I made the decision to use object-oriented programming to complete this task. I don't have much experience with it, so I anticipate learning a lot from this task. Since a group of three would need to put in more effort and come up with a better solution, I also made the decision to solve the problem alone. If working alone on this project proves to be too difficult, I'll work in a group on the following project.

Comments regarding the different tasks:

1) I created a Container Class with get and set methods using fundamental OOP techniques. Although I created it for all the parameters the Container class accepts, I'm not sure whether it is actually needed. Leaving it there for the time being
2) I created the ContainerSet class, which is contained in the container set.py file, to manage a set of containers. The idea behind this class-object is that there is currently just an unorganized pile of containers waiting to be loaded onto a container ship on the docks. Later, in another class, the implementation of the sorting process will take place.In the Container
3) The function that creates a set of containers at random is found inside the ContainerSet class, and it utilizes the function generate_random_container() from the container class to create a "random" container (that is: either a 20-feet container or a 40-feet container).
4)
5)

```
def load_container(self, container):
    container_length = container.get_length()
    if container_length == 20:
        for height in range(self.height):
            for width in range(self.width):
                for length in range(self.length):
                    if self.get_nth_container(height, width, length) is None:
                        self.insert_container(container, height, width, length)
                        return
    elif container_length == 40:
        for height in range(self.height):
            for width in range(self.width):
                for length in range(self.length - 1): # -1 because of the 40 foot container can't be placed at the end of the ship
                    if self.get_nth_container(height, width, length) is None and self.get_nth_container(height, width, length + 1) is None:
                        self.insert_container(container, height, width, length)
                        self.insert_container(container, height, width, length + 1)
                        return
```

Reported experiments:

1) The amount of time required to load containers onto a ship, as opposed to loading a new ship from a file containing the data of an already loaded container ship, using the loading algorithm to determine if a 40-foot container can be put such that there are no gaps in the ship's loading.

   Comparing loading a set of containers with loading a ship, it takes 26 seconds to load 6,000 containers into a 23 by 22 by 18 ship, yet it takes less than one second to load the ship from a previously sorted file.

Other comments that should be included in the contract:

- The naming conventions.
-