Software components:

Classs:
Cortex

- Handles data (time series array of LFP), data preprocessing, fits data to spiking model predefined, and holds parameters.

Class attributes:

Lfp_data (array of timeseries)
Spike_model (class)
Connectivity_matrix (matrix of connectivity weights between neuronal layers)
Neron_density (array of density in each neural layer 1-5)
Sampling rate (int describing frequency of timeseries of lfp data)

Functions:
Preprocessing_data(self)

- preforms filtering into frequency bands to help with model fitting and normalization of the LFP data

fit_model(self,error_tolerence)

- fits the LFP data to the model class given an error tolerance that is used for each frequency band.
- First runs the model and checks to see if the LFP generated matches the frequency bands of the input LFP data
- Checks error tolerance
- Updates model parameters with gradient descent and runs the model again
- Repeat until error_tolerance is reached

output_connectivity(self)

- returns the parameters of the model fit to the data

visualize_connectivity(self)

- outputs a visual representation of the neural activity in the form of 5 neuronal layers with densitys and connections between layers with a density that reflects the connectivity of the model parameters

Class:
SpikingNetwork

- intitializes a network of spiking neurons with predefined connectivity and neuronal density parameters that can be fit to the neural data

Class attributes:

Num_neurons (int defines number of neruons in the network)

Connectivity_matrix (matrix of connectivity between each layer and the other layers of the model.
Neuronal_threshold (array that defines the membrane potential of each neuron)

Functions:
Simulate_step (self)

- updates the neuronal_threshold via a monte carlo simulation and applies the update given the connectivity_matrix
- checks if neruons reach their firing threshold potential
- resets neurons that spike

run_stimulation(self, T, dt)

- funs the simulation_step function over a set time period (T) and time steps (dt)

output_parameters:

- outputs the model parameters which are neuronal densities of each layer and the connectivity matrix

generate_LFP:

- simulates an lfp by summing the membrane potentials at each time step by weighting the membrane potentials by the depth of network layer
- outputs stimulated LFP

Interactions to accomplish use cases:

Use Case: Find connectivity and neuronal density given an experimentally recorded LFP. This generalized use case fits all three use cases in the functional specifications that can be used for different analysis processes.

For this generalized case, the Cortex class will need to be inialized given a recorded LFP and sampling rate. The Cortex class will the need to go through a preprocess_data step to transform the data into a usable form for the model. Then the user will need to use the fit_model function and input an error tolerance. This step will initialize the model that runs for T time (the number of timesteps of the input LFP) at a step of dt (the sampling rate of the LFP). The model will fit the generated LFP to the experimental LFP by updating the model parameters (the neuronal density and the connectivity matrix). This step will take the longest time since a single spiking model will take a considerable time to generate. The user can then visualize the model with the visualize_connectivity function.

Preliminary plan:

1. Implement the neuronal spiking class to generate realistic LFPs and neuronal spiking data
2. Implement the Cortex class and intitalize
3. Run one iteration of the model_fit class to see how well a single run fits the parametres
4. Run on multiple interations of the model_fit class to see if changes need to be made to the gradient descent step of the function
5. Final touches and optimization