Summary of the Dock Scene Program

The following are the variables, classes and functions that you need to work with in order to complete the exercises and assignments. You can (and must) use other basic Python functionality and control structures.

containerList[]	(Global) variable that denotes a list of all containers in the scene
Class: shipContainer	class that represents a single container of size 8 (width in x direction) by 40 (length in y direction) by 8 (height in z direction) with the following attributes that you can access/edit: • name ... name of the container • shipper ... shipping company who owns the container • origin ... location where the container was last filled • pos ... a vector consisting of three values (x,y,z) that can be accessed using pos.x and so on
addContainer(…)	Function to create a new container and add it to the dock scene; needs to minimally specify the new container's location and has the following optional arguments: • color ... color of the container (by default the color is determined by the shipper for 4 default shipping companies) • name ... name of the container • shipper ... shipping company who owns the container • origin ... location where the container was last filled
setShipper(…)	Sets the shipper of a container to a certain value (optionally also the origin)
setOrigin(…)	Sets the origin of a container to a certain value (optionally also the shipper)
moveCraneTo(x,y) ...	moves the hook of the crane in a horizontal plane without changing its vertical position; when using this and any other crane movement function, you must make sure that the crane does not collide with anything. Any collision is indicated by coloring the damaged piece red.
liftTo(z)	lifts the hook to the given z position. The maximal position is 53
lowerTo(z)	... lowers the hook to the given z position. The lowest position a container on the ship has is -4. The lowest on the dock is 6.
attachHook()	attaches the hook to the container it is placed at right now. If a container has a position (x,y,z), then the hook needs to be at that position.
releaseHook()	releases the container currently on the hook. If the container is not on a solid surface, it will come crashing down, causing damage (indicated by coloring the damaged piece red).
whatsOnTheHook()	Return which container (if any) is currently on the crane's hook

In-class Exercises

These exercises are designed to get comfortable with basic data types and control structures in Python. They also are steps towards your two assignments.

Exercise 1 (in class Tues 1)

Set the shipper of all containers whose y position is less than 0 (that is, the containers in the front of the ship) and that are in the lowest vertical position (have a vertical position of -4) as "ScrapNPort".

Exercise 2 (in-class Thu 1)

Write a function `moveCraneToXYZ(xcoord,ycoord,vertical-position)` that does the following:

- Lift the crane's hook to its heighest position
- Move the crane's hook to the `xcord,ycoord` position
- Lower the crane's hook to `vertical-position`

Each of these 3 steps are provided as functions. You just write a new function that calls them in order with the right parameters.

Write two lists of 3 containers that represent stacks of containers on a ship. Make sure that they are on the surface of the load area of the ship.

Exercise 3 (in-class Tue 2)

Write a list named `unload-places` that contains 15 positions (each a pair of `(x,y)` coordinates) pairs in a grid next to one another, where we can place containers on the dock.

Write a piece of code that moves the crane to the container with the name "First", picks it up and moves it to the first position in unload-places and releasing it there.

Write a loop that calls the function `moveCraneToXYZ(xcoord,ycoord,vertical-position)` for each container in a list, picking it up, moving and releasing it in a free position in `unload-places`.

Exercise 4 (in-class Tue 3)

Write and experiment with a program on readings lines from a text file and splitting it at a comma. Do this in your local Python installation. We will provide a sample file called `containers.csv`.

Exercise 5 (in-class Thu 3)

Extend your code from Exercise 4 by adding new containers to the dock scene based on the data from each line of a text file.

Exercise 6 (in-class Tue 4)

Write a loop that finds all containers from a certain shipper. Write another loop that finds all containers from a list that are highest up on the ship (that is, which do not have another container on top of them).