Ideas for Implementation:

Helpful Click Decorators & Utilities

- click's `--help` option will be ideal for users
- `click.option()`: this will be very useful for flags that may be used
  - will be able to use commands like `is_flag`, `flag_value`, `count`, `help`, etc.
- `click.group()`: this will allow FRE to be broken up into parts and subparts for each part
  - will be able to use commands like `add_command`
- `click.progressbar()`: potential for the user to see progress while something runs like `fre run`
- `click.confirm()`: potential for the user to verify actions and proceed
- `click.style()`: can style text with wanted configurations if needed (can use click.secho())
- `click.pause()`: stops executing current command and waits for user input to continue

Potential Errors

- environments: I'm working with mainly `venv` right now, but `conda` and `venv` on other computers may fail
  - this is likely due to the setup, within `setup.py`` and calling ` pip install --editable` after
  - however, using ^ this allows for cleaner execution, where the user only has to type `$scriptname` in the command line instead of `python $scriptname.py` (followed by any needed commands and flags of course)

Questions for Users/Devs

- do we want to use flags (`click.option()`), confirmations (`click.confirm()`), or a mix of both to allow users to run what they want, how they want?
  - this means that users can either use certain flags (i.e `--execute`), which will be included and explained in the `--help` feature, or they will just be prompted for what features they want and can decide if they want it with [y/N]

Potential Additional Uses for Click

- program using BeautifulSoup to scrape GFDL pages for immediate tutorial guidance after prompting for GFDL login