

Concurrency.py

```

1: # Anthony Tesoriero, Joseph Salemo, Joshua Lunsk, October 17 2019, Concurrency
2: # Status: Mostly Completed
3:
4: # Works for n = 3 and n = 4
5:
6: # Idea Notes
7: # Size = n (3 for example); CurrentPosition = pos
8: # North = -n; South = +n; East = +1; West = -1
9:
10: # -----
11:
12: import random
13:
14:
15: # ----- Initialization ----- #
16: class Concurrency:
17:
18:     def __init__(self):
19:
20:         # Size of grid (n x n)
21:         # self.n = int(input("What is the size of your grid? (n x n) "))
22:         self.n = 3
23:         print("Set to (", self.n, "x", self.n, ") grid.")
24:         print()
25:         print("1. Spiral Run: Agents start in opposite corners, and move cou
26:         print("2. Slither Run: Agents start in opposite corners, and move in
27:         self.version = int(input("Would you like to run 1 or 2? "))
28:
29:         # self.version = 2
30:
31:         self.moves = 0
32:
33:         # Initialize agents [Name, space, current direction]
34:         self.agent1 = ["Agent 1", 1, 3]
35:         self.agent2 = ["Agent 2", self.n**2, 1]
36:         self.foundGems = 0
37:
38:         # Gets 4 unique random nums
39:         rands = []
40:         for i in range(15):
41:             rand = random.randint(1, self.n ** 2)
42:             if rand not in rands:
43:                 rands.append(rand)
44:
45:         # List of gems [name, position]
46:         self.gems = [{"an emerald", rands[0]}, {"a crown", rands[1]}, {"a co
47:
48:         self.setType()
49:
50:         # ----- Starting Functions ----- #
51:
52:     def setType(self):
53:         # Runs spiralRun when version 1 is chosen
54:         if self.version is 1:
55:             self.printHeader("Spiral Run")
56:             self.spiralRun()
57:         # Runs slitherRun when version 2 is chosen
58:         else:
59:             self.printHeader("Slither Run")
60:             self.slitherRun()
61:
62:     def printHeader(self, runName):
63:         print()

```

```

64:         print("-----")
65:         print("----- Running", runName, "-----")
66:         print("-----")
67:         print()
68:
69:         # ----- Gem Functions ----- #
70:
71:         # Finds gem with agent
72:         def findGem(self, gem, agent):
73:             self.foundGems += 1
74:             print(agent[0], "found", gem[0], "in room", gem[1])
75:             print(4 - self.foundGems, "gems left.")
76:             gem[1] = -1
77:
78:         # Find if agent position equals gems
79:         def gemCheck(self, agent):
80:             for gem in self.gems:
81:                 if agent[1] is gem[1]:
82:                     self.findGem(gem, agent)
83:
84:         # ----- Movement Functions ----- #
85:
86:         # Checks if direction attempt is valid
87:         # dir: 1 = north, 2 = east, 3 = south, 4 = west
88:         def checkDir(self, agent, dir):
89:             # North
90:             if dir is 1 and agent[1] - self.n < 1:
91:                 return False
92:             # East
93:             if dir is 2 and agent[1] % self.n is 0:
94:                 return False
95:             # South
96:             if dir is 3 and agent[1] + self.n > self.n ** 2:
97:                 return False
98:             # West
99:             if dir is 4 and (agent[1] - 1) % self.n is 0:
100:                 return False
101:             # No failures
102:             return True
103:
104:         # Turn agent left
105:         def turnLeft(self, agent):
106:             # North turn west
107:             if agent[2] is 1:
108:                 agent[2] = 4
109:             # East turn north
110:             elif agent[2] is 2:
111:                 agent[2] = 1
112:             # South turn east
113:             elif agent[2] is 3:
114:                 agent[2] = 2
115:             # West turn south
116:             else:
117:                 agent[2] = 3
118:
119:         # Turn agent right
120:         def turnRight(self, agent):
121:             # North turn east
122:             if agent[2] is 1:
123:                 agent[2] = 2
124:             # East turn south
125:             elif agent[2] is 2:
126:                 agent[2] = 3

```

Concurrency.py

```
127:         # South turn west
128:         elif agent[2] is 3:
129:             agent[2] = 4
130:         # West turn north
131:         else:
132:             agent[2] = 1
133:
134:     # Moves agent
135:     # dir: 1 = north, 2 = east, 3 = south, 4 = west
136:     def moveForward(self, agent):
137:         self.gemCheck(agent)
138:         if agent[2] is 1 and self.checkDir(agent, agent[2]):
139:             agent[1] -= self.n
140:         elif agent[2] is 2 and self.checkDir(agent, agent[2]):
141:             agent[1] += 1
142:         elif agent[2] is 3 and self.checkDir(agent, agent[2]):
143:             agent[1] += self.n
144:         elif agent[2] is 4 and self.checkDir(agent, agent[2]):
145:             agent[1] -= 1
146:
147:     # Moves both agent1 and agent 2
148:     def moveBoth(self):
149:         self.moveForward(self.agent1)
150:         self.moveForward(self.agent2)
151:         self.moves += 1
152:
153:     # Turns both agent1 and agent2
154:     # dir: 0 for left, 1 for right
155:     def turnBoth(self, dir):
156:         if dir is 0:
157:             self.turnLeft(self.agent1)
158:             self.turnLeft(self.agent2)
159:         elif dir is 1:
160:             self.turnRight(self.agent1)
161:             self.turnRight(self.agent2)
162:
163:     # ----- Spiral Moving for Version 1 ----- #
164:
165:     # Run in a counter-clockwise spiral
166:     def spiralRun(self):
167:
168:         dec = 0
169:
170:         while dec != self.n-1:
171:             for i in range(self.n-1 - dec):
172:                 self.moveBoth()
173:             dec += 1
174:             self.turnBoth(0)
175:
176:         self.moveBoth()
177:         self.gemCheck(self.agent1)
178:         self.gemCheck(self.agent2)
179:
180:         if self.foundGems is 4:
181:             print("All gems found in", self.moves, "moves!")
182:         else:
183:             print("FAILED")
184:             print(self.gems)
185:
186:     # ----- Slither Moving for Version 2 ----- #
187:
188:     # Run in a slither snaking pattern
189:     def slitherRun(self):
```

```
190:
191:         dir = 0
192:
193:         while self.foundGems is not 4: # and runSafety < 20:
194:             # Loop for each column
195:             for i in range(self.n):
196:                 # More forward until wall
197:                 for i in range(self.n-1):
198:                     self.moveBoth()
199:                 # Mod to return 0 or 1, meaning turn left or right.
200:                 self.turnBoth(dir % 2)
201:                 self.moveBoth()
202:                 self.gemCheck(self.agent1)
203:                 self.gemCheck(self.agent2)
204:                 self.turnBoth(dir % 2)
205:                 dir += 1
206:
207:             self.turnBoth(1)
208:             self.turnBoth(1)
209:
210:             if self.foundGems is 4:
211:                 print("All gems found in", self.moves, "moves!")
212:             else:
213:                 print("FAILED")
214:                 print(self.gems)
215:
216:     # ----- Driver ----- #
217:
218:     # Run Concurrency
219:     game = Concurrency()
```