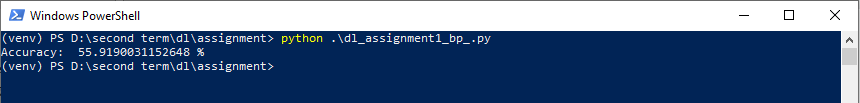
**Back Propagation**

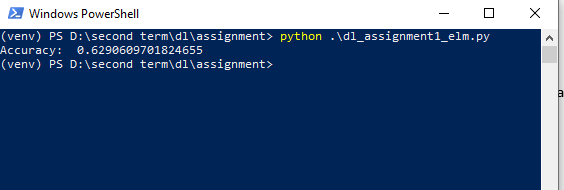
* Steps to execute code:
  1. Import both the files dl\_assignment1\_bp\_.py and dl\_assignment1\_bp\_network.py in any IDE and run the first file or in command prompt just write python dl\_assignment1\_bp\_.py and the program will be executed.
  2. Once the code is executed, it gives an accuracy of (The result screenshot is given below). Personally I used windows powershell as a command prompt to execute the program.



* Explanation of code:
  1. Initially dataset(.csv) is read using pandas dataframe.
  2. Splitting the dataset in training and testing in the proportion 70:30 using sklearn’s train\_test\_split() function.
  3. Letter the dataframe is converted to basic python structure – list and dictionary for ease.
  4. Then initialized neural network with four hidden layers and random weights for each layer.
  5. Then network is trained with the learning rate of 0.0001 and 500 epochs (This is done based on trial and error method, best is learning rate and epochs are selected after several results)
  6. Then Back Propagation algorithm is used, where errors are backpropagated and weights are forward propagated.
  7. Finally, the accuracy is calculated on test set and it is found as: 55.91
* Dataset Information:
  1. The same dataset is used for both Back Propagation and ELM.
  2. The dataset used is EEG Eye State (<https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State>)
  3. Number of features: 14
  4. Number of instances: 14980
  5. Class variables: 0 or 1

**ELM**

* Steps to execute code:
  1. Use the same steps as shown in Back Propagation except the file. Here the filename is dl\_assignment1\_elm.py.
  2. Result is: 62.90



* Explanation of code:
  1. First two steps are same as the previously shown in Back Propagation algorithm.
  2. Number of neurons decided is 15 based on the same technique – trial and error giving the based accuracy of: 62.90. That is the difference of ELM that in ELM numbers of neurons are larger compared to other neural networks.
  3. Initially weights are randomly generated using random function.
  4. Then output weights are calculated based on sigmoidal neurons.
  5. At last, accuracy is calculated and result found is: 62.90