Recent Progress:

1. Given up on GE for API fixes or updates.
2. SQL database for cross instance/code language communication concept proven.

   Involves using an SQL database as a common data resource. A timing schedule is required so that a process doesn't 'step' on another process and cause concurrency issues.

   Using SQLite3 is may not be a great choice - it is not designed for 'fast' use but is included with python3 and ironpython.

   Other SQL database packages *could* be explored - Must be compatible with Ipy32.
3. AMQP (Advanced Message Queuing Protocol) for cross/instance code language communication concept proven. (suggested by Phil)

   Involves sending messages to an external service running on the local machine. These messages are then collected via code interfaces.

   Results are consistently faster than SQL method (≈10-20 times faster)

   Unsure if it's best to use this alone, or in combination with a SQL database.
4. AMQP SQL method for cross/instance code language communication concept proven.

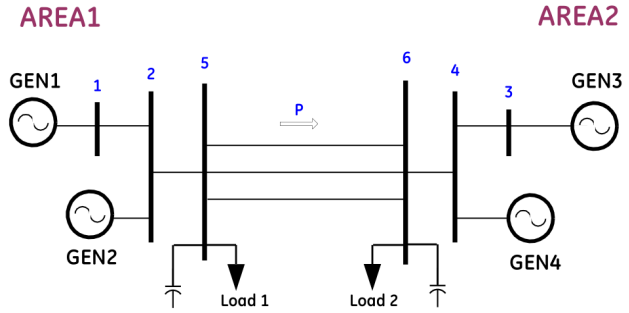
   Provides consistently faster SQL access (≈8-11 times faster than SQL only)

   Possibly the way to go - More forward development thinking required before decision.
5. Read papers from the 70's by Mills, Taylor, and Cresap (surprisingly good).
6. PSLF terminal output can be suppressed by EPCL command : `dispar[0].noprint = 1`
7. GitHub repository updated:

   `https://github.com/thadhaines/` (SQL and AMQP tests also on GIT)

Current Tasks:

1. Figure out how Python 3 is going to send data to, and receive data from, PSLF.
2. Define Agent actions for AGC/LFC (ACE calculations), Shunt Control, β calculation
3. ODE solver == numpy / scipy `odeint` in Python 3
4. Continue experimenting with a multi-area model:



5. Investigate line current data (add branch section agents to model)
6. Refine data output - keep quickplotter in mind - Dictionary structure, variable naming, functionality, meta...

Future Tasks: (Little to No Progress since last time / Things coming down the pipe)

1. Add Ramp perturbance Agent
2. An agent for every object: Shunt, SVD, Branch, Transformer, Power Plant, ...
3. Think about having all simulation parameters be defined in one file that is fed into simulation. i.e. Have all perturbance, AGC, LTD, etc., related code in a text file so python doesn't have to be a 'user' thing.
4. Package code into library and refactor (think of a nice name):
   Power System Long-Term Dynamic Simulation → PSLTDSim
5. Investigate Runge-Kutta integration.. (probably inside scipy...)

$$x_{n+1} = x_n + h(1/6)\,[\,k_1 + 2k_2 + 2k_3 + k_4\,]$$

Fourth-order Runge-Kutta method

$$k_1 = f(x_n)$$
$$k_2 = f[\,x_n + (h/2)\,k_1\,]$$
$$k_3 = f[\,x_n + (h/2)\,k_2\,]$$
$$k_4 = f(x_n + h\,k_3)$$

6. Enable multiple dyd files to overwrite / replace previously defined agents/parameters
7. Identify System Slack bus programmatically (maybe just assume in first area?)
   or calculate system slack error differently... An average of slack error?

Current Questions:

1. Overview of planned PSLF scenarios? → Similar to Heredia but on MiniWecc Scale?
2. Is there more available/relevant event data that may help us to verify simulations of specific instances (wind ramps or other behavior) that the novel research will focus on?
   (Heredia paper data helpful for some wind ramp data context)