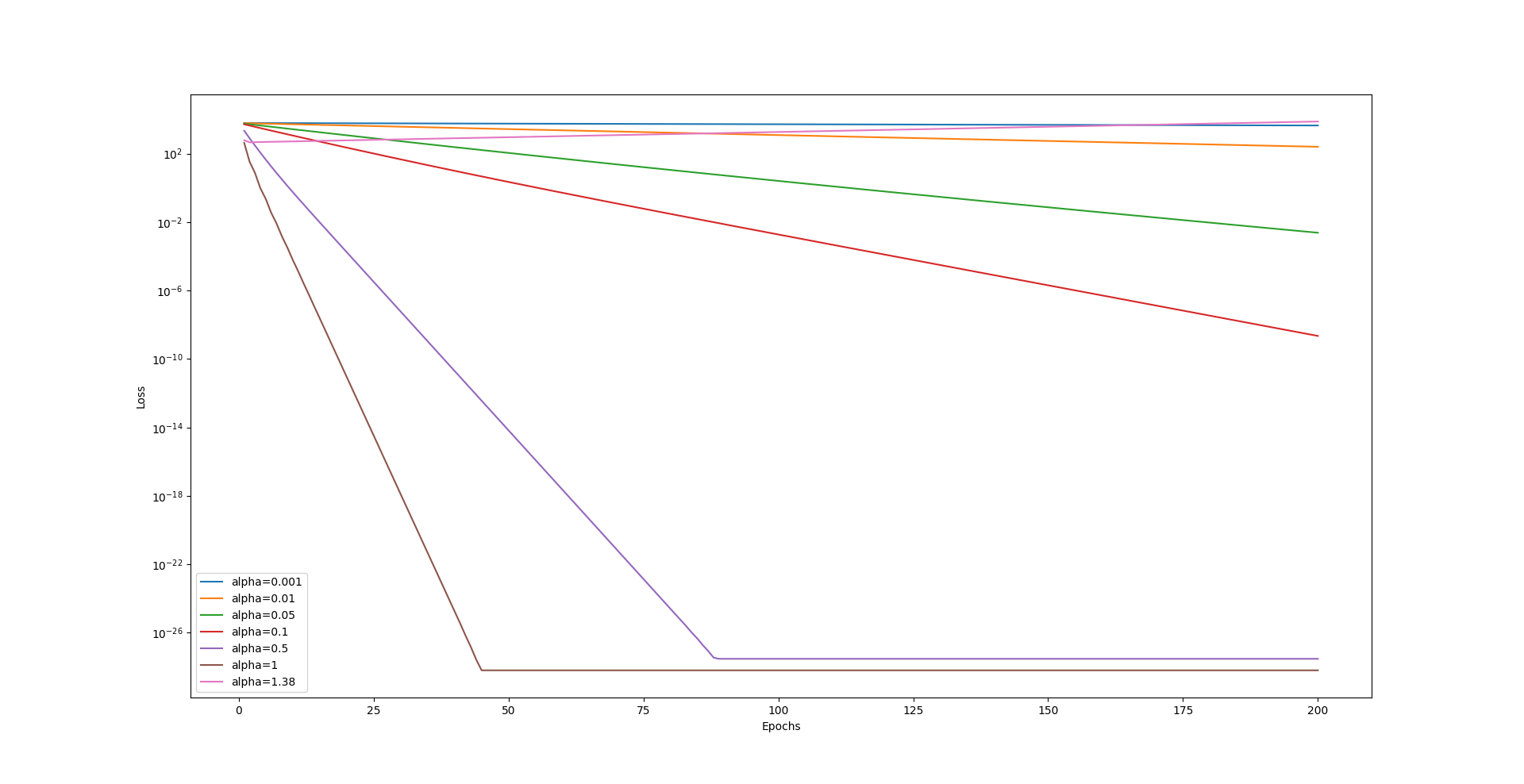
* After playing with values of epoch and alpha, I found that at:
  + **alpha=0.1 & epochs=100:** Training Loss= 0.0014202 Test Loss = 0.0018851
  + Remind you this is not the best answer. In fact, the graph plotted below shows we can achieve almost zero loss (in 10^(-29)) for various alpha and epoch values.
* Now we want to find the relationship between alpha and Number of epochs.
* The best way to do this is to plot the value of loss for various alpha and no of epochs.
* So, I developed a function named plot.py. The code for plot is as shown below.



* This code helps me to iterate through various alpha’s and plot a graph between Loss and Epochs. Where x-axis is the no of epochs and y axis is the testing loss. Below is the graph and the interpretation:
* As you can see each color line represents the values of loss at no of epoch for a particular alpha.



* We can observe the following:
  + On increasing the alpha (Learning rate) the loss lowers down till alpha=1. Post that the model starts overshooting, this is shown for alpha =1.38 (pink line)
  + Now for a particular alpha, we can observe that on increasing epoch we see the loss decreasing and then becomes constant over no of epochs. Except for alpha>1, at this point the model keeps overshooting on increase of epochs.
  + As we can see from the graph, most have approached a loss of 0 by 100. But if we scale the Y- axis to “log” we get a clearer picture. The graph is shown below:
  + So below you can see at alpha=1 and epoch>=45 the loss is <=6.45x10^(-29). This can be confirmed by putting these values in application.py as well.



**Hence, we understand on increasing alpha(alpha<=1) and no of epochs we get a lower loss.**