**What’s for Breakfast?**

Intro to Docker Exercise:

*Containerize a simple python script*

**Folder will contain:**

* Breakfast.py

**Build an Image:**

1. Create Docker file
   1. Will walk through step-by-step and ensure people understand each line
2. Build an image
   1. Docker build .
   2. The initial build will take a bit of time
3. Make sure your image is there
   1. Docker images
4. Notice it does not have a name ☹
5. Name the image
   1. Docker tag (imageID) breakfast
   2. Note: you should actually name in the build step
   3. E.g. Docker build -t breakfast .
6. Try to containerize this image with “docker run breakfast”; what happens?
   1. Oh no! I’m getting errors! Why is this?
   2. Hint: with docker run, we default to being in “attached” mode. What could that mean?
      1. We’re “attached” in the sense that we can view the output of the container. However, we are not actually able to enter anything
   3. We will need to run this container in interactive mode!
7. Containerize the image in interactive mode this time, so that it works
   1. Docker run -it breakfast
8. Notice that the container is no longer running. Use docker ps -a to view the container name
9. Try to restart that container. What happens now?
   1. By default, “docker start” will run in “detached” mode, so we run into this same issue where we can’t input our values
10. Now, try to restart the container in *attached* and *interactive* mode, so that we can work with the container again
    1. E.g. docker start -a -i f8afc88b9ed4
    2. Note: We need to specific -a for “attached.” We also need to re-emphasize that we’re using the container in interactive mode “-i.” However, we do not need the -t flag because that flag persists from the original “docker run” command that we typed earlier in the exercise

**Hooray!!** You just built an image from scratch and created workable containers!!

If time, move to Part 2

**2. Troubleshooting**

1. Run “docker logs containername” on any container to see logs
   * This can be good for going back and seeing that actually happened with your containers