Objectives

- The idea of this test is to create a basic lens and sensor simulator, adding complexity to demonstrate specific skills.
- As a minimum python test, you should complete all the **Base Work** section tasks.
- Based on your knowledge and experience, select the points from the **Specific Skills** section that you feel comfortable doing in a reasonable amount of time.
- Return your work as a compressed file `python-test-<your-name>.tar.gz`. It should contain a `Readme.md` file explaining which parts of the test you did and any relevant comments.

Base Work

1. Define a base python class `BaseProcessor` with:
    a. An abstract method `process(image)` to process the input 2D numpy data (image) and return output 2D numpy data (image).
    b. A boolean property (both setter and getter) `enable`.
2. Using `BaseProcessor` as parent class define `Lens` class with:
    a. An integer property `height` (both getter and setter).
    b. An integer property `width` (both getter and setter).
    c. The `process` method should validate that the shape of the input numpy data matches the Lens' `height` and `width` properties. If the shape matches, the `process` method needs to return the input image otherwise `raise ValueError`.
3. Using `BaseProcessor` as parent class define `Sensor` class with:
    a. A property `gain` (int both getter/setter).
    b. The `process` method output value should be `gain * input` (both input and output are 2D numpy arrays).
4. Add docstrings for all defined python classes and methods using the Google standard.

Specific Skills

Documentation

Add documentation for all defined python classes.

- Use sphinx to generate html documentation.
- Create a tutorial jupyter notebook as tutorial for the package use.

Packaging

Create a python-package (`camera-simulator`) containing all above classes. The package needs to:

- Define a `setup.py` to create python wheels.
- The dependencies need to be clearly defined within the python package and be used by `setup.py`.

Testing

Write unit tests for the package.

Docker

Create a `Dockerfile` to run the unit tests.

Advanced Python

- Implement a `Lens` decorator for `Sensor`s, such that each time the `Sensor` process is called, the `Lens` process is called.
- Allow the `Sensor` to be used as an iterator of 10 elements. At each cycle, the returned image is the original image plus the index of the iteration.
- Create a function `mymean` that generates a random image, passes that image through a `Sensor` object and returns the mean of the image. Create a console entry point that calls that function. If the package is installed, the entry point should be accessible as `pysensor`.
- Demonstrate the use of the `concurrent` package to create a pool of 5 workers and call the previously created mymean function 100 times.