```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
environment for 2D navigation with intermediate goals 0, 1, and 2+
"""
import numpy  as np
import sys

class Env2D():
    def __init__(self, size=5, succ_rew=10, tsteprew=-0.01, subgoals=1, seed=0,
stoch=0.0):
        np.random.seed(seed)
        if subgoals > 2 or subgoals < 0:
            sys.exit("Can have at least 0 and at most 2 subgoals (asked for " +
str(subgoals) +")")
        self.size     = size
        self.stoch    = stoch
        if stoch < 0 or stoch > 1:
            sys.exit("Stochasticity " + str(stoch) + " is outside acceptable range [0,1]")
        self.subgoals = subgoals
        self.tsteprew = tsteprew
        self.succ_rew = succ_rew
        self.initialise_map(size, subgoals)
        self.x        = 0 # x location
        self.y        = 0 # y location
        self.v1       = 0 # Whether first subgoal has been visited in this episode 0/1
        self.v2       = 0 # Whether second subgoal has been visited in this episode 0/1

    def initialise_map(self, size, subgoals):
        self.fix_cells = np.zeros((size,size))
        #if subgoals > 0:
        self.fix_cells[0 ,size-1]     = 1  # First subgoal
        #if subgoals > 1:
        self.fix_cells[size-1, 0]     = 2  # Second subgoal
        self.fix_cells[size-1,size-1] = 10 # Final goal

    def reset(self):
        self.x  = 0
        self.y  = 0
        self.v1 = 0
        self.v2 = 0
        self.initialise_map(self.size, self.subgoals)

    def get_state(self):
        return [self.x/(self.size-1),
                self.y/(self.size-1),
                self.v1,
                self.v2]

    def step(self,action):
        if np.random.random() < self.stoch:
            # Generate a random action
            action   = np.random.randint(0,4)
        if action not in [0,1,2,3]:
            sys.exit("Invalid action: " + str(action))
        if action   == 0:
            # Up
            self.x = max(self.x-1,0)
        elif action == 1:
            # Right
            self.y = min(self.y+1,self.size-1)
        elif action == 2:
            # Down
            self.x = min(self.x+1,self.size-1)
        else:
            # Left
```

```
            self.y = max(self.y-1,0)
        steprew    = self.tsteprew
        done       = False
        if self.fix_cells[self.x,self.y] == 1 : self.v1 = 1 # Reached first subgoal
        if self.fix_cells[self.x,self.y] == 2 : self.v2 = 1 # Reached second subgoal
        if self.fix_cells[self.x,self.y] == 10:
            # Reached goal state, terminate the episode
            done     = True
            if self.subgoals == 0:
                steprew += self.succ_rew
            if self.subgoals == 1:
                if self.v1 == 0:
                    steprew += 1
                else:
                    steprew += self.succ_rew
            if self.subgoals == 2:
                if self.v1 + self.v2 == 0:
                    steprew += 1
                elif self.v1 + self.v2 == 1:
                    steprew += 2
                else:
                    steprew += self.succ_rew
            #print("Final state: " + str(self.get_state()) + ", final reward = " +
str(np.round(steprew,decimals=3)))
        return done, steprew
```