* Classes
  + Environment
    - Number of grid rows
      * int
    - Number of grid columns
      * int
    - Number of robots
      * int
    - Number of packages
      * Int
    - Location of picking station
  + State
    - Init() -> state
      * Calls initial state()
    - Robot coordinates
      * ix2 int array (if i robots)
    - Stack coordinates
      * jx2 int array (if j stacks)
    - Carry stack boolean values
      * Size i boolean list (if i robots)
    - Ordered boolean values
      * Size j boolean list (if j stacks)
    - enumerate\_state(self) -> int
  + Action
    - Actions for all robots to take
      * Size i list of strings
    - enumerate\_action(self) -> int
* Functions
  + Initial\_state() -> state
    - Creates a state object
    - Randomly chooses 20 of the 50 coordinates for stacks to occupy
    - Randomly chooses 10 of the 50 coordinates for robots to occupy
    - Sets all carry boolean values to False
    - Calls order\_items()
    - Returns state object
  + order\_items(state) -> state
    - See if a random number is less than 0.25
    - If so then random choose a number from 1...20, this number determines which stack has an item that was ordered, chance state accordingly and return the state
    - If not hen return the original state
  + policy(state) -> action
    - Since we are not implementing the q-learning alg yet, just pick a random action
    - Pick a random action, calculate new state, check if the state transition is possible
    - If the state transition is possible, return it
    - If not, pick a new random action and try again
  + calculate\_state(state, action) -> state
    - Determine which new state will be entered if taking this action in the current state
    - Returns new state object
  + possible\_state(state1, state2) -> boolean (sang)
    - Checks if transitioning from state1 to state2 is possible
    - Check if robots are outside the grid in state2
    - Check if 2 robots are in the same location in state2
    - Check if 2 stacks are in the same location in state2
    - Check if 2 robots moved through one another (I.e. switched locations transitioning from state1 to state2
    - Return a boolean value if its available
  + cost\_function(state1, state2) -> int
    - Determine the reward signal for a particular state transition
  + Explore() -> boolean
    - Determines whether to explore or exploit
    - **Not needed yet**
  + update\_qvalue(state, action, reward, Qtable) -> Qtable
    - Updates Q-value in Q-table
    - **Not needed yet**