<AI Startup> Programming Challenge

Skills needed:

- Python 3.4+ (specify which version used in `README` file)
- Python `multiprocessing` (https://docs.python.org/3.7/library/multiprocessing.html)
- Python `numpy` (http://www.numpy.org/)
- Python `opencv` (https://pypi.org/project/opencv-python/)

Requirements:

1. Make a python program that runs from the command line (`python3 YOUR_SCRIPT.py`)
2. This main program should spawn three `multiprocessing.Process`. Do not use threads.
3. In the main application:
    a. Ask user for the number of random images to generate.
        i. Store in `multiprocessing.Value` named `num_images`.
    b. Ask the user for the dimensions of the image.
        i. Store in `multiprocessing.Value` named `height` and `width`.
    c. Create two `multiprocessing.Queues` named `queue_a` and `queue_b`.
    d. Create one `multiprocessing.Array` named `array_a`.
    e. Spawn the three `multiprocessing.Process` (additional instructions below).
    f. Wait for a new image from `queue_b`.
    g. Store the image in `array_a`.
    h. Allow the user to continue to the next image with '`<ENTER>`' or to quit the program with '`q`'.
        i. Use `multiprocessing.Event` named `event_quit` to signal program completion to the processes.
    i. Properly join all processes and cleanly exit.
4. The first process should:
    a. Use numpy to generate an RGB image of a solid color. The color generated should be randomly selected from the following list of RGB colors: `['black', 'white', 'red', 'yellow', 'lime', 'aqua', 'blue', 'fuschia']`
    b. After the image is generated it should pass the image to the second process using `queue_a`.
5. The second process should:
    a. Wait for a new image from `queue_a`.
    b. Inspect the image in order to determine its color. Use only the image data (no metadata).
    c. After inspecting the image and determining its color, use `opencv` to watermark the image with the name of the color. The second process may assume that the color is one of the colors from the above list.
    d. Draw a filled circle of the complementary color (opposite on the color wheel) in the middle of the image using a radius of ¼ the minimum image dimension. Avoid using for loops to do this.
    e. After watermarking and drawing a circle on the image, place the new image onto a different `queue_b`.
6. The third process should:
    a. Continually read from `array_a`. Remember that the contents of this array will be updated from the main application.
        i. `numpy.frombuffer` will be useful for reading and writing this array efficiently.
    b. Display the image using `opencv.imshow`.
7. Document all code using python docstrings.
8. Write unit tests for all functions which will be executed by pytest (`pytest test_YOUR_SCRIPT.py`)
9. Include a `README` file. Compress the project directory and include your name in the filename.