Objects in Space

You’ve been contracted by Interstellar Industries to help them determine the optimal number of robotic helpers to add to their newest spaceship. There are actually two kinds of robot they want to test: robots which fix damage to the deck of the ship and robots which fix other damaged robots. They will provide the dimensions of a rectangular deck, the starting locations of each kind of robot, and asteroid location and movement. They will evaluate the output to reach their conclusions.

Some additional information they provided:

1. Asteroids move in straight lines in the x or y direction from a starting point to an ending point
2. If an asteroid moves into a space occupied by a robot, it breaks the robot
3. Both robots follow a certain logic when they move:
   1. First they check the spaces in front, in back, to the left, and to the right for another object (an asteroid, another robot, or damage to the ship)
   2. If they can act on that object, they will take that action and then finish moving (ship repair robots will repair damage, robot repair robots will repair other robots)
   3. Otherwise they will pick a random direction (forward, backward, left, right) and move in it

Deliverables

Define base classes and derived classes for this problem. Assume the deck will be a list of lists that stores whether the deck is damaged or not (Boolean values). Assume there will be another list of lists to store robot positions and a third to store asteroid positions (this is so we can more easily represent an asteroid passing over robots and ship damage in the simulation).

Framework

Nouns (group things with similar characteristics):

Interstellar Industries, Robotic Helpers, Spaceship, Ship-Fixing Robot, Robot-Fixing Robot, Deck, Asteroid, starting point, End Point, Broken Robot, Damaged Deck, Deck-Tile

List all nouns that are core to the problem:

Robot, Ship-Fixing Robot, Robot-Fixing Robot, Asteroid, Broken Robot, Deck, Deck-Tile

List base classes:

Moving Object

Deck\_Tile

List your derived classes:

Robot ----🡪 Robot-Fixing Robot, Ship-Fixing Robot

Asteroid

Detail each class by providing attributes and methods:

**Class Name Attributes Methods**

Moving Object Have Coordinates, occupies space Get\_X(), Get\_Y(), Occupy()

Robot Can Move Randomly, Can Scan Areas, Can Break Move(), Scan(), Break(), a get\_status()

Robot-Fixing Robot Can Repair Robot Repair\_robot()

Ship-Fixing Robot Can Repair Ship Repair\_ship()

Asteroid Target Location, Move linear, Boolean exists determine\_target(), f move(), disintegrate()

Deck\_Tile Has a state (broken/unbroken) get\_status()

Y 3 My attempt at a robot-level grid, so grid\_list= [y0, y1, y2, y3…]

2 where y1 = [x1x,2,x3,x4,x5,x6] and y2 = [x1,x2,x3,x4,x5,x6]

1 So my coordinate for a robot at (3,2) = grid\_list[y2][x3] Right?

0 1 2 3 4 5 6

Define each class in Python code (this step may be done on a computer):

### Moving Object Class

class Moving\_object():

### Used as a base class, cannot stand independant

def \_\_init\_\_(self, x\_coordinate, y\_coordinate):

self.\_x = x\_coordinate

self.\_y = y\_coordinate

def \_\_str\_\_(self):

return "You should never see this"

def get\_coordinates(self):

new\_coordinates = self.\_x + ", " + self.\_y

print(new\_coordinates)

def update\_grid(self):

### I'm not really sure how I would go about updating the grid where

### the new Moving\_object exists at. Only thing I can think of is a for-

### loop like the one which follows.

global grid\_list

for i in range(len(grid\_list)):

if gridlist[i] == self.\_y:

for j in gridlist[i][j]:

if gridlist[i][j] == self.\_x:

gridlist[i][j] = self.\_x + "\_occupied"

print("Grid has been updated.")

robot1 = Moving\_object("x2","y4")

robot1.get\_coordinates()

robot1.update\_grid()