University of Puerto Rico
Mayagüez Campus
Electrical & Computer Engineering Department

Agents Report

- Group A -
Daniel Mestres Piñero
Natanael Santiago Morales
Leonel Osoria Toledo

ICOM5015 - 001D
March 6, 2023

Introduction

An agent is anything that can be viewed as receiving data from its environment through sensors and acting upon that environment through actuators [1]. Agents can be humans, robots, and even software agents, which will be utilized in this report. An agent's rationality depends on the performance measure that defines success, prior knowledge of the environment, the actions that the agent can perform and the agent's percept sequence to date [1]. An artificial intelligence problem, as the one we seek to solve, is defined by PEAS, which has 4 components being [1]:

1. Performance measure
2. Environment
3. Actuators
4. Sensors

The purpose of this report is to implement a software agent and an environment to compare different types of agents and their performance in said environments. The following section contains the exercises aimed to be completed. With these exercises, we aim to answer questions such as: "Can a reflex agent be perfectly rational?", "Can a random agent outperform a simple reflex agent?" and "Can a reflex agent outperform a random agent?".

Exercises

2.11 - Implement a performance-measuring environment simulator for the vacuum-cleaner world depicted in Figure 1. Implementation should be modular so that the sensors, actuators, and environment characteristics (size, shape, dirt placement, etc.) can be changed easily.

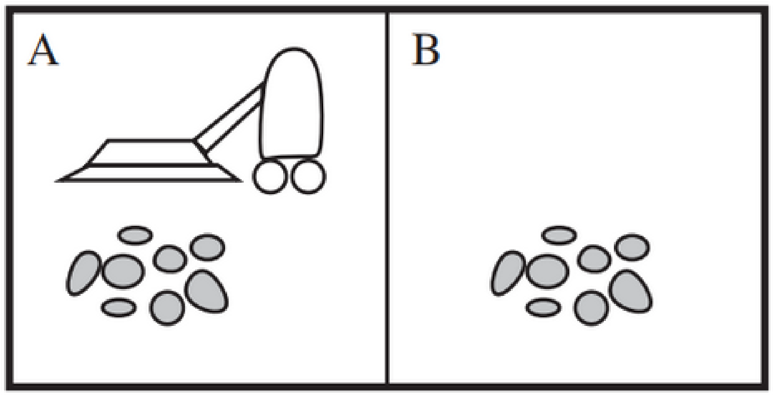


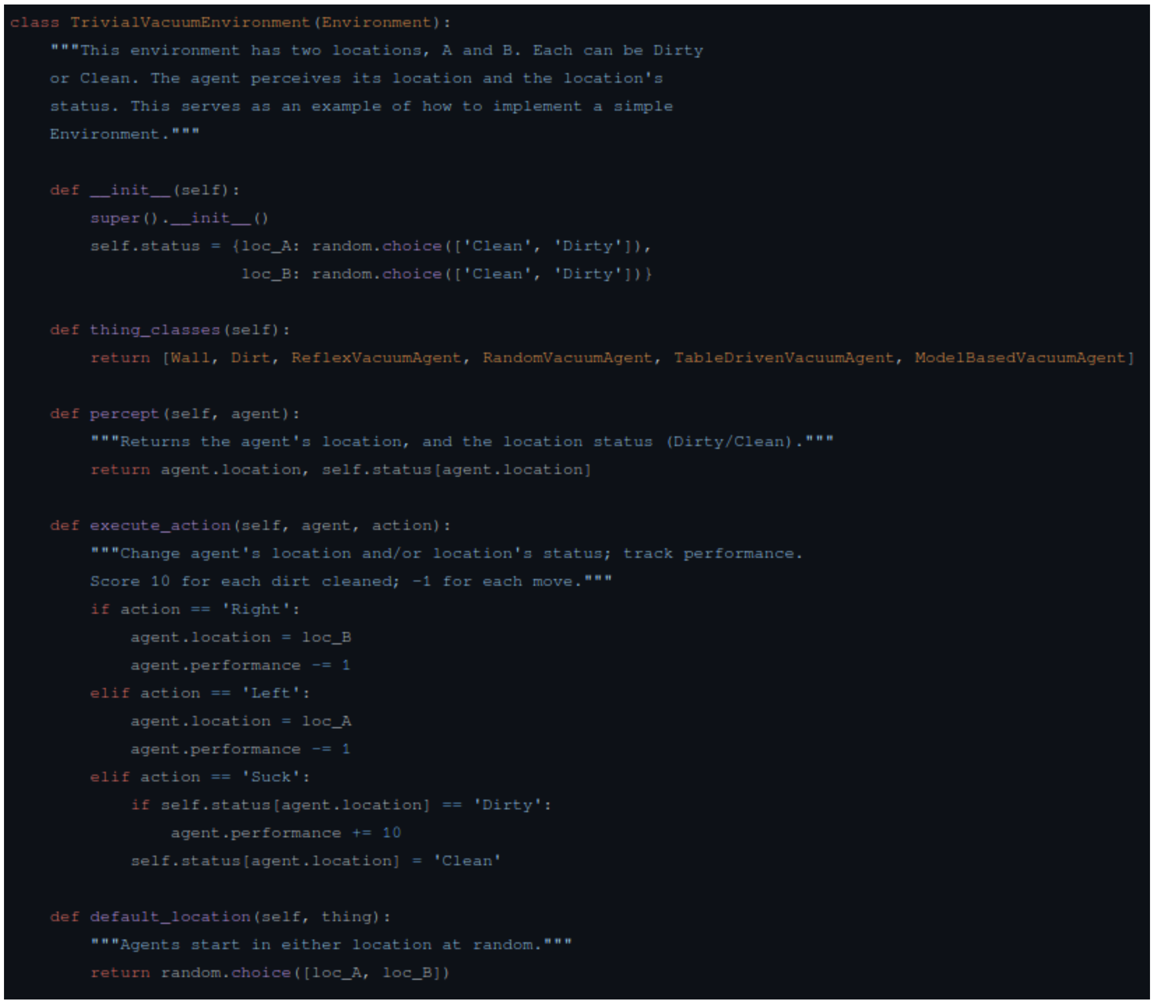
Figure 1: Vacuum-cleaner world with two locations [1]

2.14 - Consider a modified version of the vacuum environment in Exercise 2.11 in which the geography of the environment—its extent, boundaries, and obstacles—is unknown, as is the initial dirt configuration. (The agent can go Up and Down as well as Left and Right.)

1. Can a simple reflex agent be perfectly rational for this environment? Explain.
2. Can a simple reflex agent with a randomized agent function outperform a simple reflex agent? Design such an agent and measure its performance on several environments.
3. Can you design an environment in which your randomized agent will perform poorly? Show your results.
4. Can a reflex agent with state outperform a simple reflex agent? Design such an agent and measure its performance on several environments. Can you design a rational agent of this type?

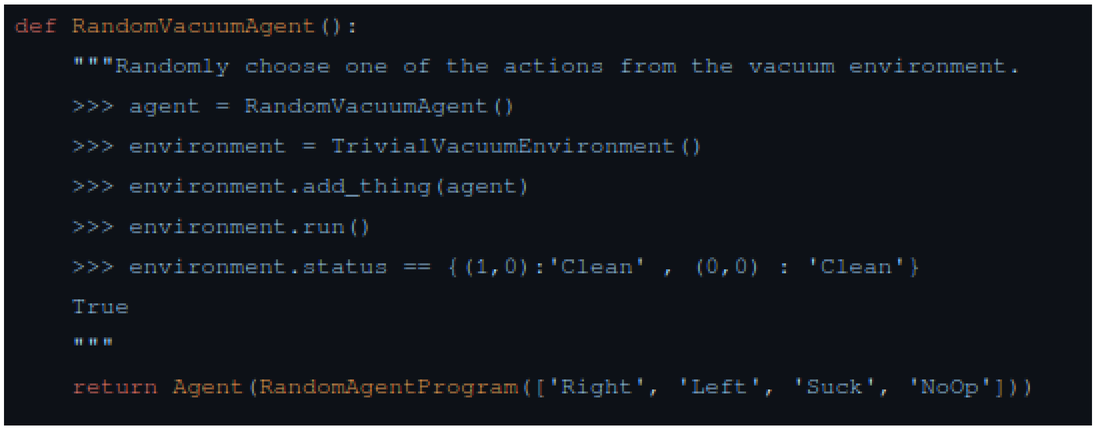
Procedure

For both exercises to be implemented, the UC Berkeley code repository [2] was utilized, specifically its agents file, as it already provides an excellent and convenient implementation for both different types of agents and environments. For the 2.11 exercise, the trivial vacuum environment implementation was used to simulate the environment shown in figure 1, where only 2 locations are present and their status, clean or dirty, is randomly decided at run-time. The agent in this environment has only 4 actions, these being: Right, Left, Suck, and NoOp. The trivial vacuum environment measures performance by subtracting 1 with every “left” or “right” action and adding 10 for every “suck” action, NoOp having no effect on the performance measure. The optimal performance is assumed to be 19 when both locations are dirty, 10 when only one location is dirty and the agent starts in said location and 9 when only one location is dirty but the agent starts in the other location. The cases when the environment starts with both locations clean were ignored, as their performance measures would be 0. The implementation for the environment and agents utilized can be seen in the next figures:



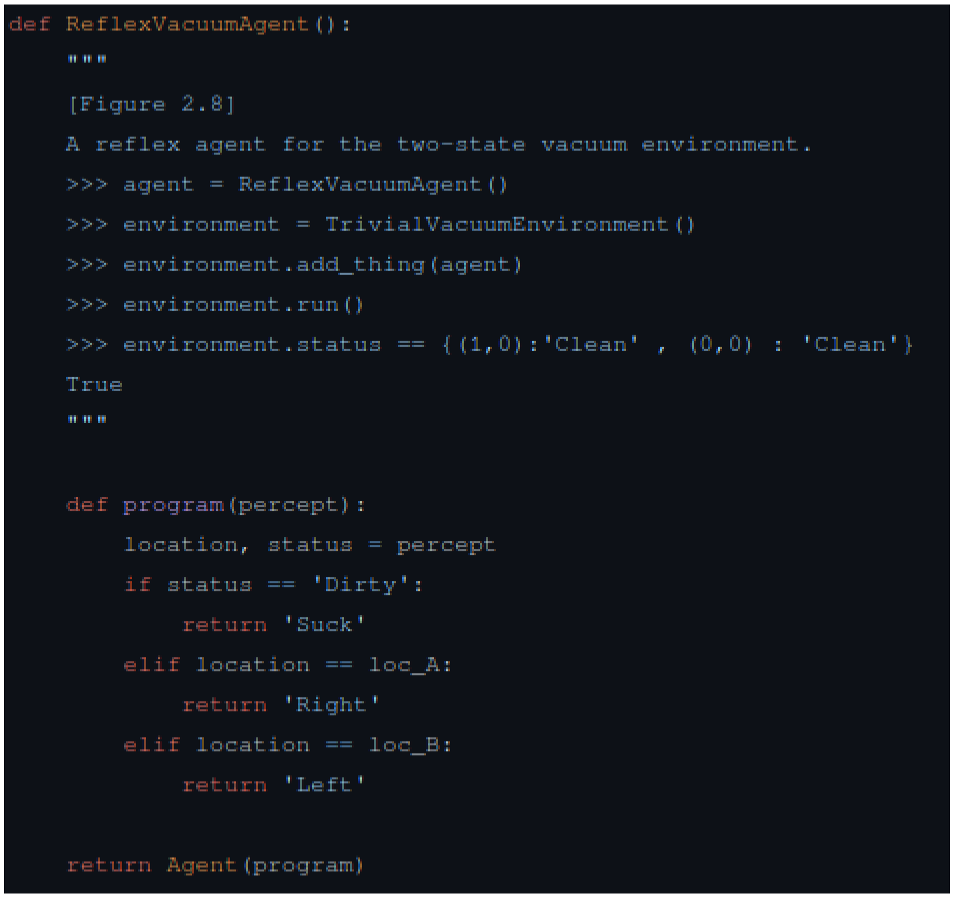
```python
class TrivialVacuumEnvironment(Environment):
    """This environment has two locations, A and B. Each can be Dirty
    or Clean. The agent perceives its location and the location's
    status. This serves as an example of how to implement a simple
    Environment."""

    def __init__(self):
        super().__init__()
        self.status = {loc_A: random.choice(['Clean', 'Dirty']),
                       loc_B: random.choice(['Clean', 'Dirty'])}

    def thing_classes(self):
        return [Wall, Dirt, ReflexVacuumAgent, RandomVacuumAgent, TableDrivenVacuumAgent, ModelBasedVacuumAgent]

    def percept(self, agent):
        """Returns the agent's location, and the location status (Dirty/Clean)."""
        return agent.location, self.status[agent.location]

    def execute_action(self, agent, action):
        """Change agent's location and/or location's status; track performance.
        Score 10 for each dirt cleaned; -1 for each move."""
        if action == 'Right':
            agent.location = loc_B
            agent.performance -= 1
        elif action == 'Left':
            agent.location = loc_A
            agent.performance -= 1
        elif action == 'Suck':
            if self.status[agent.location] == 'Dirty':
                agent.performance += 10
            self.status[agent.location] = 'Clean'

    def default_location(self, thing):
        """Agents start in either location at random."""
        return random.choice([loc_A, loc_B])
```

Figure 2: Trivial Environment [2]



```
def RandomVacuumAgent():
    """Randomly choose one of the actions from the vacuum environment.
    >>> agent = RandomVacuumAgent()
    >>> environment = TrivialVacuumEnvironment()
    >>> environment.add_thing(agent)
    >>> environment.run()
    >>> environment.status == {(1,0):'Clean' , (0,0) : 'Clean'}
    True
    """
    return Agent(RandomAgentProgram(['Right', 'Left', 'Suck', 'NoOp']))
```

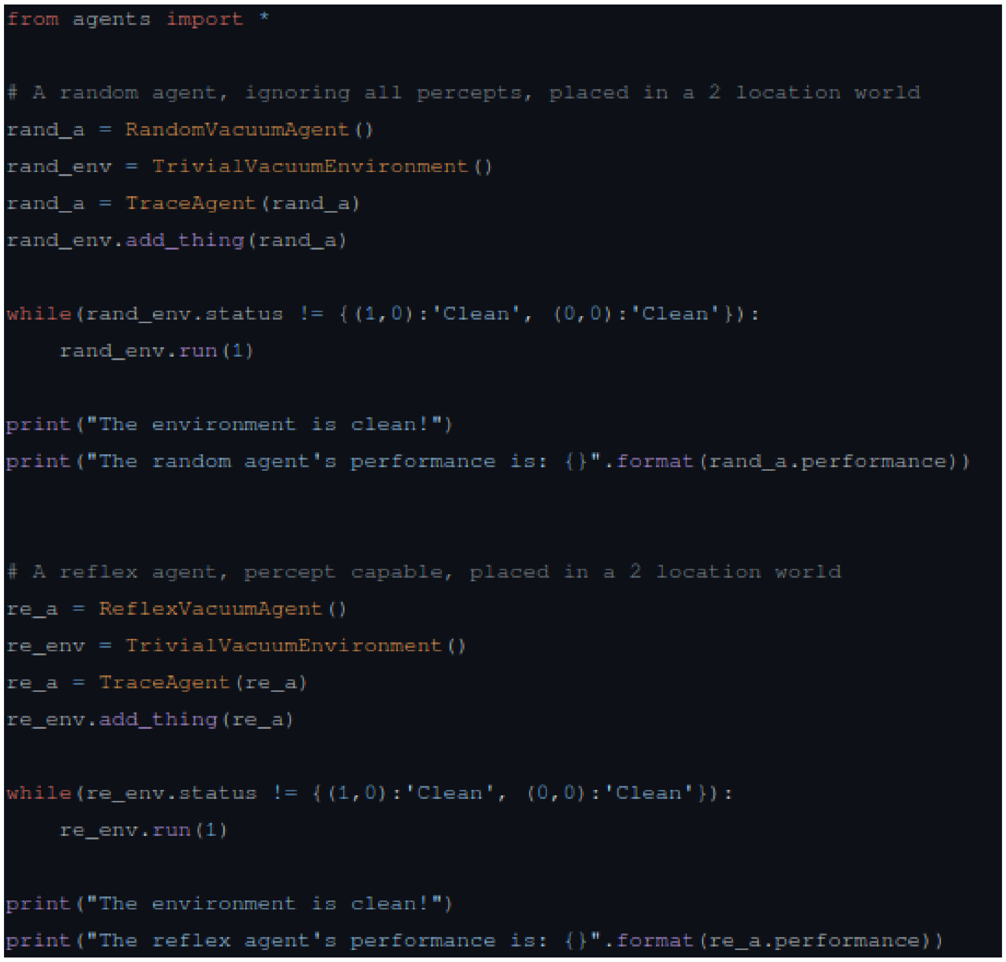
Figure 3: Random Agent [2]



```
def ReflexVacuumAgent():
    """
    [Figure 2.8]
    A reflex agent for the two-state vacuum environment.
    >>> agent = ReflexVacuumAgent()
    >>> environment = TrivialVacuumEnvironment()
    >>> environment.add_thing(agent)
    >>> environment.run()
    >>> environment.status == {(1,0):'Clean' , (0,0) : 'Clean'}
    True
    """

    def program(percept):
        location, status = percept
        if status == 'Dirty':
            return 'Suck'
        elif location == loc_A:
            return 'Right'
        elif location == loc_B:
            return 'Left'

    return Agent(program)
```

Figure 4: Reflex Agent [2]

A simple program was developed to measure the performance of both agents in the trivial vacuum environment, where the agents continue to execute actions until the environment is completely clean. The next figure illustrates this program and the following table provides performance measures obtained for both agents after running the program several times.



```python
from agents import *

# A random agent, ignoring all percepts, placed in a 2 location world
rand_a = RandomVacuumAgent()
rand_env = TrivialVacuumEnvironment()
rand_a = TraceAgent(rand_a)
rand_env.add_thing(rand_a)

while(rand_env.status != {(1,0):'Clean', (0,0):'Clean'}):
    rand_env.run(1)

print("The environment is clean!")
print("The random agent's performance is: {}".format(rand_a.performance))


# A reflex agent, percept capable, placed in a 2 location world
re_a = ReflexVacuumAgent()
re_env = TrivialVacuumEnvironment()
re_a = TraceAgent(re_a)
re_env.add_thing(re_a)

while(re_env.status != {(1,0):'Clean', (0,0):'Clean'}):
    re_env.run(1)

print("The environment is clean!")
print("The reflex agent's performance is: {}".format(re_a.performance))
```

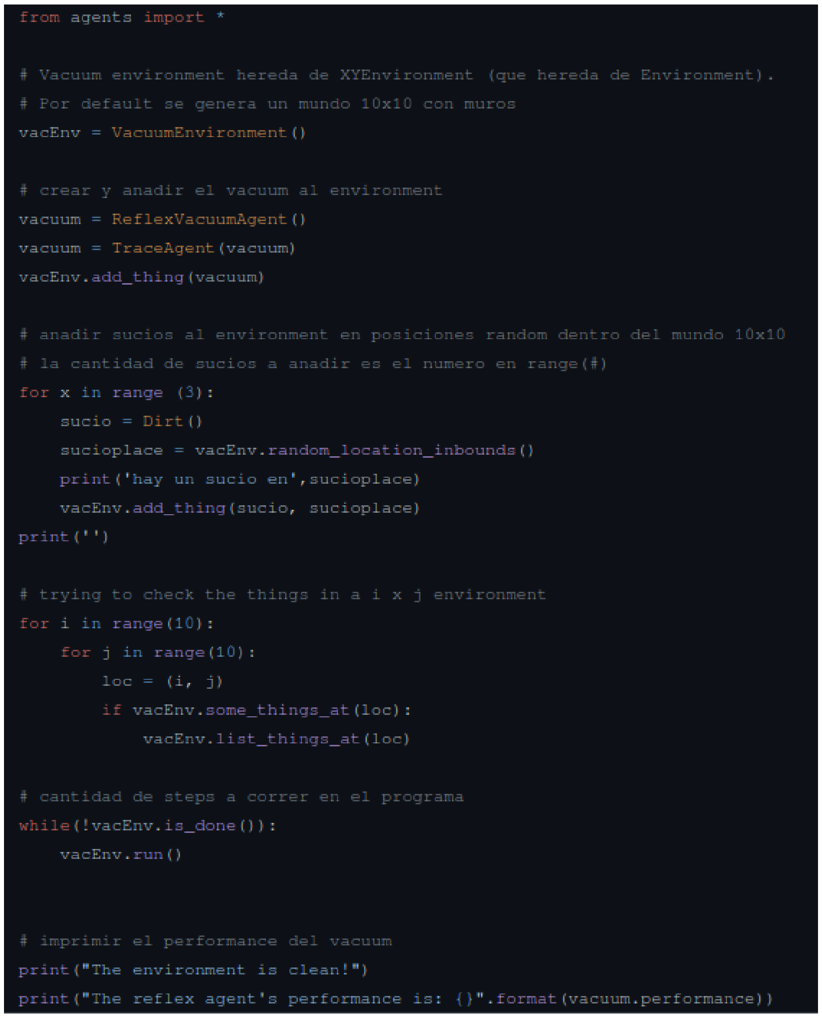
Figure 5: Random vs. Reflex Agent Program

TABLE I
RANDOM VS. REFLEX AGENT TRIVIAL PERFORMANCE

Test	Random Performance	Reflex Performance
1	8	9
2	-4	9
3	3	9
4	9	9
5	9	19

6	-12	9
7	8	9
8	9	9
9	9	9
10	4	10

Continuing with the 2.14 exercise, a 2D vacuum environment was attempted to be implemented utilizing the UC Berkeley code repository once again. The plan was to implement a 10x10 environment with randomly placed “Dirt” tiles for the agent to find and clean. However the team was unable to implement a reflex or random agent that operated in a 2 dimensional environment. The following figure shows our attempt at the exercise:



```python
from agents import *

# Vacuum environment hereda de XYEnvironment (que hereda de Environment).
# Por default se genera un mundo 10x10 con muros
vacEnv = VacuumEnvironment()

# crear y anadir el vacuum al environment
vacuum = ReflexVacuumAgent()
vacuum = TraceAgent(vacuum)
vacEnv.add_thing(vacuum)

# anadir sucios al environment en posiciones random dentro del mundo 10x10
# la cantidad de sucios a anadir es el numero en range(#)
for x in range (3):
    sucio = Dirt()
    sucioplace = vacEnv.random_location_inbounds()
    print('hay un sucio en',sucioplace)
    vacEnv.add_thing(sucio, sucioplace)
print('')

# trying to check the things in a i x j environment
for i in range(10):
    for j in range(10):
        loc = (i, j)
        if vacEnv.some_things_at(loc):
            vacEnv.list_things_at(loc)

# cantidad de steps a correr en el programa
while(!vacEnv.is_done()):
    vacEnv.run()


# imprimir el performance del vacuum
print("The environment is clean!")
print("The reflex agent's performance is: {}".format(vacuum.performance))
```

Figure 6: Second Exercise Attempt

Analysis

Analyzing the data obtained in Table I, we can appreciate how the reflex agent's performance is always optimal in the trivial vacuum environment, obtaining the optimal value in the three different cases where a dirty location is present. In this trivial vacuum environment, a simple reflex agent can be considered perfectly rational, as demonstrated by the data obtained in table I. An interesting observation is the random agent obtaining optimal values, similarly to the reflex agent, but these performances are attributed to simple luck, as the agent has a 25% chance for each decision to be the correct one given the circumstances. Nevertheless, we can consider that both the random agent and reflex agent can be perfectly rational in the trivial vacuum environment world.

Continuing with the second exercise, we note how a simple reflex agent cannot be perfectly rational in an environment with unknown extent, boundaries, and obstacles as in order to be perfectly rational the environment must be fully observable and the agent must make decisions based on past decisions. Neither of these conditions are achieved with the environment and simple reflex agent implemented. The agent would simply get stuck when it encounters a wall. Carrying on, in theory a simple reflex agent with a randomized agent function could outperform a simple reflex agent, as the random agent would avoid getting stuck against an obstacle. However the randomized agent would in theory perform very poorly in 2D environments with long straight paths, only possible moves being up and down, or left and right. This is due to half its possible moves resulting in not moving at all, subtracting from its performance. Analyzing the fourth and final question, if a reflex agent has knowledge of its state it can avoid getting stuck against an obstacle and be much more efficient in making decisions. This in theory would make it outperform a simple reflex agent without state.

Conclusion

To conclude the report, we note how a simple reflex agent is always perfectly rational in the trivial vacuum environment, having only 2 locations, with a state of dirty or clean. The random agent on the other hand, produces inconsistent results, although it occasionally produces perfectly rational results. Clearly the simple reflex agent is the winner in this environment. However in the 2D world, the reflex agent would in theory have problems with obstacles and the random agent would have problems with long straight paths, as it would result in not moving roughly 50% of the time. In theory, the clear winner in this environment would be a reflex agent with state, as it potentially avoids getting stuck on an obstacle and could navigate long straight paths efficiently. Although the second exercise could not be implemented to get quantifiable data to compare the two agents, we could extrapolate from the behavior the agents would have and answer the questions of: “Can a simple reflex agent with a randomized agent function outperform a simple reflex agent?”, “Can you design an environment in which a randomized agent would perform poorly?”, and “Can a reflex agent with state outperform a simple reflex agent?”.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Hoboken, NJ: Pearson, 2021.

[2] aimacode (2016) aima-python [Source code]. https://github.com/aimacode/aima-python