In the first part of the exercise we will learn to build a simple neural network model using Keras library which is a high level neural network API with TensorFlow in the backend for the low level operations.
In the second part we will use the scikit-learn wrapper together with the cvGridSearch function in scikit-learn to search for the best parameters to build our neural network

Part 1 - building a simple neural network

Following the link:
http://machinelearningmastery.com/tutorial-first-neural-network-python-keras/

1. Load the indian-puma-diabetis dataset from the following link: http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data
2. Define a "sequential" model in Keras with three layers: input, hidden and output. The input layer should have layers as the number of features, for the hidden layer we will use 8 neurons and for the output layer we have one neuron, predicting diabetes https://keras.io/getting-started/sequential-model-guide/
3. Compile the model, don't forget to define the loss function and the optimizer as params to the compile function
4. Fit the model to your data, define as params the number of epochs and the batch_size
5. Evaluate the model on the training data using the "evaluate" function
6. Predict the results on some of the training data using the "prdict" function

Part 2 - cross validate to find best params

Following the link:
http://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

1. Define a function returning a keras model
2. Build a KerasClassifier with the param: build_fn equal to the function you have just defined. https://keras.io/scikit-learn-api/
3. We will build a GridSearchCV object in scikit-learn which accepts a dictionary with all values we want to check and runs the model on all possible values, returning the model with the best params
   http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

4. We will first tune the batch_size and epoches params, build a dictionary with batch_size between 10 and 100 with step of 10 and epoches of 10,50,100. Run the GridSearchCv with estimator param equal to our KerasClassifier object and param_grid equal to our dictionary we just built.
5. Fit the grid and print the best_score_ and best_params_
6. Use the best epoch and batch_size as constants in the model and let's learn now the next hyper-parameters:
    a. The optimization algorithm: https://keras.io/optimizers/
    b. Learning rate and momentum
    c. Network weight initialization
    d. Neuron activation function
    e. Dropout regularization
    f. Number of neurons in the hidden layer

Good Luck!