① Define your Neural Net

class MyNN

- Give number of input layer elements as input
- Choose your number of hidden layers
- Choose the number of neurons per hidden layer
- Choose your activation functions
- Give the number of neurons to output layer / give it an activation function
- You can use a specific (like Xavier) initializer for the weights / biases.

---

② - Read / Preprocess your data and convert it to a numerical / tabular format

- Divide your tabular data (matrix or dataframe) into 3 parts : training / validation / test sets.
  (Can do normalization/scaling here)
- Define : batch-size, learning rate, number of epochs, patience if using Earlystopping.
- Decide on your Loss & Optimizer methods
- Training / validation (with Earlystopping)

```
for each epoch
    for each training batch
        feed the batch to model (gradient computation is on/enabled)   this says, do learn/update weights
        ↳ compute loss
        ↳ give loss to optimizer → does the weight updating through backpropagation
        ↳ collect this training loss
    compute / report the average training loss of this epoch
    for each validation batch
        ↳ here.
        disabled the model from backpropagating so it won't learn from validation set.
        feed batch to model
        ↳ compute the loss, collect them
    compute / report the average validation loss of this epoch

    collect the "patience" number of past and current validation
    loss. If new loss is not less that the saved others,
    stop the epoch loop, return the model. Training is done.
    Otherwise, continue the epochs.
```

Now, the model is ready. Try it on the test set to see how well it generalizes.