simple_randomized_agent-Aditya

September 19, 2021

1 Intelligent Agents: Reflex Agents for the Vacuum-cleaner World

1.1 Instructions

Total Points: undergrad 10, graduate students 11

Complete this notebook and submit it. The notebook needs to be a complete project report with

- your implementation (you can use libraries like math, numpy, scipy, but not libraries that implement inteligent agents or search algorithms),
- documentation including a short discussion of how your implementation works and your design choices, and
- experimental results (e.g., tables and charts with simulation results) with a short discussion of what they mean.

Use the provided notebook cells and insert additional code and markdown cells as needed.

1.2 Introduction

In this assignment you will implement a simulator environment for an automatic vacuum cleaner robot, a set of different reflex agent programs, and perform a comparison study for cleaning a single room. Focus on the **cleaning phase** which starts when the robot is activated and ends when the last dirty square is cleaned. Someone else will take care of the agent program needed to navigate back to the charging station after the room is clean.

1.3 PEAS description of the cleaning phase

Performance Measure: Each action costs 1 energy unit. The performance is measured as the sum of the energy units used to clean the whole room.

Environment: A room with $n \times n$ squares where $n = 5$. Dirt is randomly placed on each square with probability $p = 0.2$. For simplicity, you can assume that the agent knows the size and the layout of the room (i.e., it knows n). To start, the agent is placed on a random square.

Actuators: The agent can clean the current square (action `suck`) or move to an adjacent square by going `north`, `east`, `south`, or `west`.

Sensors: Four bumper sensors, one for north, east, south, and west; a dirt sensor reporting dirt in the current square.

1.4 The agent program for a simple randomized agent

The agent program is a function that gets sensor information (the current percepts) as the arguments. The arguments are:

- A dictionary with boolean entries for the for bumper sensors `north`, `east`, `west`, `south`. E.g., if the agent is on the north-west corner, `bumpers` will be `{"north" : True, "east" : False, "south" : False, "west" : True}`.
- The dirt sensor produces a boolean.

The agent returns the chosen action as a string.

Here is an example implementation for the agent program of a simple randomized agent:

[1]:
```python
import numpy as np

actions = ["north", "east", "west", "south", "suck"]

def simple_randomized_agent(bumpers, dirty):
    return np.random.choice(actions)
```

[2]:
```python
# define percepts (current location is NW corner and it is dirty)
bumpers = {"north" : True, "east" : False, "south" : False, "west" : True}
dirty = True

# call agent program function with percepts and it returns an action
simple_randomized_agent(bumpers, dirty)
```

[2]:
```
'south'
```

Note: This is not a rational intelligent agent. It ignores its sensors and may bump into a wall repeatedly or not clean a dirty square. You will be asked to implement rational agents below.

1.5 Simple environment example

We implement a simple simulation environment that supplies the agent with its percepts. The simple environment is infinite in size (bumpers are always `False`) and every square is always dirty, even if the agent cleans it. The environment function returns a performance measure which is here the number of cleaned squares (since the room is infinite and all squares are constantly dirty, the agent can never clean the whole room as required in the PEAS description above). The energy budget of the agent is specified as `max_steps`.

[3]:
```python
def simple_environment(agent, max_steps, verbose = True):
    num_cleaned = 0

    for i in range(max_steps):
        dirty = True
        bumpers = {"north" : False, "south" : False, "west" : False, "east" :␣
↪False}
```

```
        action = agent(bumpers, dirty)
        if (verbose): print("step", i , "- action:", action)

        if (action == "suck"):
            num_cleaned = num_cleaned + 1

    return num_cleaned
```

Do one simulation run with a simple randomized agent that has enough energy for 20 steps.

[4]:
```
simple_environment(simple_randomized_agent, max_steps = 20)
```

```
step 0 - action: east
step 1 - action: suck
step 2 - action: west
step 3 - action: north
step 4 - action: north
step 5 - action: west
step 6 - action: north
step 7 - action: suck
step 8 - action: north
step 9 - action: suck
step 10 - action: east
step 11 - action: west
step 12 - action: north
step 13 - action: west
step 14 - action: east
step 15 - action: north
step 16 - action: suck
step 17 - action: north
step 18 - action: north
step 19 - action: north
```

[4]:
```
4
```

2 Tasks

Submission Instructions: Use this notebook to prepare your submission. Complete this section with your code and results. You can add additional Markdown blocks for your description, comments in the code and use mathplotlib to produce charts.

Note: Try to keep the code simple! In this course, we want to learn about the algorithms and we often do not need to use object-oriented design.

2.1 Task 1: Implement a simulation environment [2 Points]

The simple environment above is not very realistic. Your environment simulator needs to follow the PEAS description from above. It needs to:

- Initialize the environment by storing the state of each square (clean/dirty) and making some dirty.
- Keep track of the agent's position.
- Call the agent function repeatedly and provide the agent function with the sensor inputs.

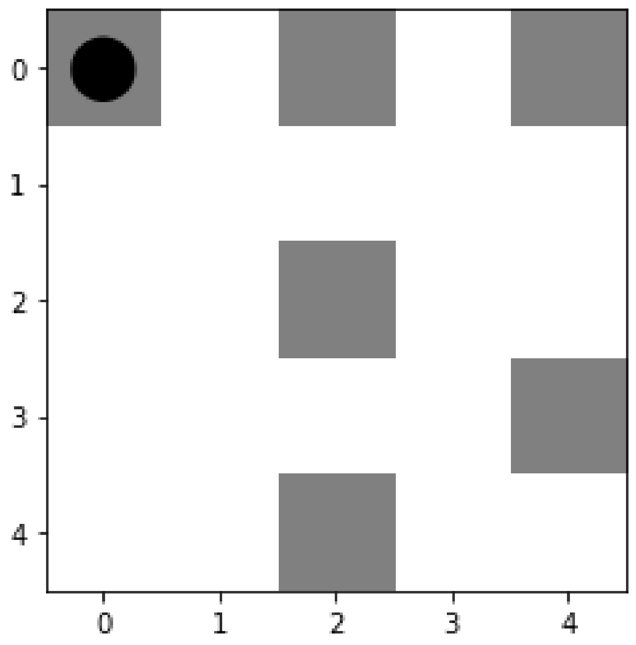
- React to the agent's actions. E.g, by removing dirt from a square or moving the agent around unless there is a wall in the way.
- Keep track of the performance measure. That is, track the agent's actions until all dirty squares are clean and count the number of actions it takes the agent to complete the task.

The easiest implementation for the environment is to hold an 2-dimensional array to represent if squares are clean or dirty and to call the agent function in a loop until all squares are clean or a predefined number of steps have been reached (i.e., the robot runs out of energy).

The simulation environment needs to work with the simple randomized agent program from above and then it can be used for your agent implementation in the tasks below.

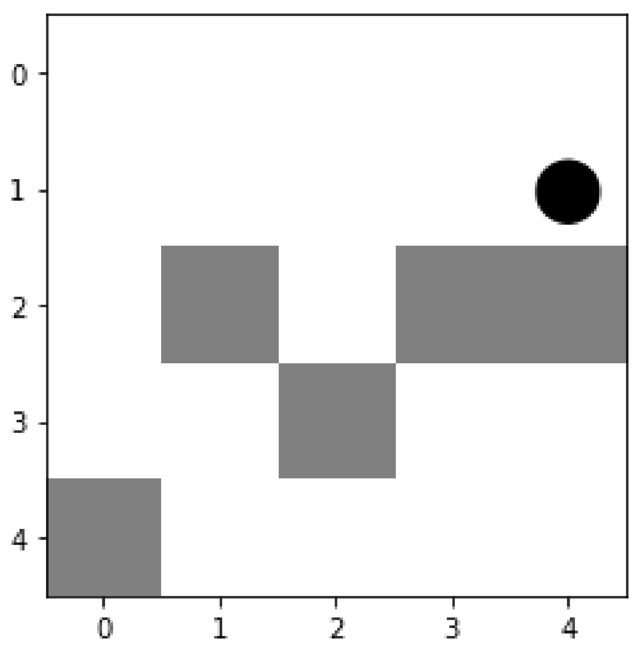
[5]:
```python
# Your code and description goes here
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import colors

#Code to visualize the room in X-scale and Y-scale and squares as the area␣
↪robot scans for clean or dirty.

def show_room(room, pos, fontsize = 24):
    """display room and robot

    Parameters
    ----------
    room : bool 2d array
        Dirt locations in the room.
    pos : list of 2 int values
        x and y location of the robot.
    fint_size : int
        size of the robot symbol.
    """

    cmap = colors.ListedColormap(['white', 'gray'])

    room = np.copy(room)
    room = room.astype(np.int64)

    fig, ax = plt.subplots()
    ax.imshow(room, cmap = cmap, norm = colors.BoundaryNorm(list(range(cmap.
↪N+1)), cmap.N))

    plt.text(pos[1], pos[0], u"\u2B24", fontsize = fontsize,
                  horizontalalignment = 'center', verticalalignment = 'center')

    plt.show()

 #True describes as that area/square tile is dirty and false as that area/
 ↪square tile is clean

room = [[True,False, True, False,True],
    [False, False, False,False,False],
    [False, False, True,False,False],
    [False,False,False,False,True],
    [False,False,True,False,False]]
pos = [0,0]

show_room(room, pos)

#Define Vaccum Environment as described above we have implemented a 5x5 square␣
 ↪room and probabilty of each room being clean is 0.2 so n=5 and p=0.2
def vacuum_environment (agent, n = 5, p = .2, maxsteps = 10000, verbose = True):

    room = np.random.choice(a=[True, False], size= (n,n), p=[p,1-p])

    if (verbose):
        print("room:\n", room)

    to_clean = np.sum(room)

    if (verbose): print("dirty squares:", to_clean)



    #Placing the robot in random place in the area

    pos = [np.random.randint(n),np.random.randint(n)]



    #numbers of squares to be cleaned

    num_clean = 0

    # amount of cost as Performance Measure

    cost = 0

    i = 1
```

```
if (verbose): print("\nstart simulation")

while cost < maxsteps and num_clean < to_clean :

    if verbose:
        show_room(room, pos)
    # Define percepts and their position
    bumpers = {
                "north": pos[0] == 0,
                "south": pos[0] == n-1,
                "west" : pos[1] == 0,
                "east" : pos[1] == n-1
                 }

    dirty = room[pos[0]][pos[1]]
    if (verbose): print("-----")
    if (verbose):
        print("step:", i)
        i = i + 1
    if (verbose): print("current position", pos)
    if (verbose): print("bumper", bumpers)
    if (verbose): print("dirty", dirty)
    # Call agent program function with percepts
    action = agent(bumpers,dirty)

    if (verbose):
        print("action", action)


    #Calculating the cost for each action it performs
    cost += 1

    if action == "north" :
        pos[0] = pos[0] - 1
        #check for any obstacles or bumper
        if pos[0] == -1:
            pos[0] = 0

    elif action == "south" :
        pos[0] = pos[0] + 1
        #check for any obstacles or bumper
        if pos[0] == n:
            pos[0] = n - 1

    elif action == "west" :
        pos[1] = pos[1] - 1
        #check for any obstacles or bumper
        if pos[1] == -1:
            pos[1] = 0

    elif action == "east" :
        pos[1] = pos[1] + 1
        #check for any obstacles or bumper
        if pos[1] == n:
            pos[1] = n - 1

    elif action == "suck" :
        if dirty:
            room[pos[0]][pos[1]] = False
            # Calculating the number of dirty tiles it has cleaned
            num_clean+= 1

return [num_clean, cost]
```

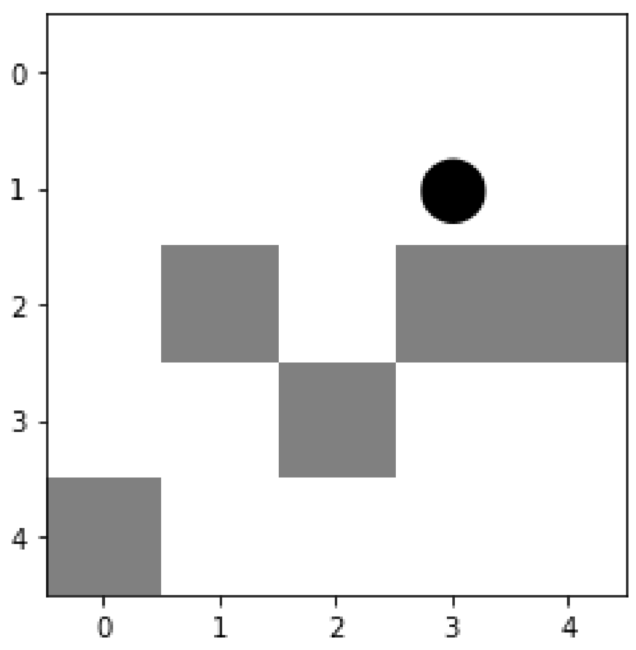


```
[6]: vacuum_environment(simple_randomized_agent, n = 5, maxsteps = 10000, verbose =␣
     ↪True)
```

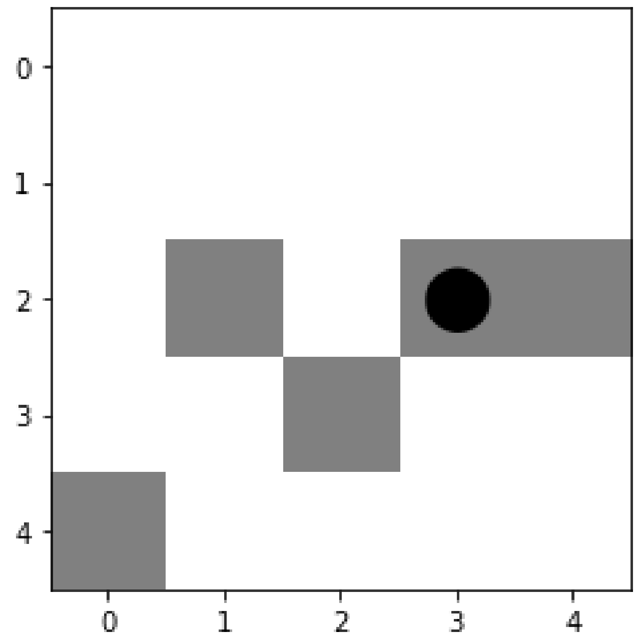
```
room:
 [[False False False False False]
 [False False False False False]
 [False  True False  True  True]
 [False False  True False False]
 [ True False False False False]]
dirty squares: 5

start simulation
```



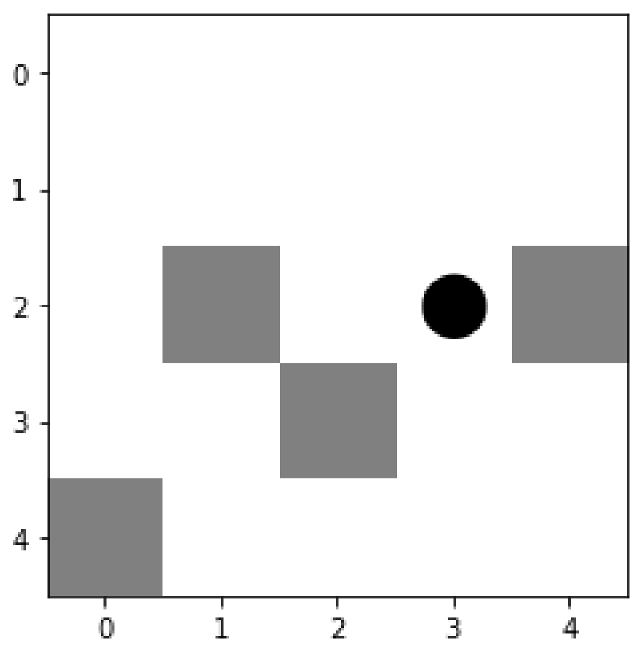
```
-----
step: 1
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



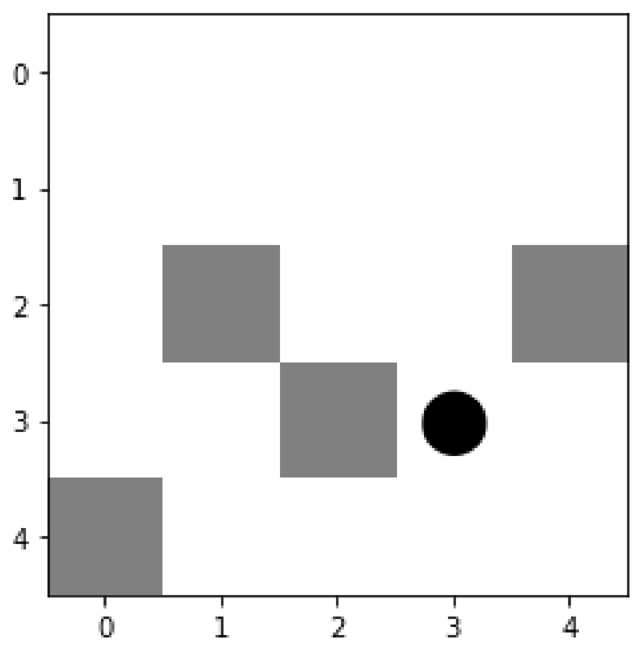
```
-----
step: 2
current position [1, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



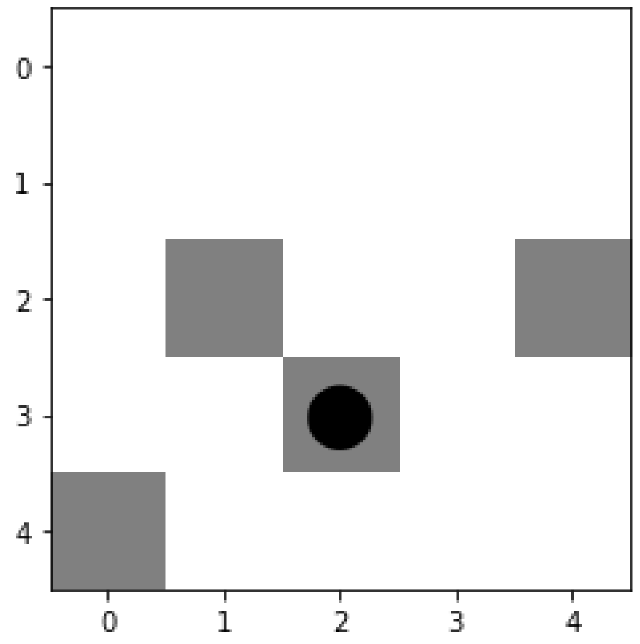
```
-----
step: 3
current position [2, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty True
action suck
```



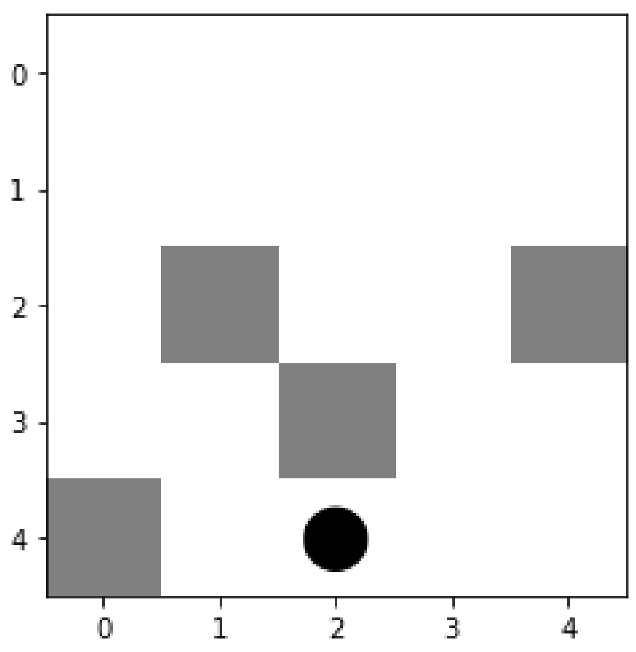
```
-----
step: 4
current position [2, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



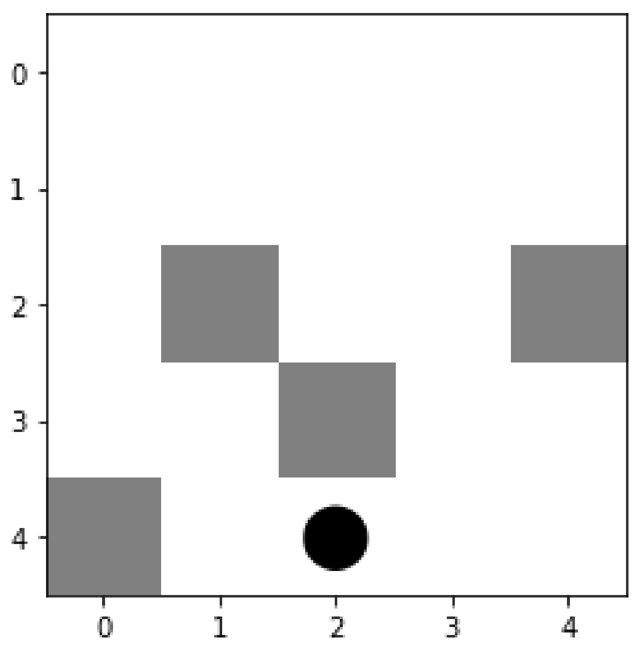
```
-----
step: 5
current position [3, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



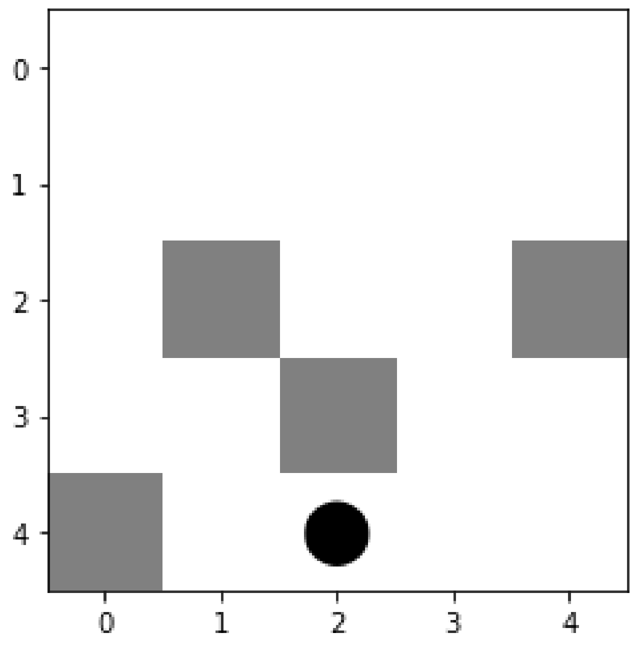
```
-----
step: 6
current position [3, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty True
action south
```



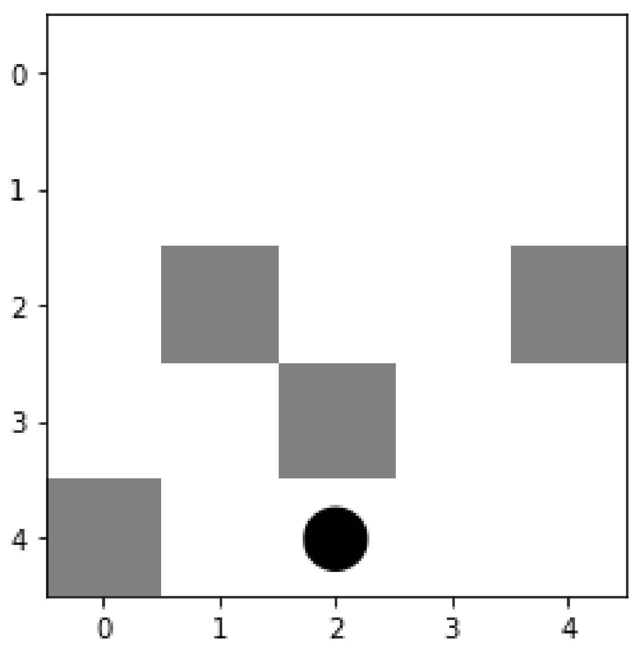
```
-----
step: 7
current position [4, 2]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action suck
```



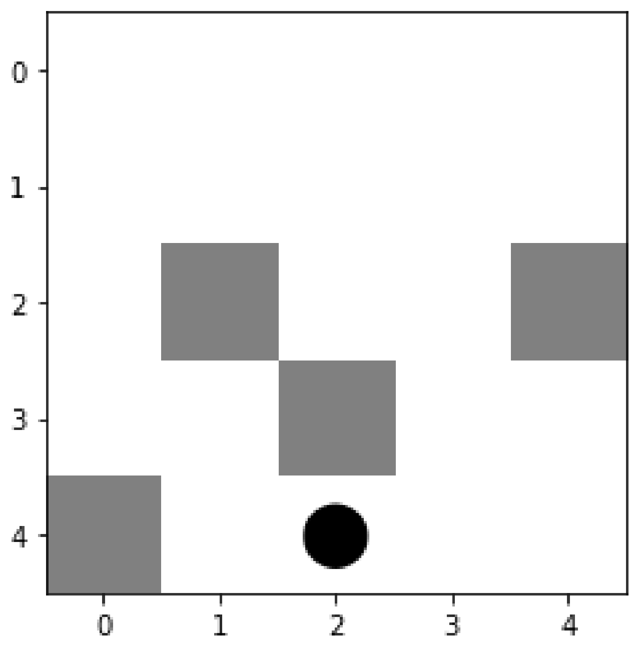
```
-----
step: 8
current position [4, 2]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action suck
```



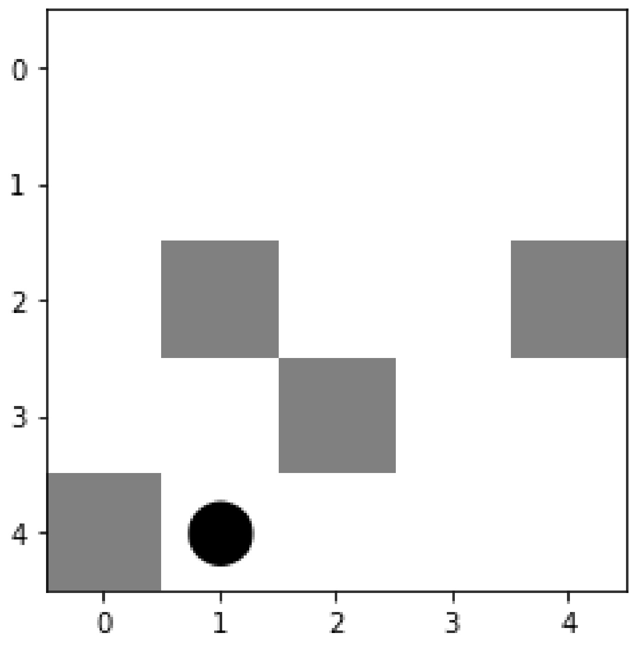
```
-----
step: 9
current position [4, 2]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action south
```



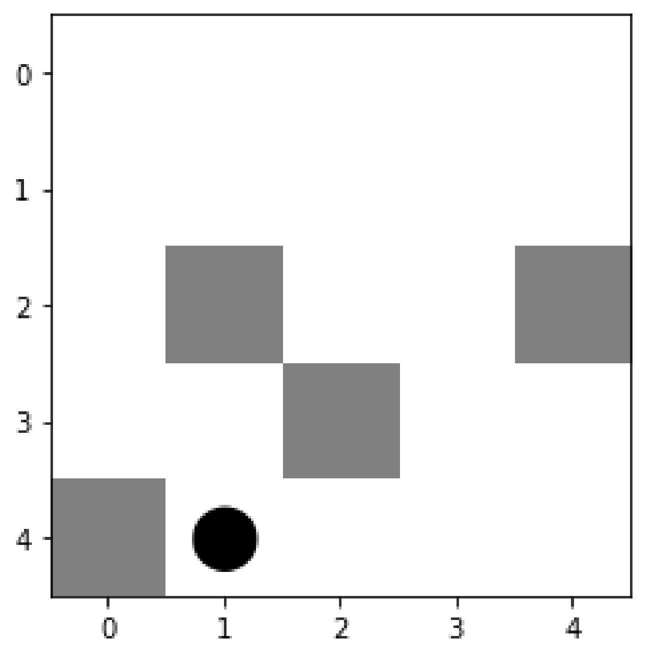
```
-----
step: 10
current position [4, 2]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action south
```



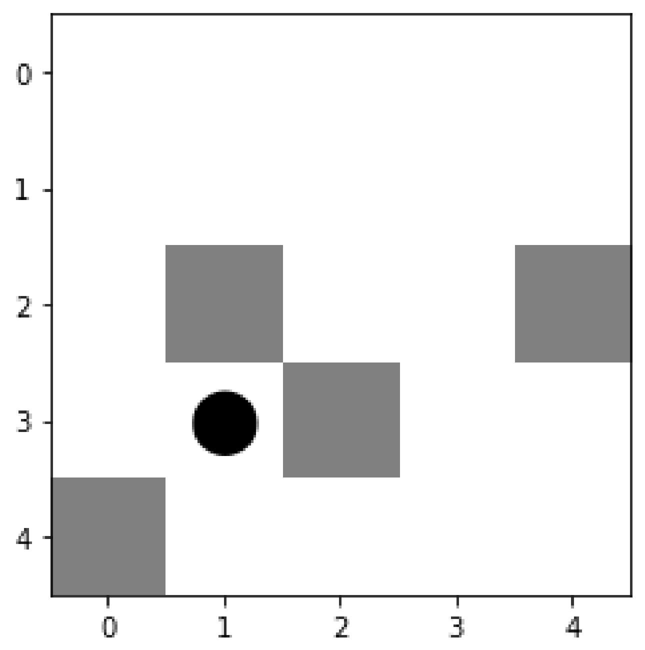
```
-----
step: 11
current position [4, 2]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action west
```



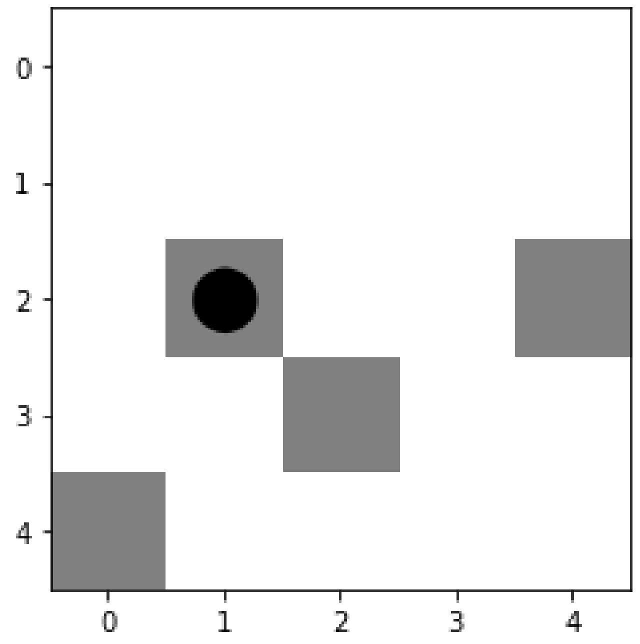
```
-----
step: 12
current position [4, 1]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action south
```



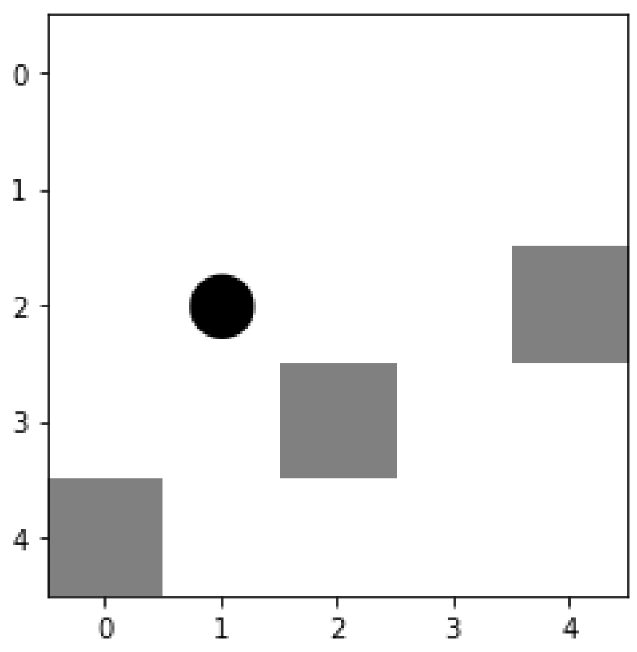
```
-----
step: 13
current position [4, 1]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action north
```



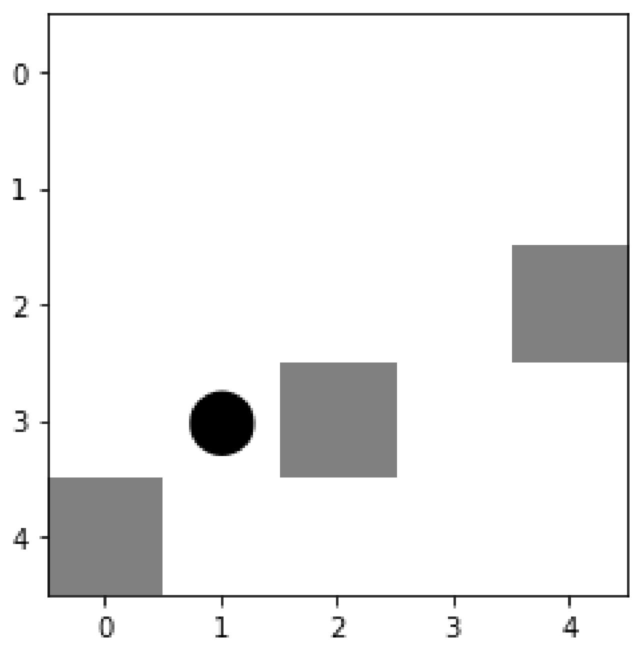
```
-----
step: 14
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



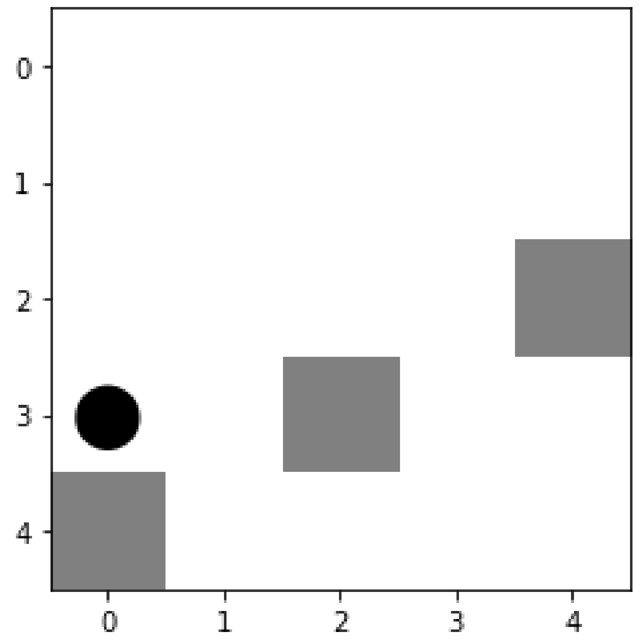
```
-----
step: 15
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty True
action suck
```



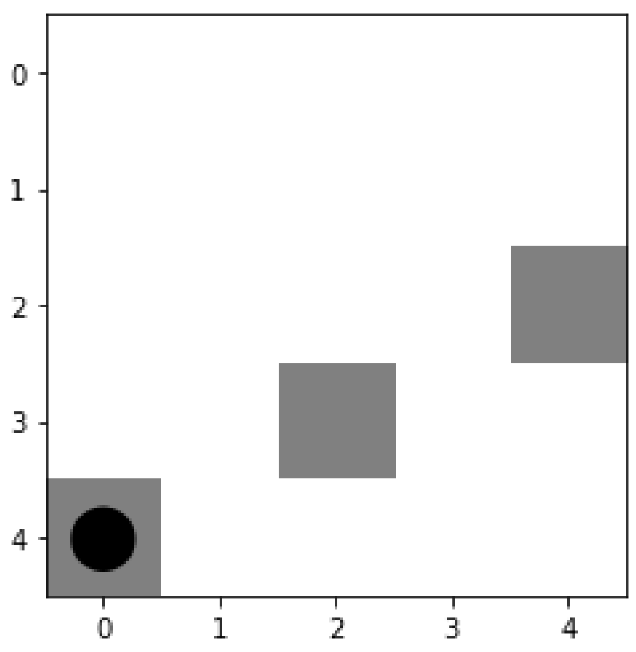
```
-----
step: 16
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



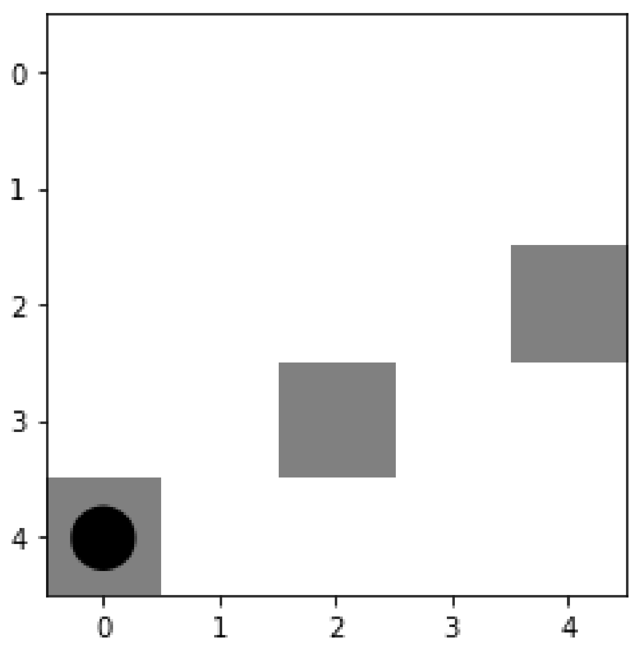
```
-----
step: 17
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



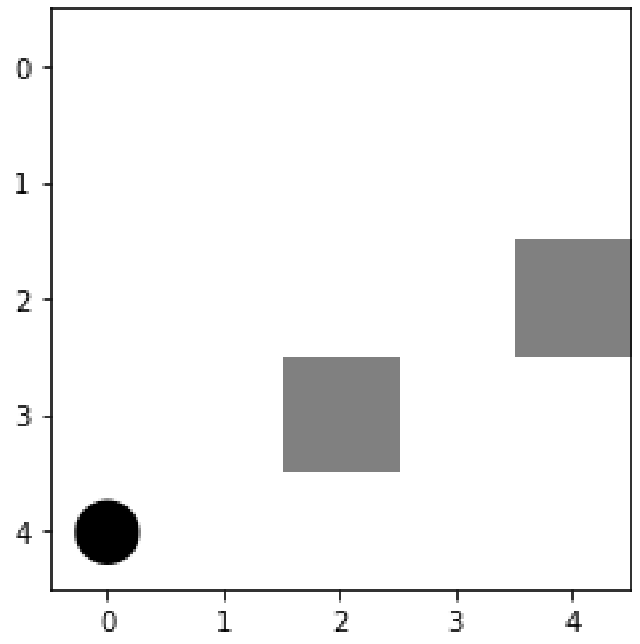
```
-----
step: 18
current position [3, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action south
```



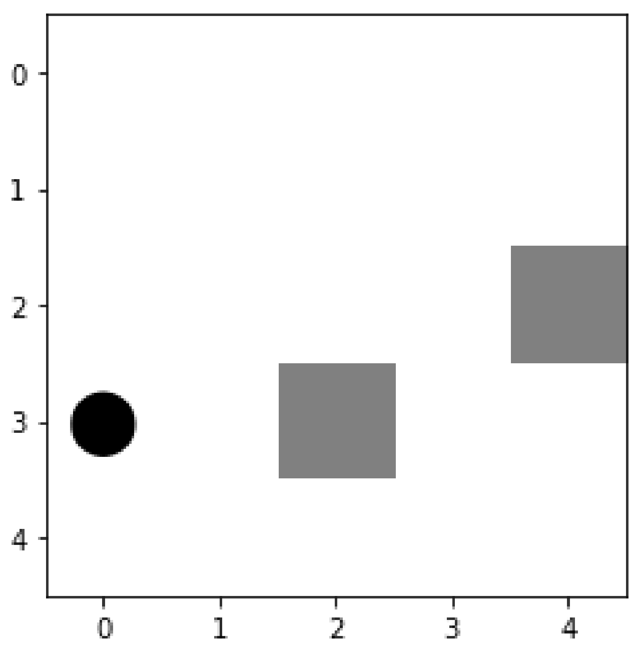
```
-----
step: 19
current position [4, 0]
bumper {'north': False, 'south': True, 'west': True, 'east': False}
dirty True
action south
```



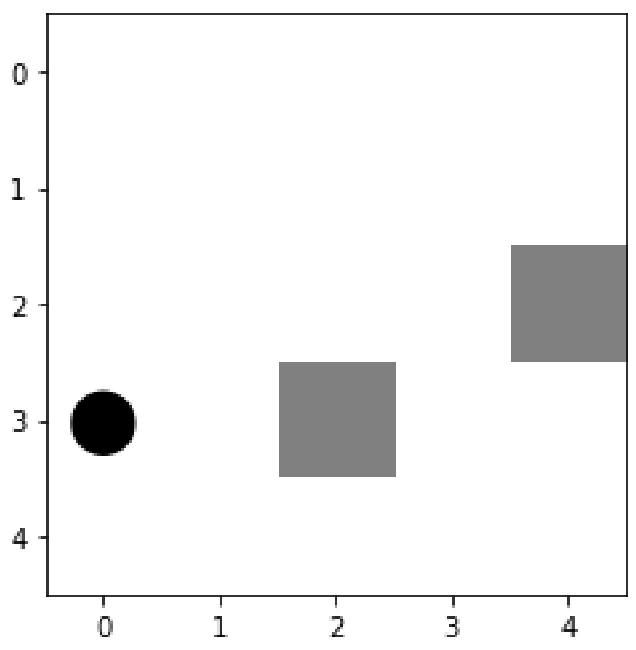
```
-----
step: 20
current position [4, 0]
bumper {'north': False, 'south': True, 'west': True, 'east': False}
dirty True
action suck
```



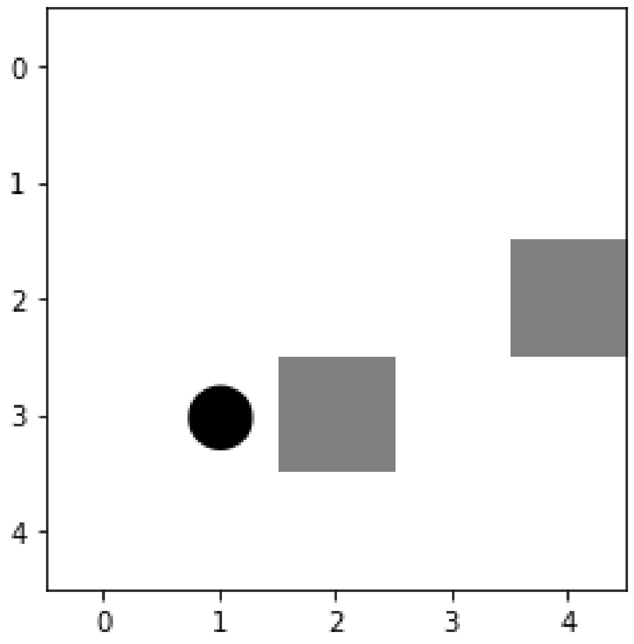
```
-----
step: 21
current position [4, 0]
bumper {'north': False, 'south': True, 'west': True, 'east': False}
dirty False
action north
```



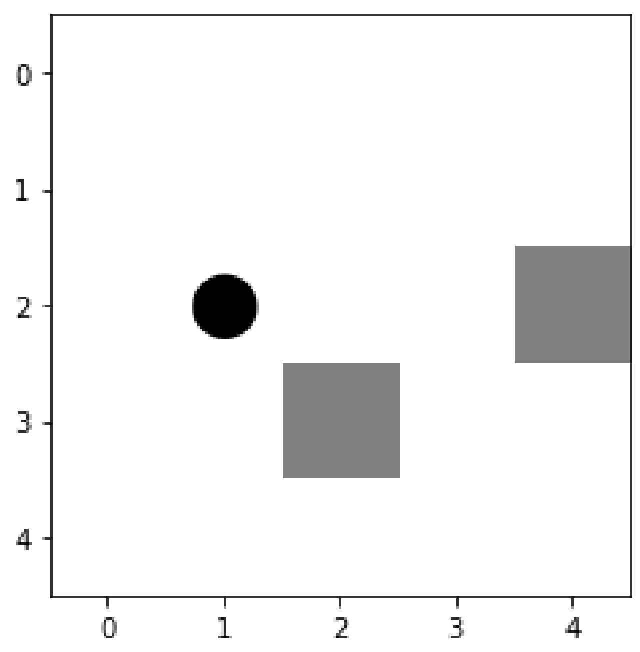
```
-----
step: 22
current position [3, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action west
```



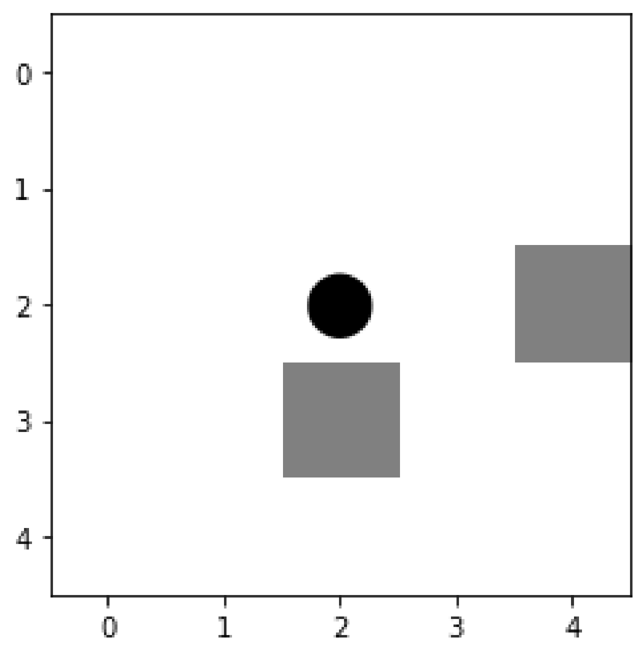
```
-----
step: 23
current position [3, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



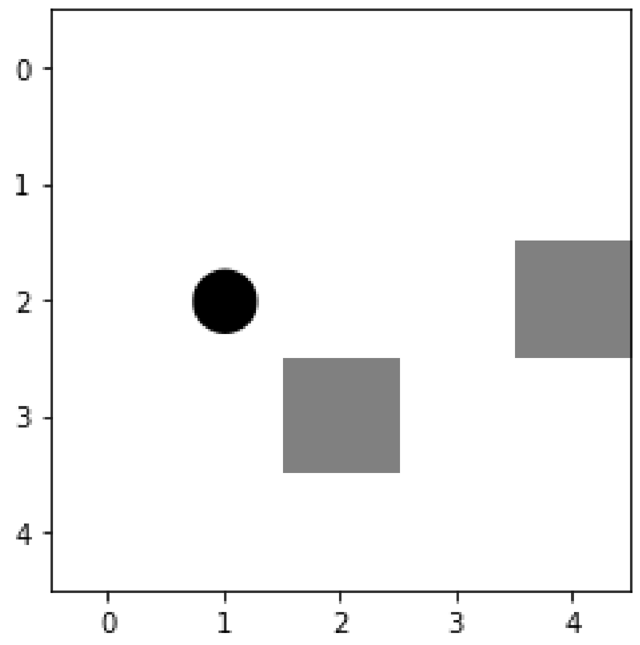
```
-----
step: 24
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



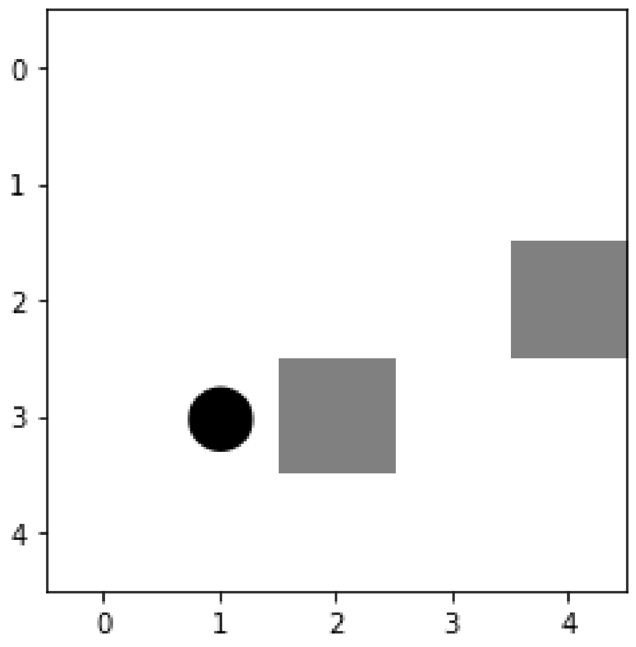
```
-----
step: 25
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



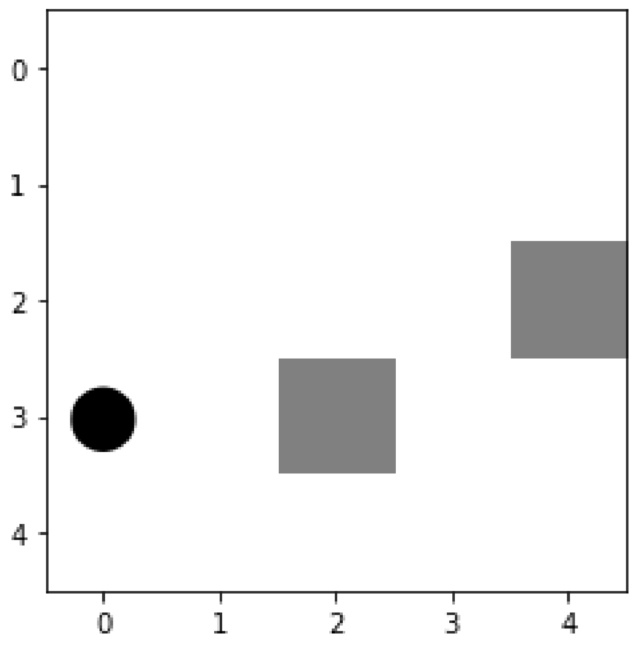
```
-----
step: 26
current position [2, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



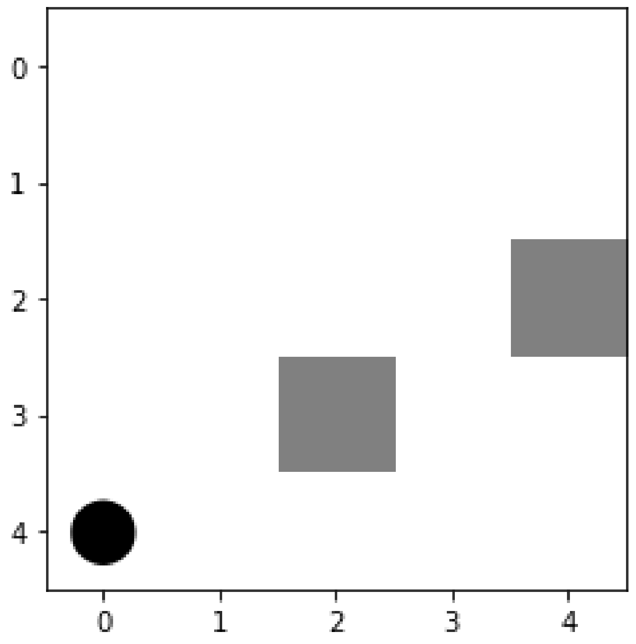
```
-----
step: 27
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



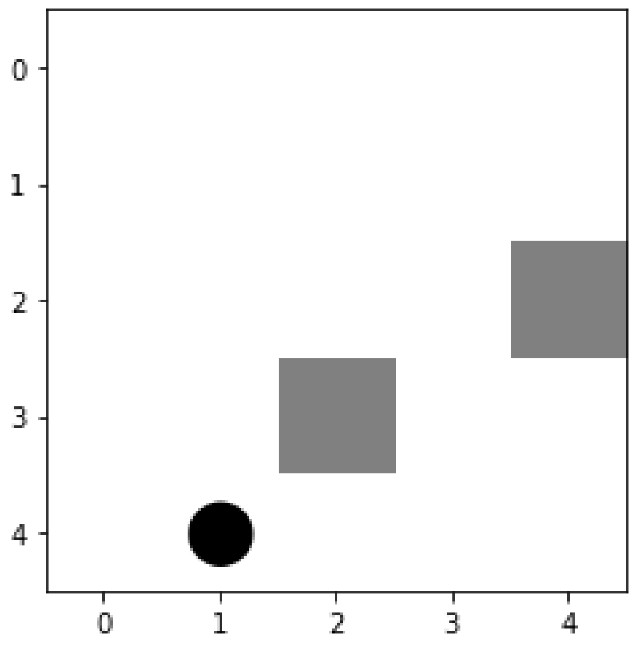
```
-----
step: 28
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



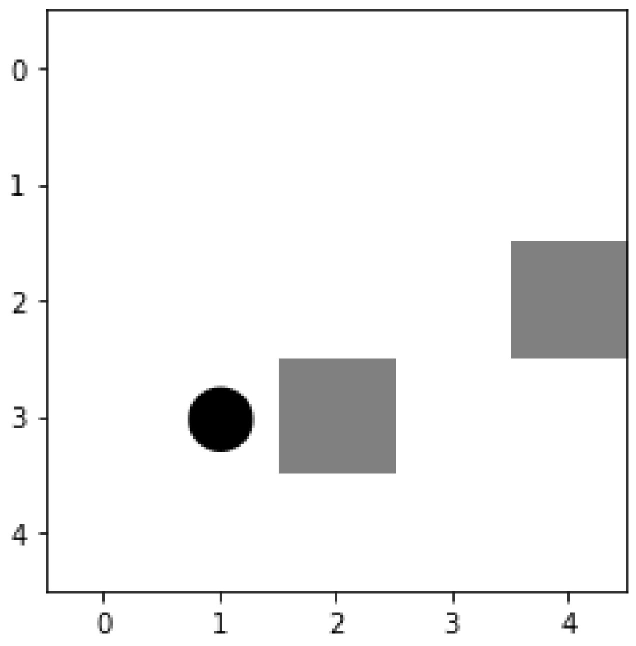
```
-----
step: 29
current position [3, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action south
```



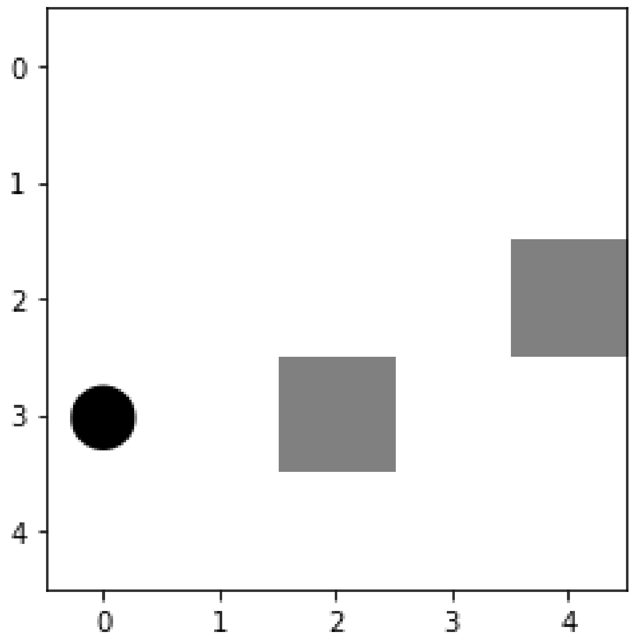
```
-----
step: 30
current position [4, 0]
bumper {'north': False, 'south': True, 'west': True, 'east': False}
dirty False
action east
```



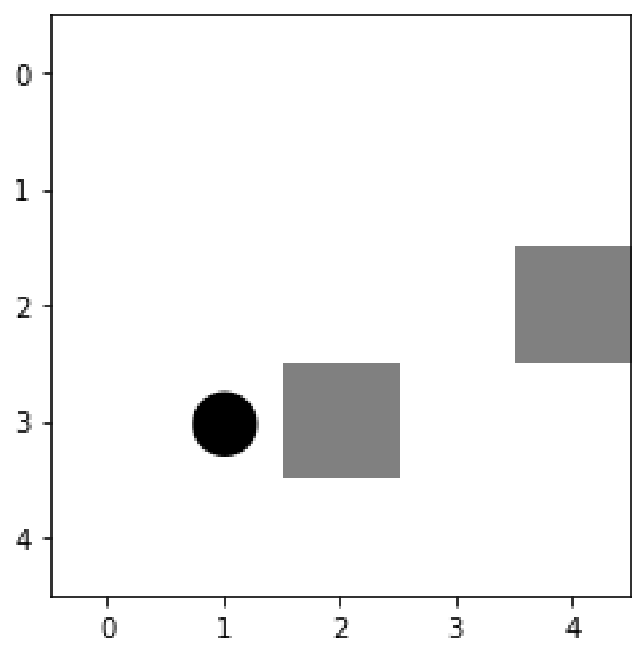
```
-----
step: 31
current position [4, 1]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action north
```



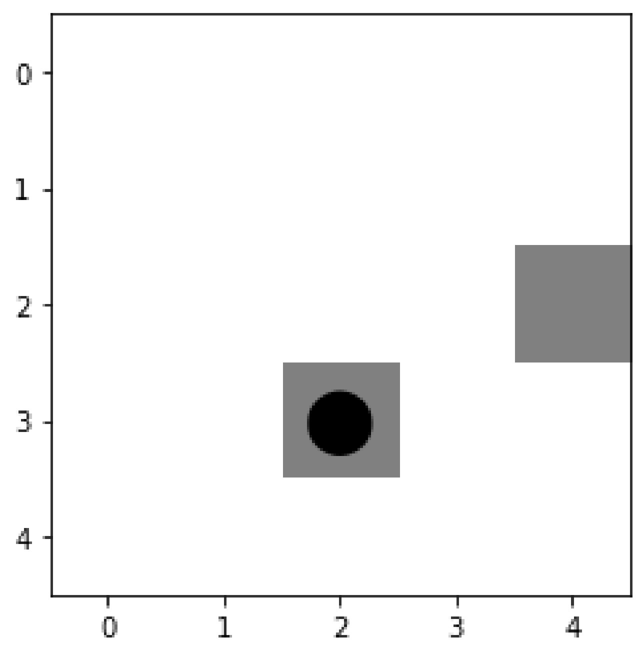
```
-----
step: 32
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



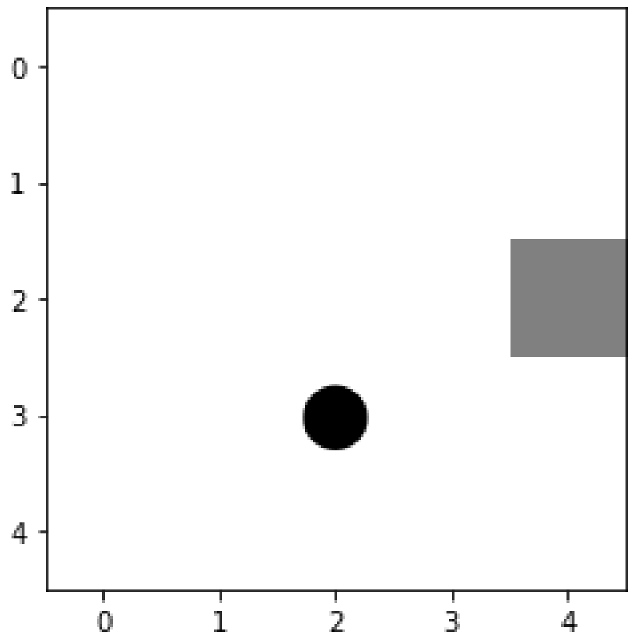
```
-----
step: 33
current position [3, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



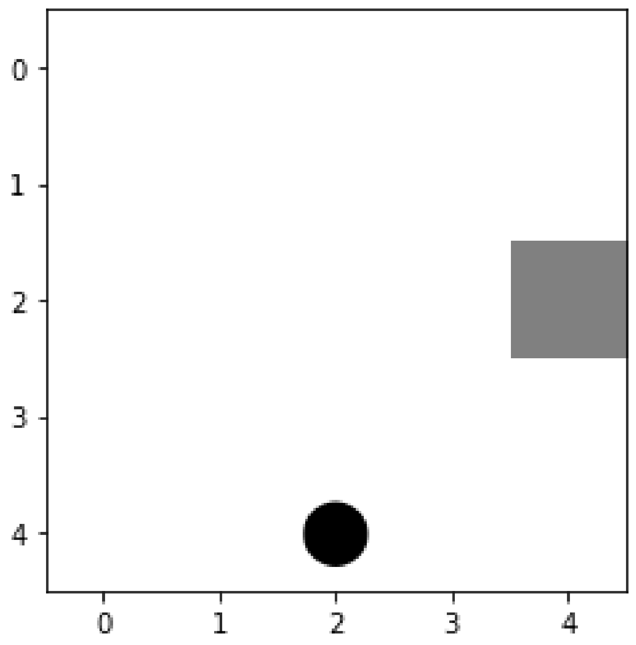
```
-----
step: 34
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



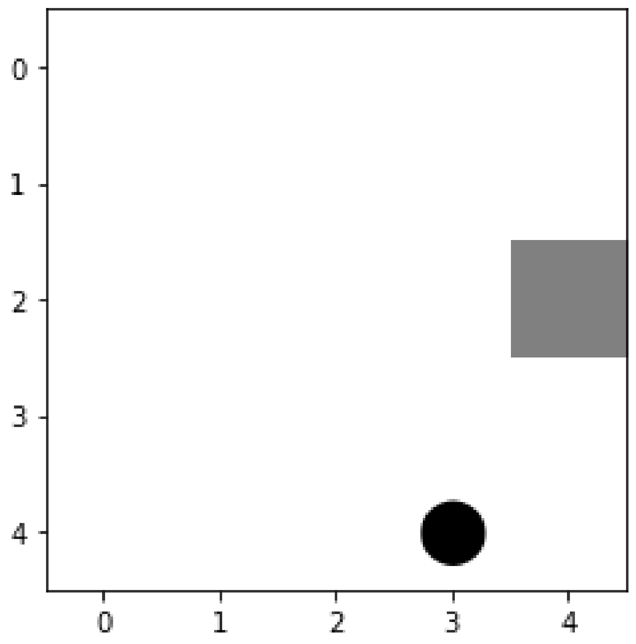
```
-----
step: 35
current position [3, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty True
action suck
```



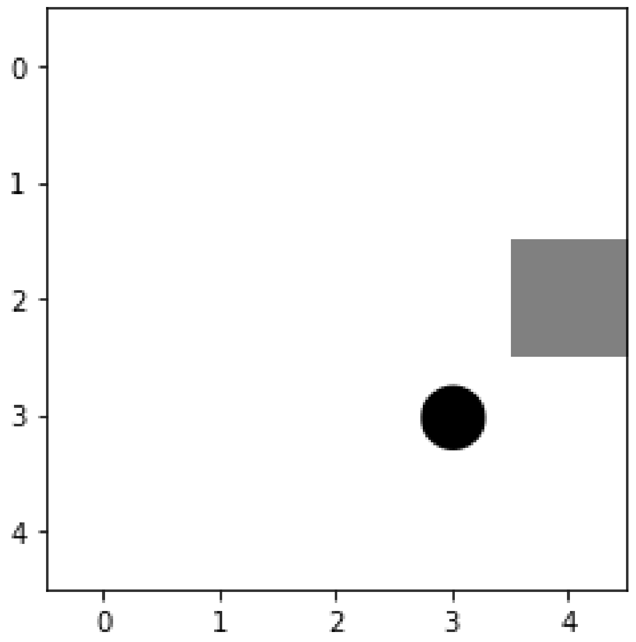
```
-----
step: 36
current position [3, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



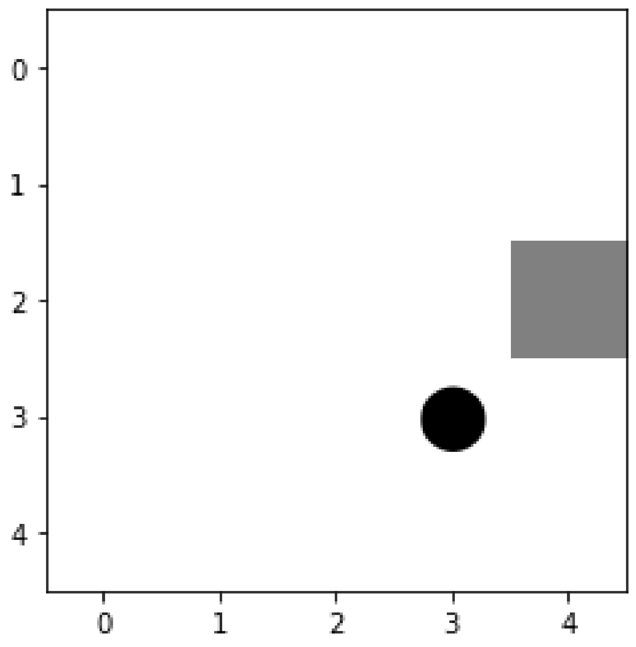
```
-----
step: 37
current position [4, 2]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action east
```



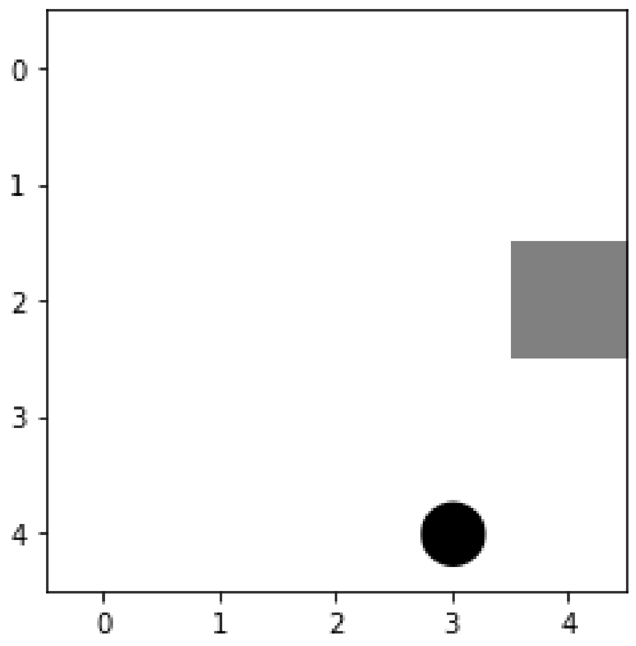
```
-----
step: 38
current position [4, 3]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action north
```



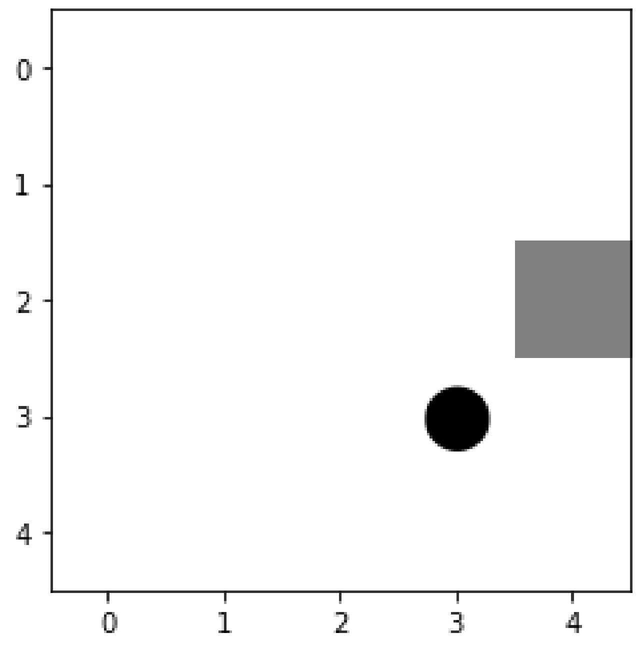
```
-----
step: 39
current position [3, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action suck
```



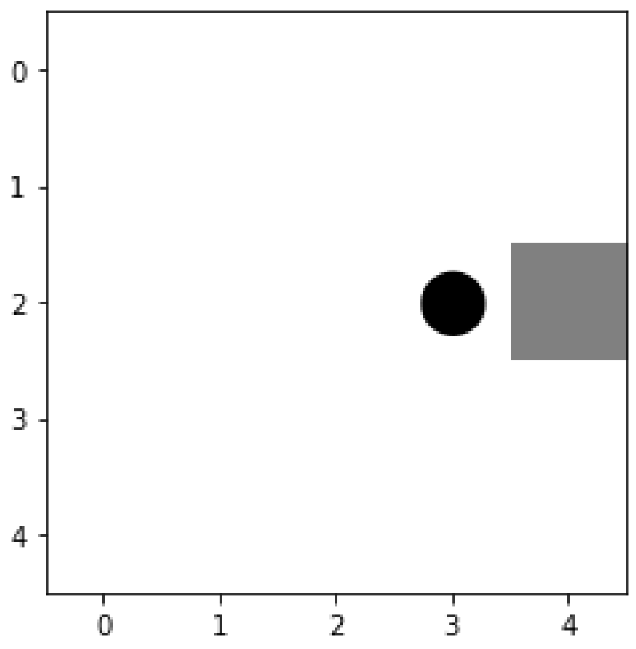
```
-----
step: 40
current position [3, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



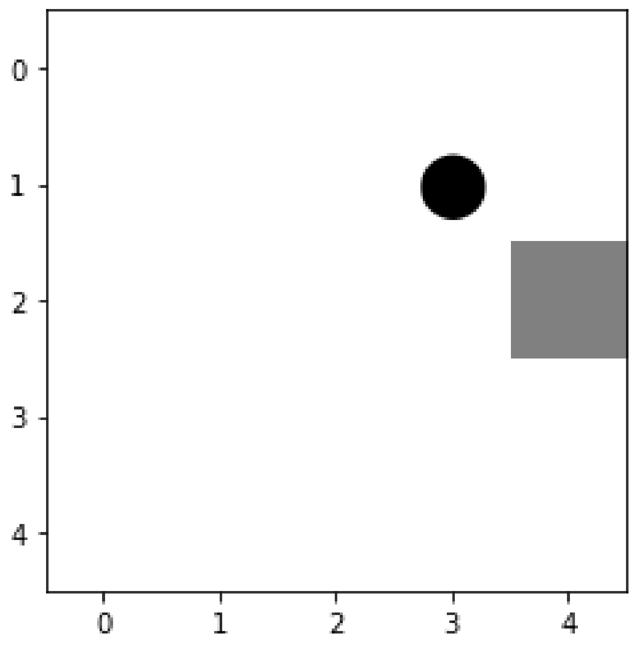
```
-----
step: 41
current position [4, 3]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action north
```



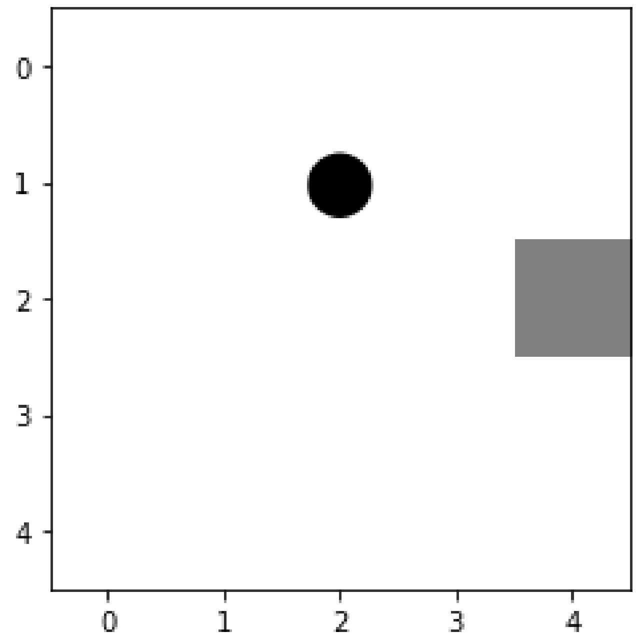
```
-----
step: 42
current position [3, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



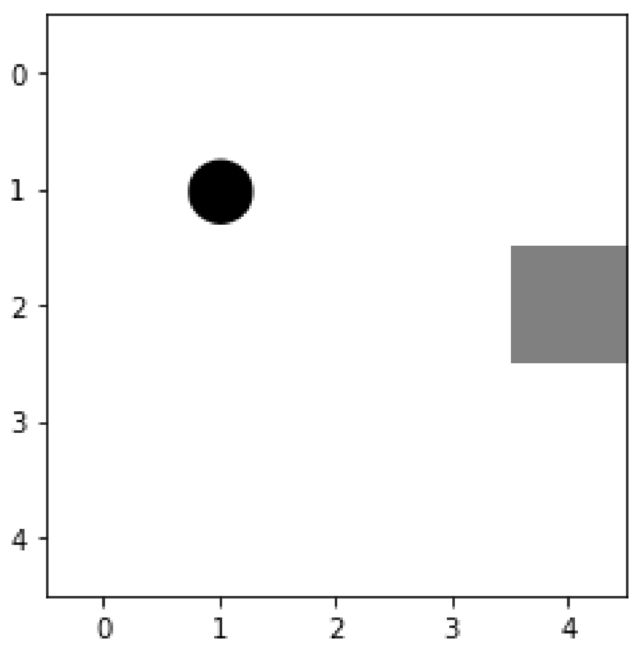
```
-----
step: 43
current position [2, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



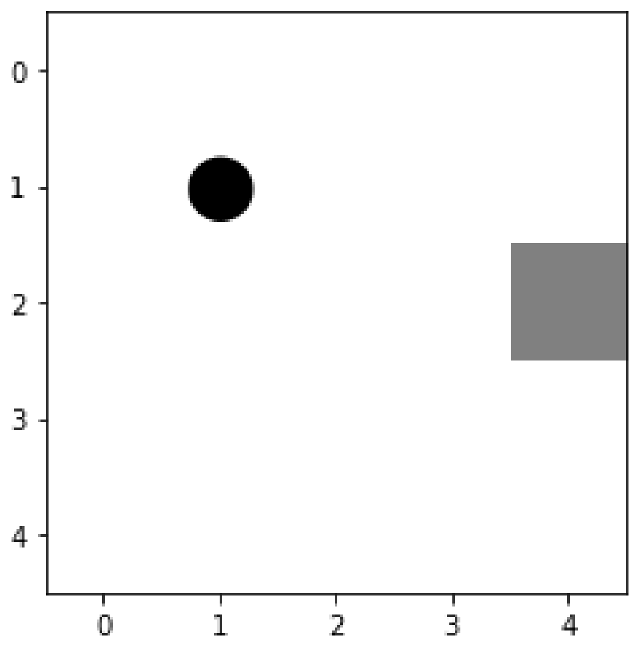
```
-----
step: 44
current position [1, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



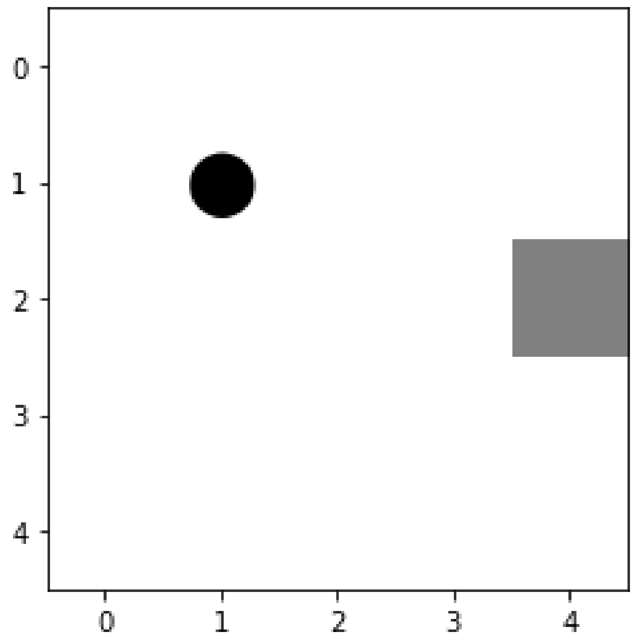
```
-----
step: 45
current position [1, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



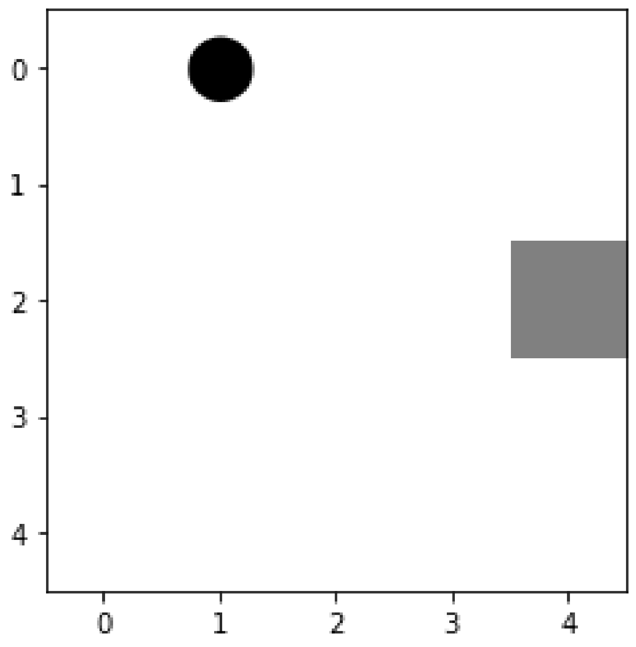
```
-----
step: 46
current position [1, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action suck
```



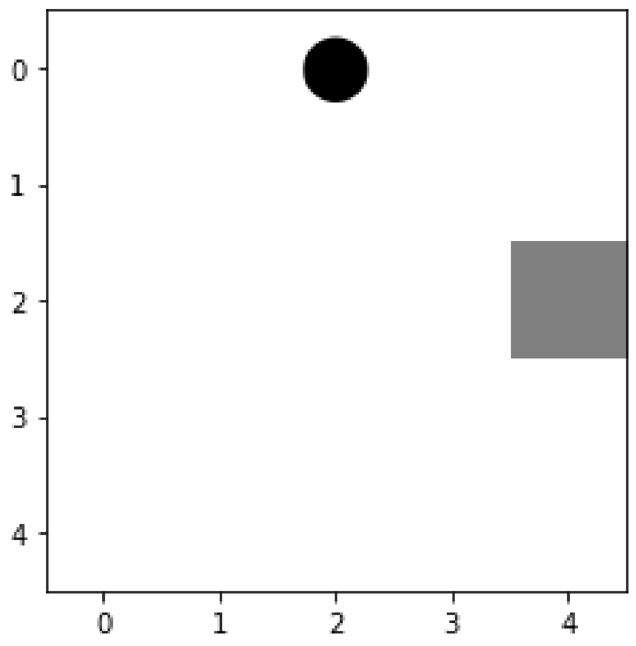
```
-----
step: 47
current position [1, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action suck
```



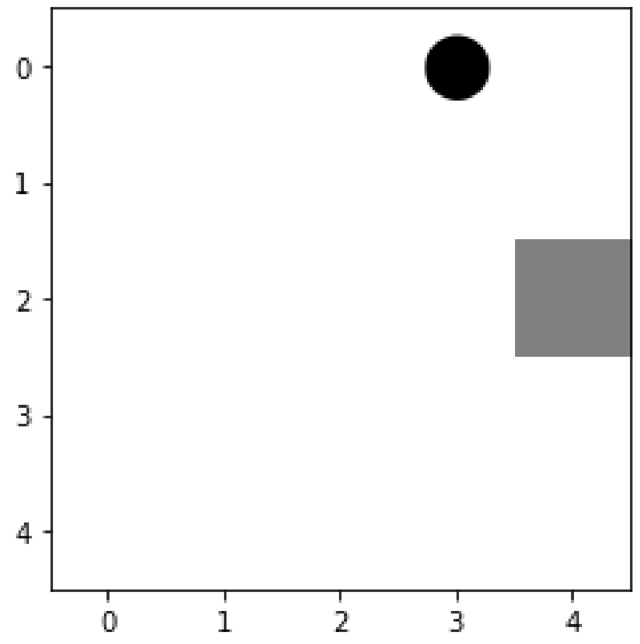
```
-----
step: 48
current position [1, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



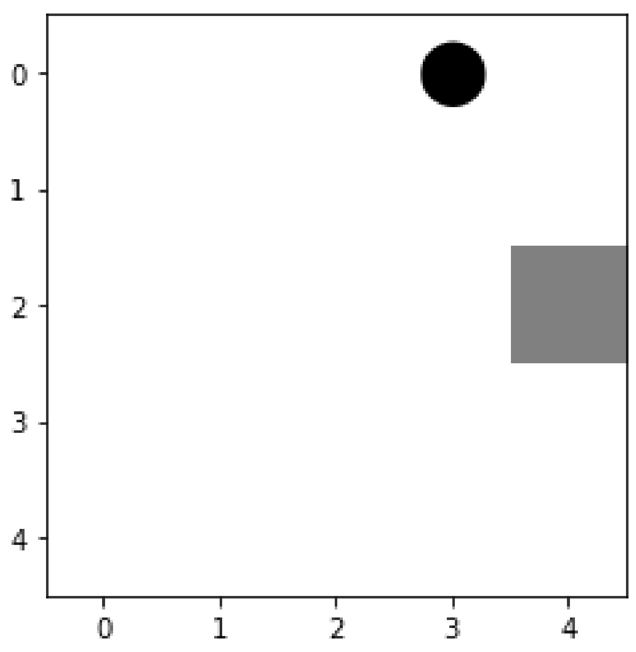
```
-----
step: 49
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



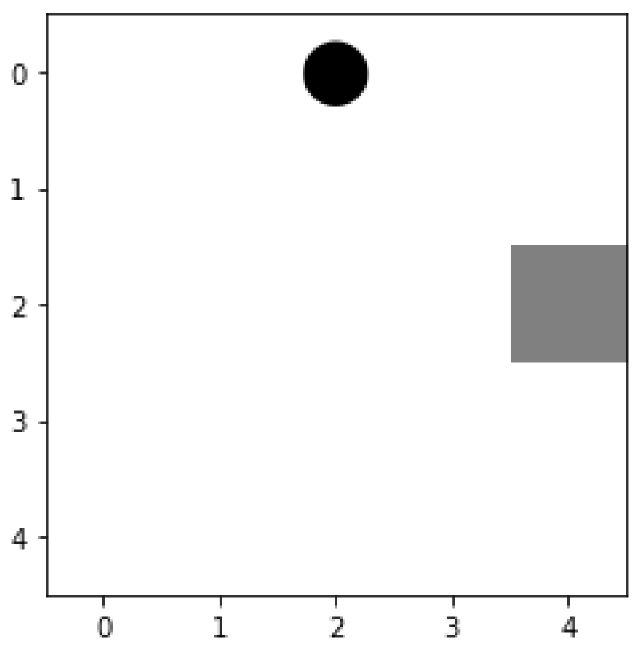
```
-----
step: 50
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



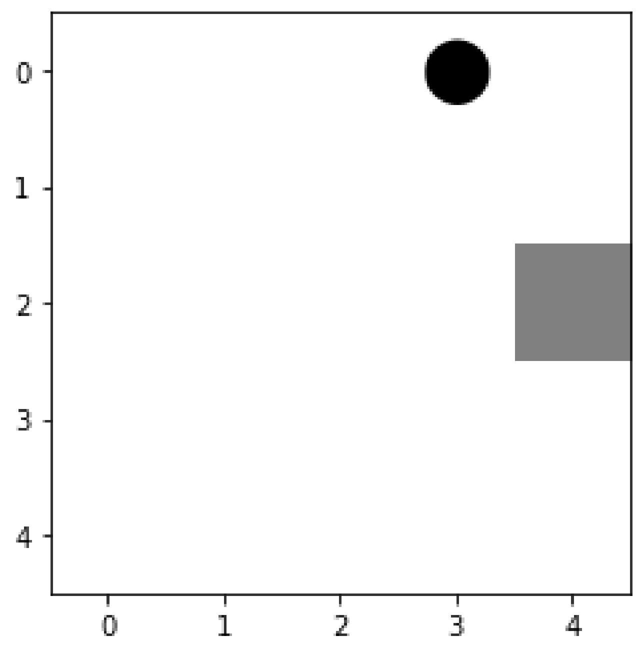
```
-----
step: 51
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action suck
```



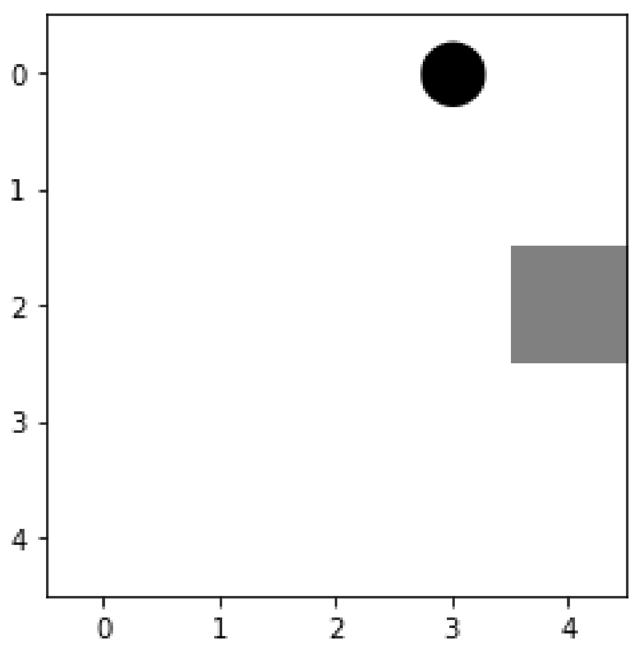
```
-----
step: 52
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



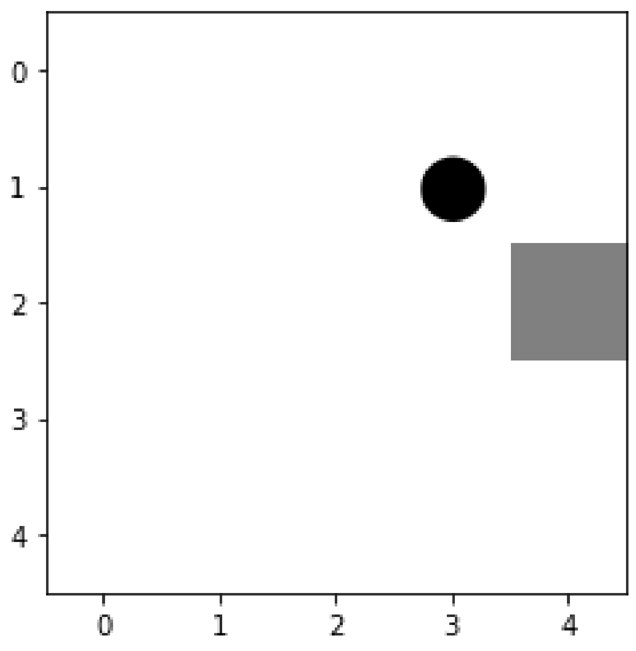
```
-----
step: 53
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



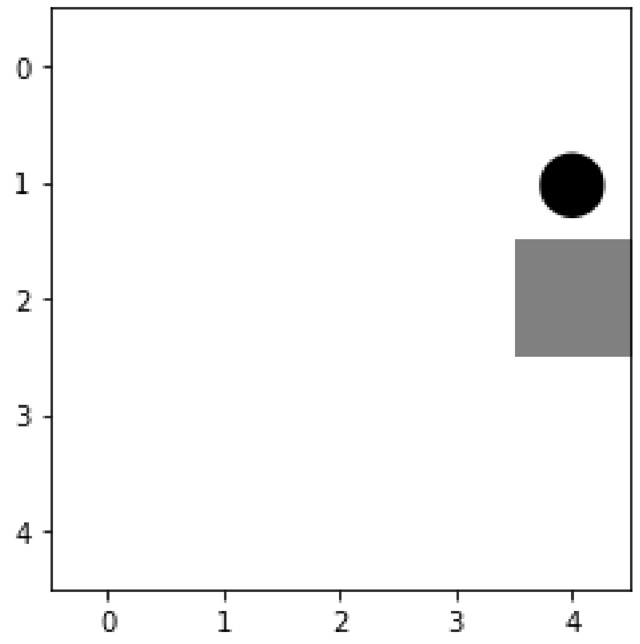
```
-----
step: 54
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



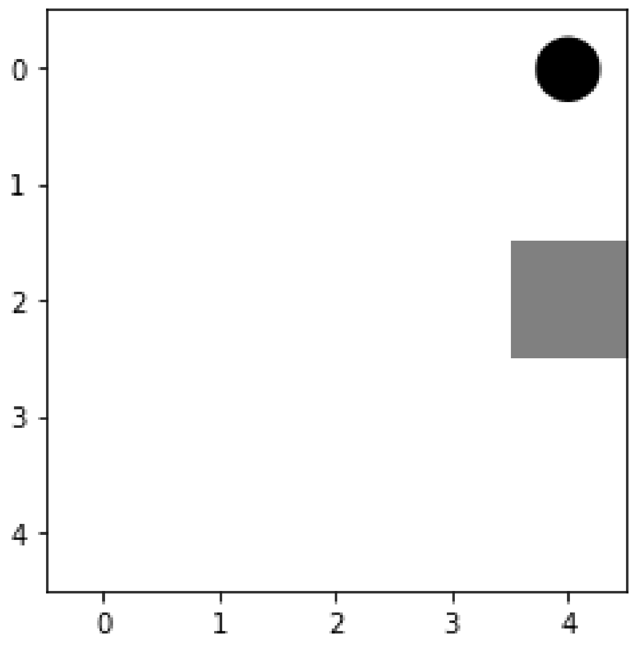
```
-----
step: 55
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



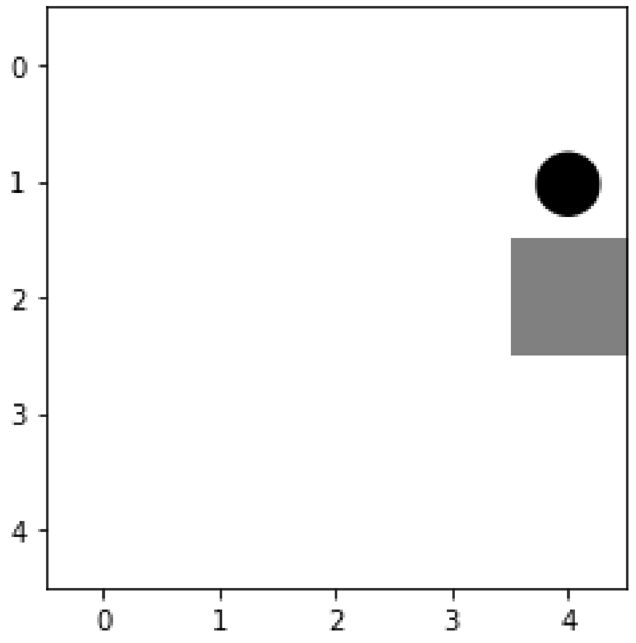
```
-----
step: 56
current position [1, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



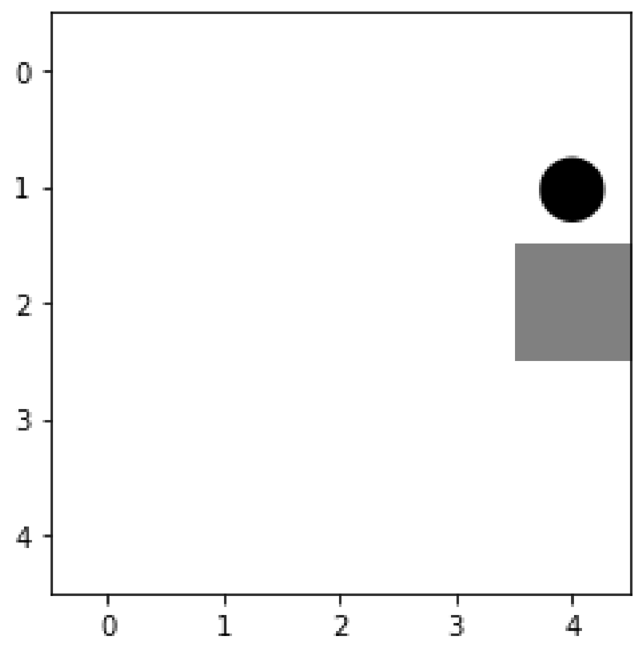
```
-----
step: 57
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action north
```



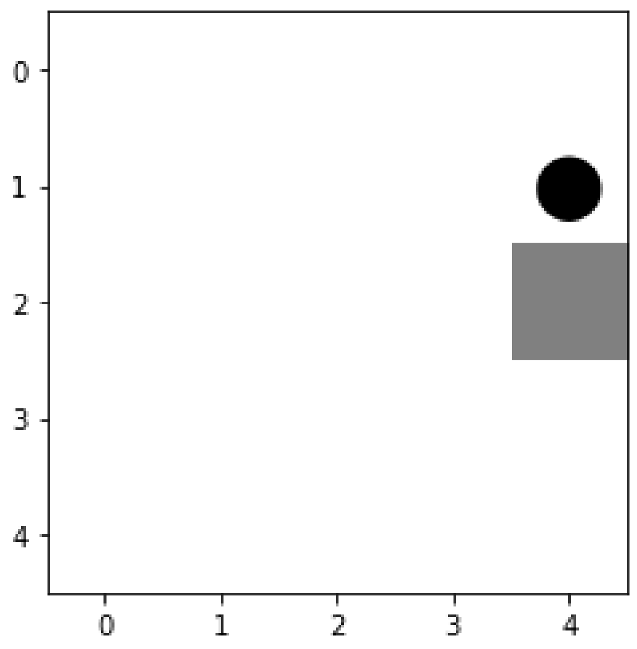
```
-----
step: 58
current position [0, 4]
bumper {'north': True, 'south': False, 'west': False, 'east': True}
dirty False
action south
```



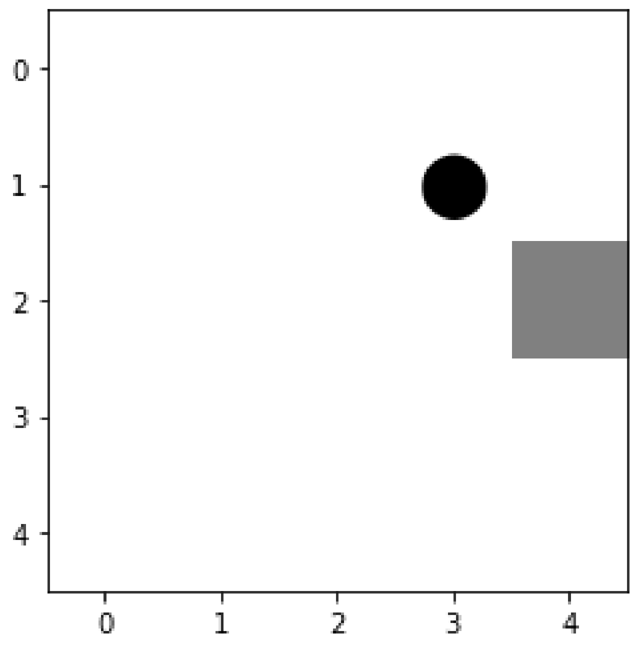
```
-----
step: 59
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action suck
```



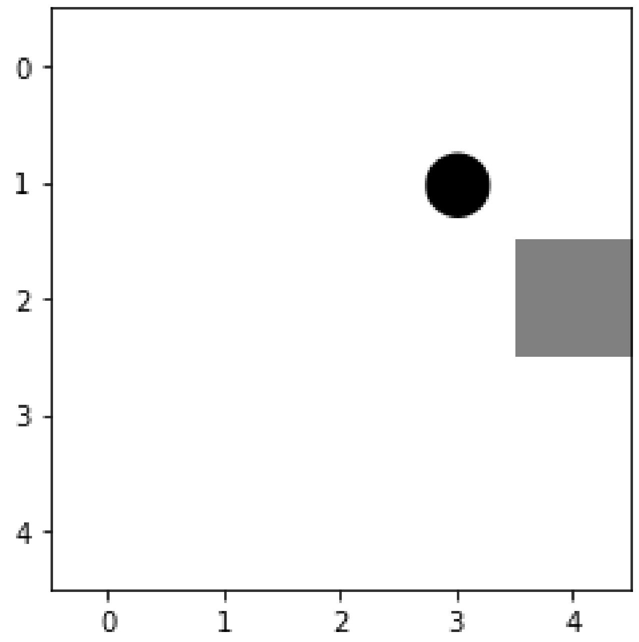
```
-----
step: 60
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action east
```



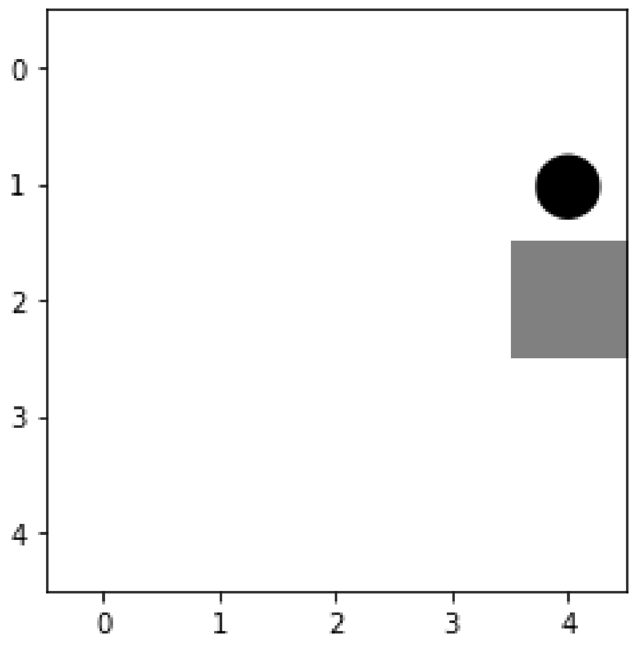
```
-----
step: 61
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



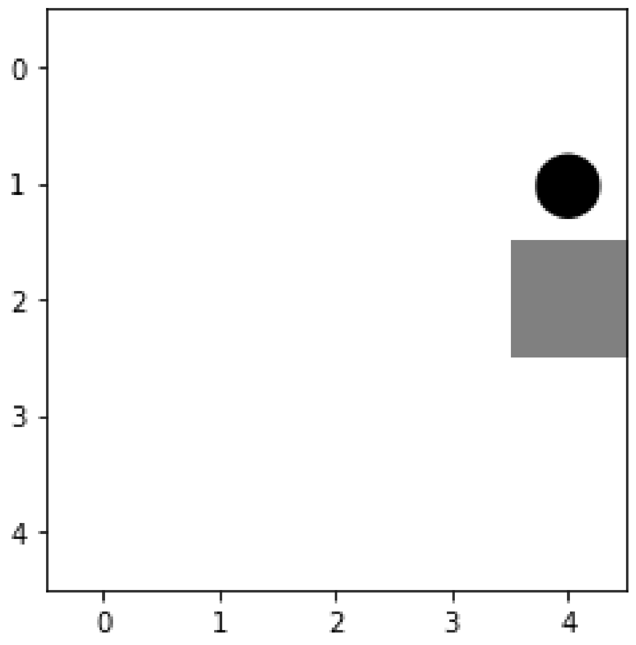
```
-----
step: 62
current position [1, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action suck
```



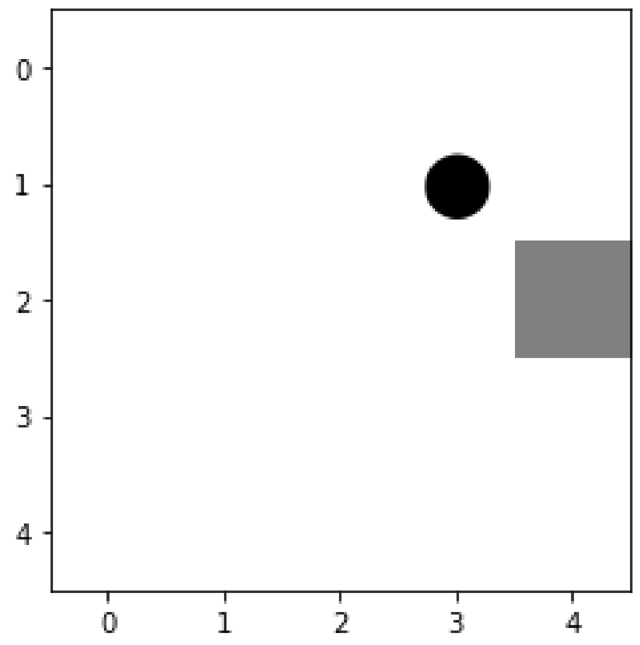
```
-----
step: 63
current position [1, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



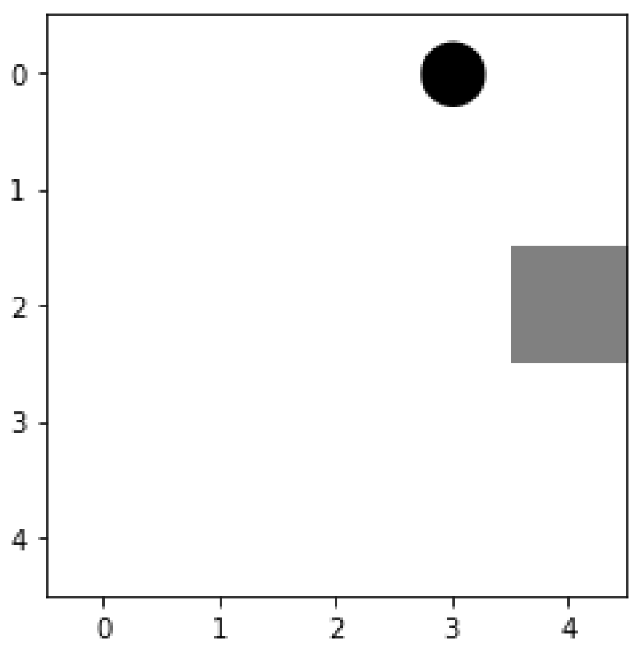
```
-----
step: 64
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action east
```



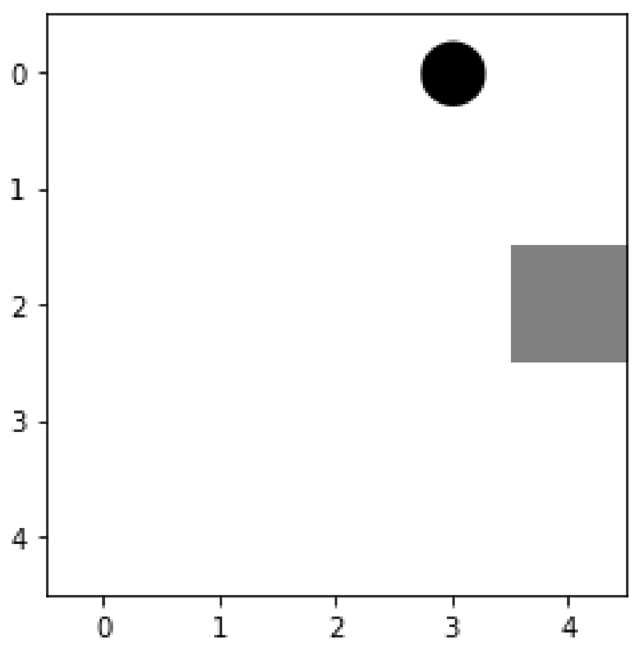
```
-----
step: 65
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



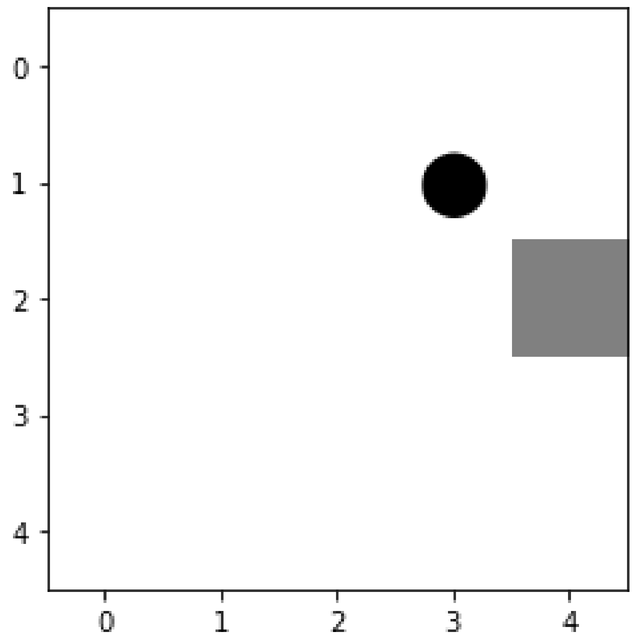
```
-----
step: 66
current position [1, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



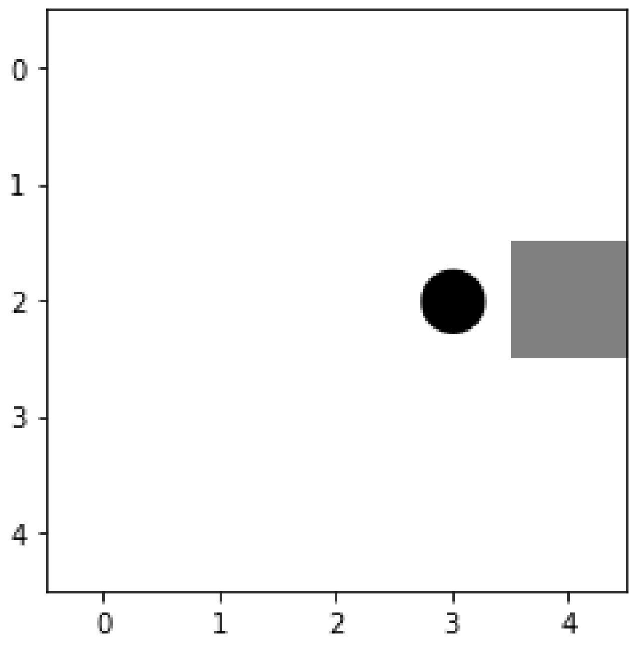
```
-----
step: 67
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



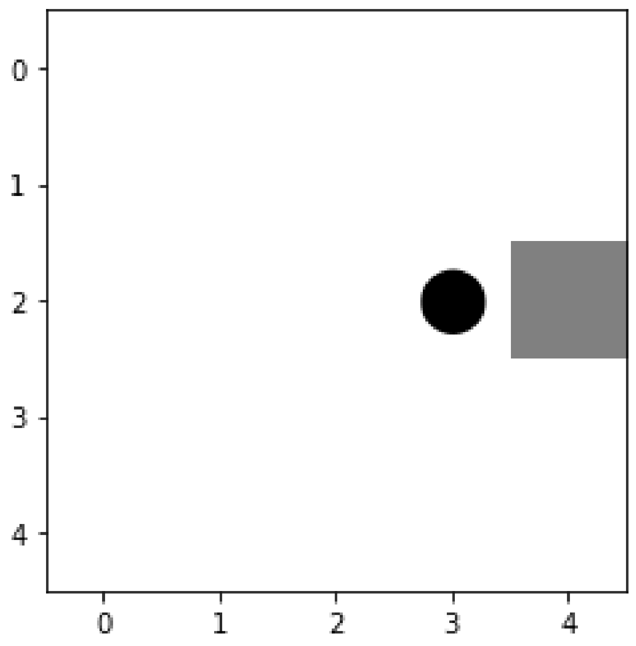
```
-----
step: 68
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



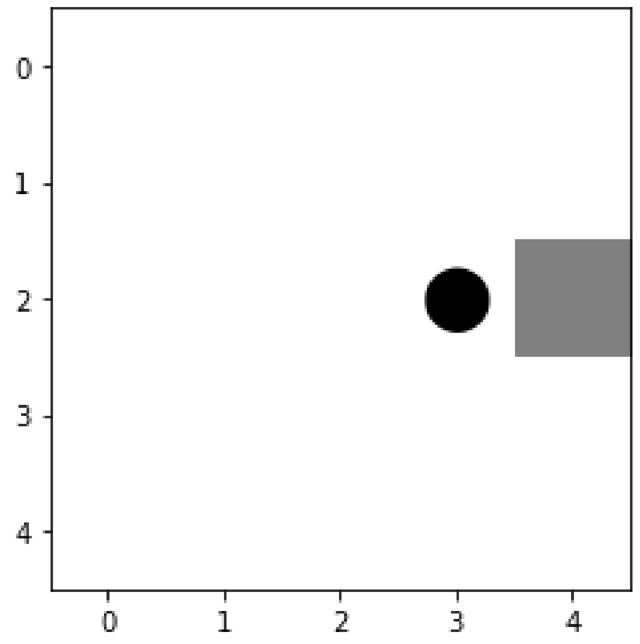
```
-----
step: 69
current position [1, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



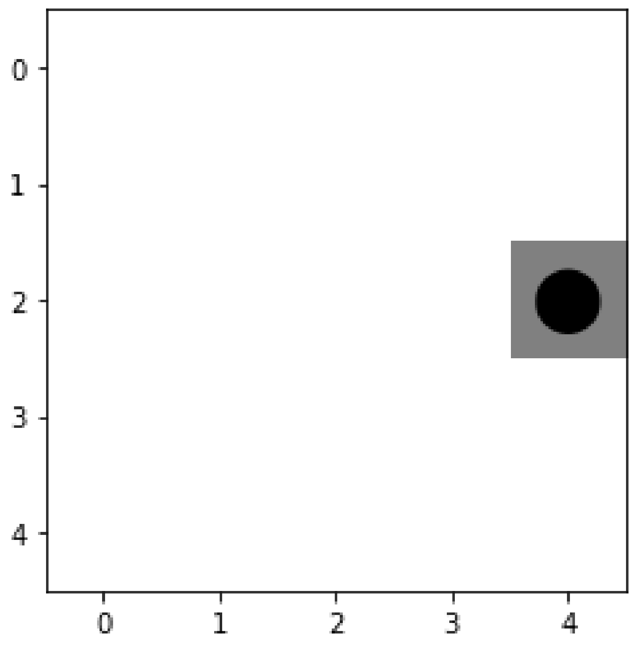
```
-----
step: 70
current position [2, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action suck
```



```
-----
step: 71
current position [2, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action suck
```



```
-----
step: 72
current position [2, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



```
-----
step: 73
current position [2, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty True
action suck
```

[6]:
```
[5, 73]
```

3 Task 2: Implement a simple reflex agent [1 Point]

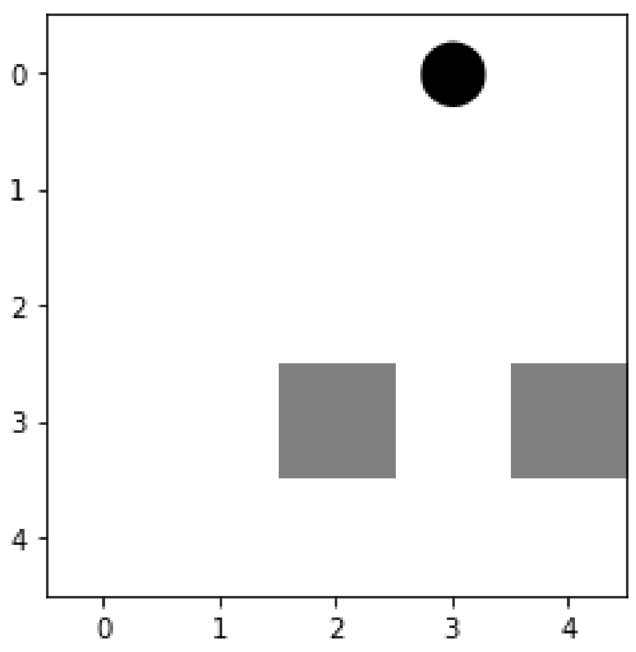
The simple reflex agent randomly walks around but reacts to the bumper sensor by not bumping into the wall and to dirt with sucking. Implement the agent program as a function.

Note: Agents cannot directly use variable in the environment. They only gets the percepts as the arguments to the agent function.

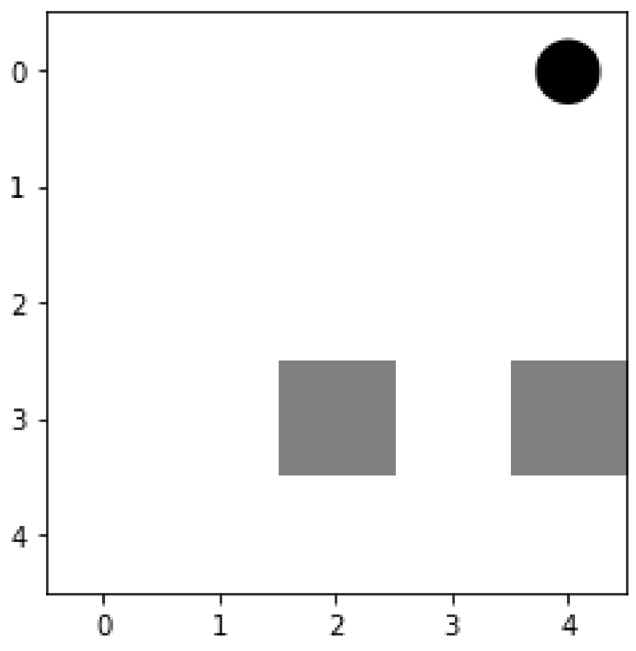
[7]:
```
# Your code and description goes here

def simple_reflex_agent(bumpers,dirty):

    directions = ["north", "east", "west", "south"]



    if dirty == True:
        return "suck";


    else :

        if bumpers["north"]:

            directions.remove("north")


        if bumpers["east"]:

            directions.remove("east")


        if bumpers["west"]:

            directions.remove("west")


        if bumpers["south"]:

            directions.remove("south")


        return np.random.choice(directions)
```

[8]:
```
vacuum_environment(simple_reflex_agent, n = 5, maxsteps = 10000, verbose = True)
```

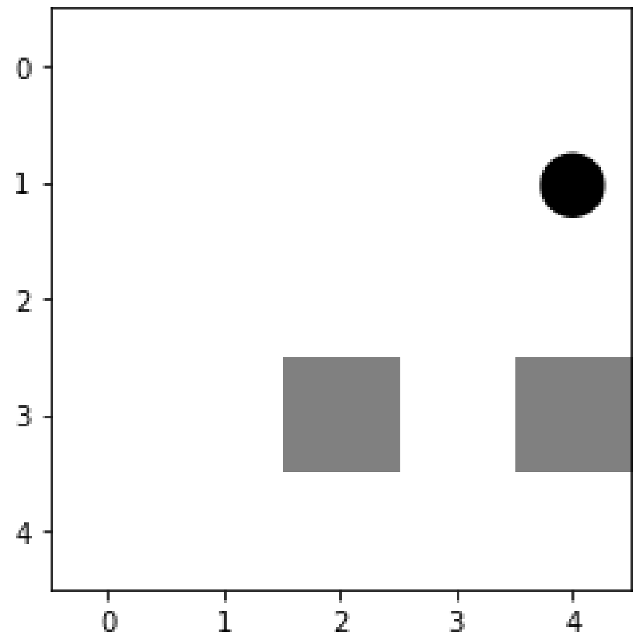
```
room:
 [[False False False False False]
 [False False False False False]
 [False False False False False]
 [False False  True False  True]
 [False False False False False]]
dirty squares: 2

start simulation
```



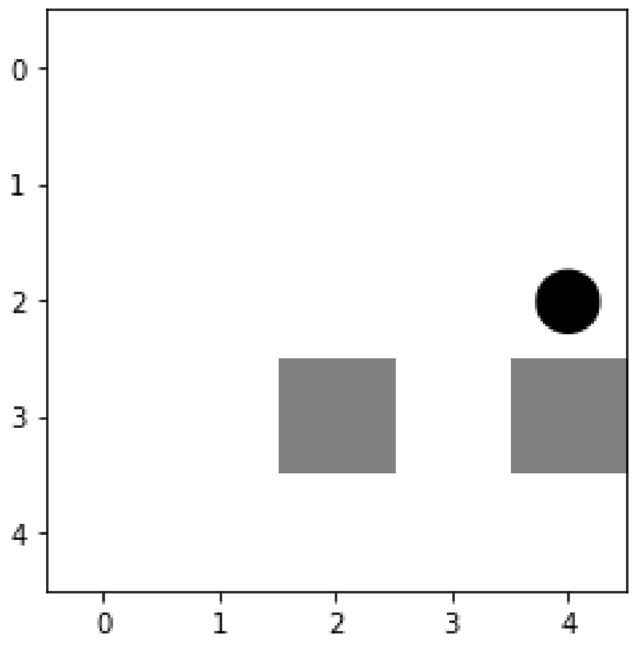
```
-----
step: 1
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



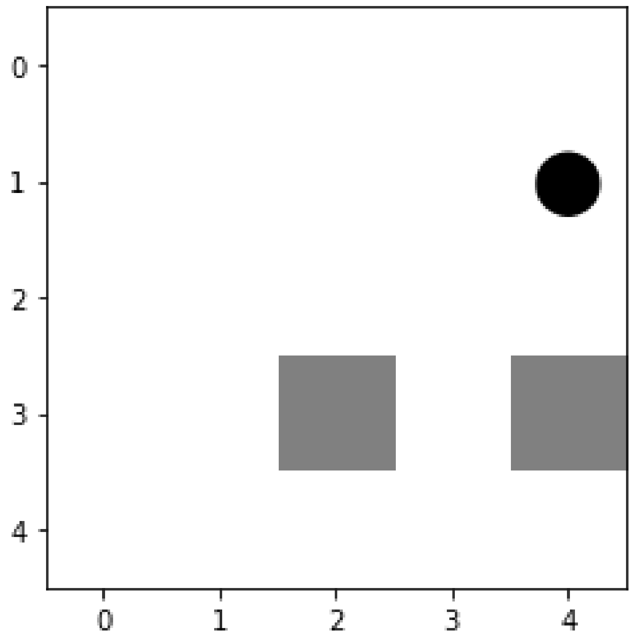
```
-----
step: 2
current position [0, 4]
bumper {'north': True, 'south': False, 'west': False, 'east': True}
dirty False
action south
```



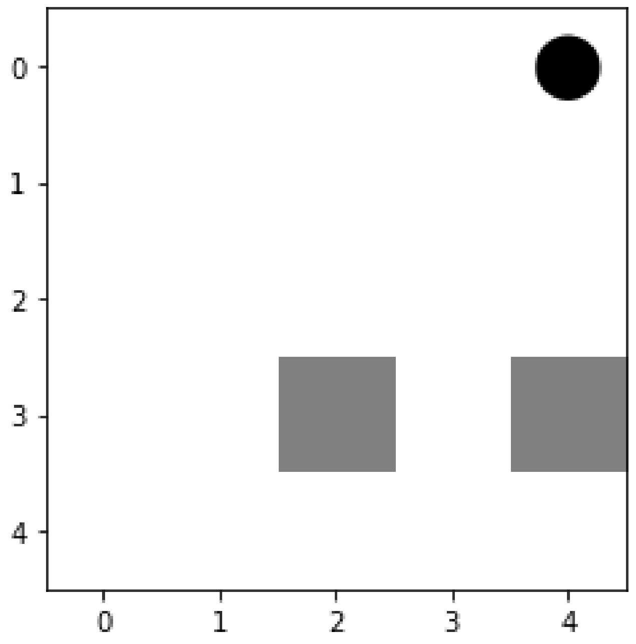
```
-----
step: 3
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action south
```



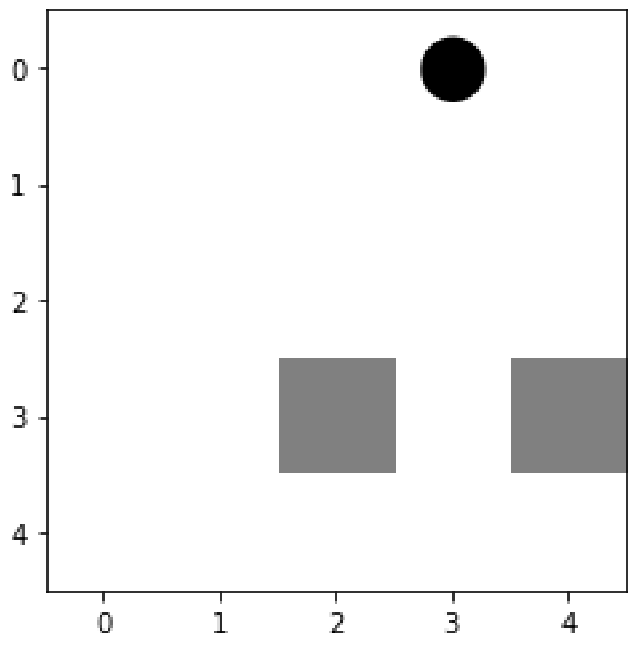
```
-----
step: 4
current position [2, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action north
```



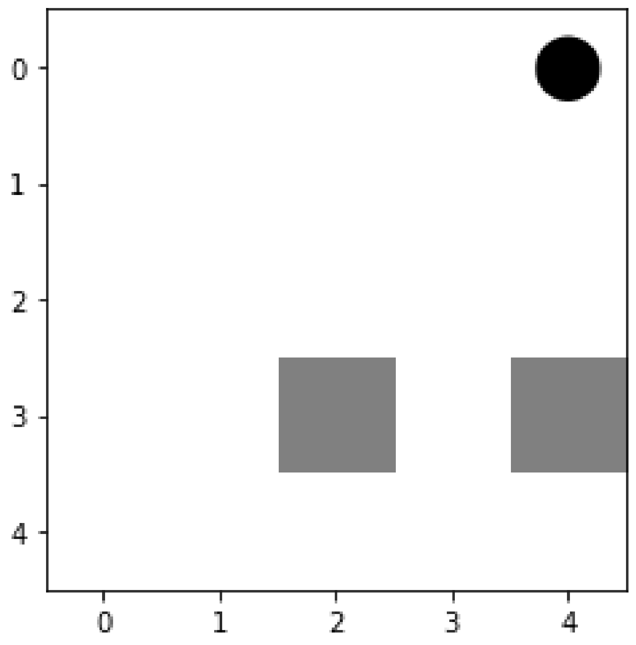
```
-----
step: 5
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action north
```



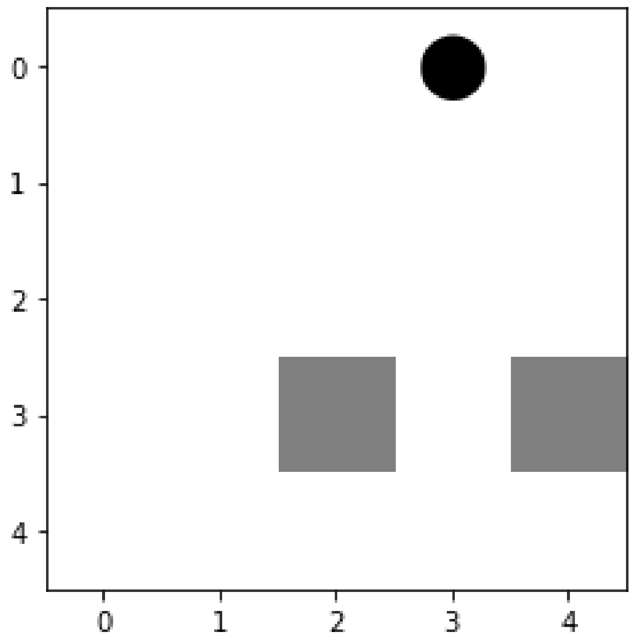
```
-----
step: 6
current position [0, 4]
bumper {'north': True, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



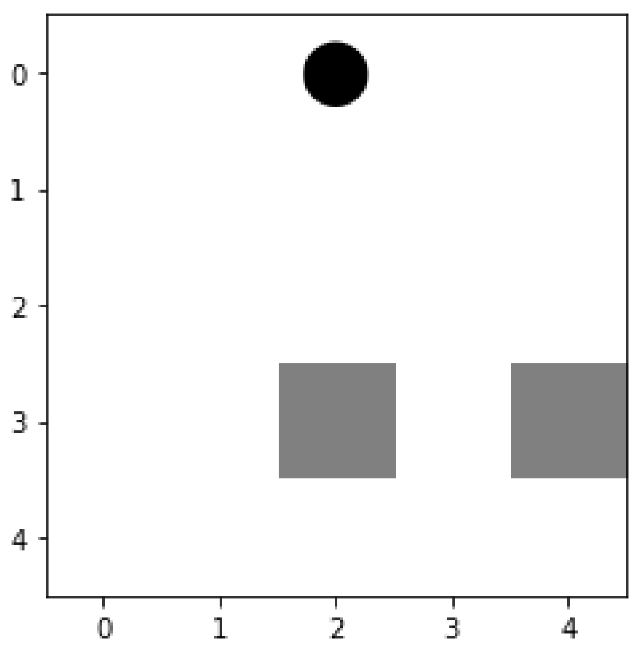
```
-----
step: 7
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



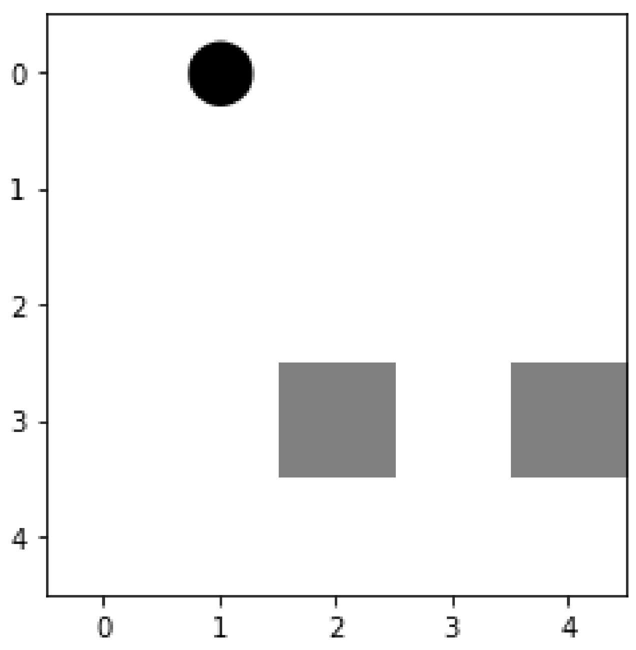
```
-----
step: 8
current position [0, 4]
bumper {'north': True, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



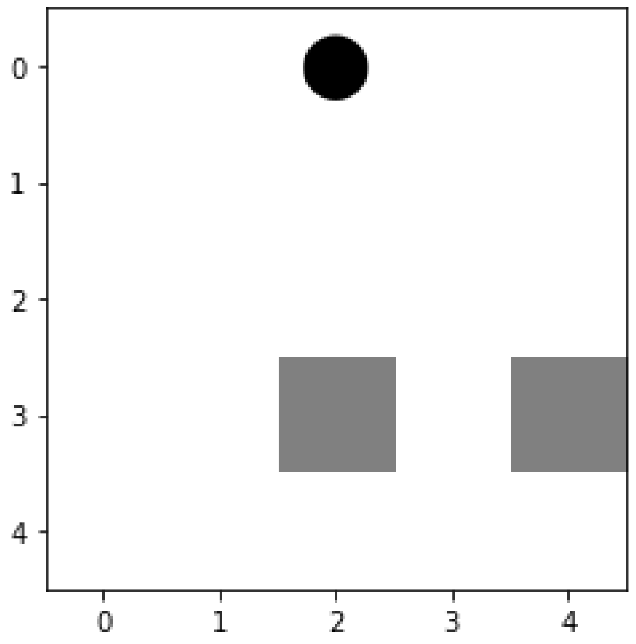
```
-----
step: 9
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



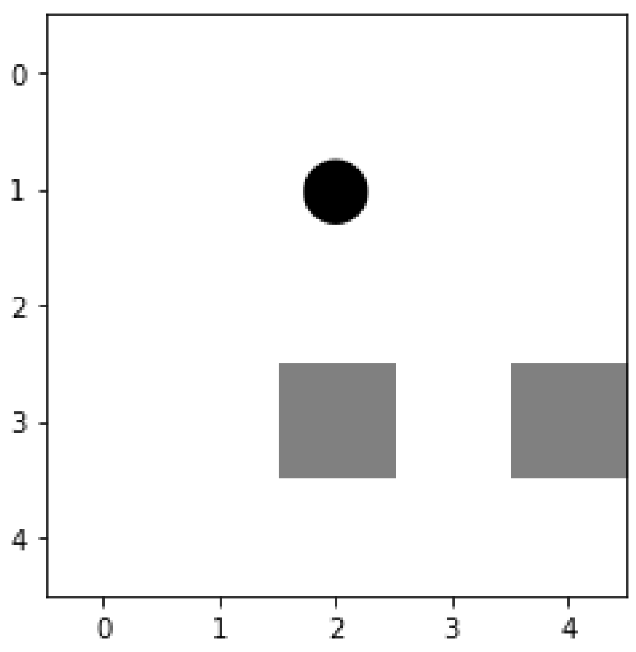
```
-----
step: 10
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



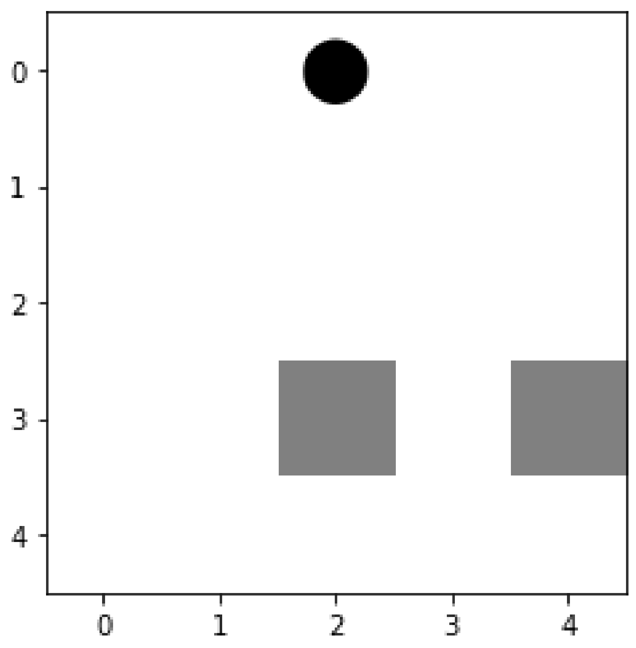
```
-----
step: 11
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



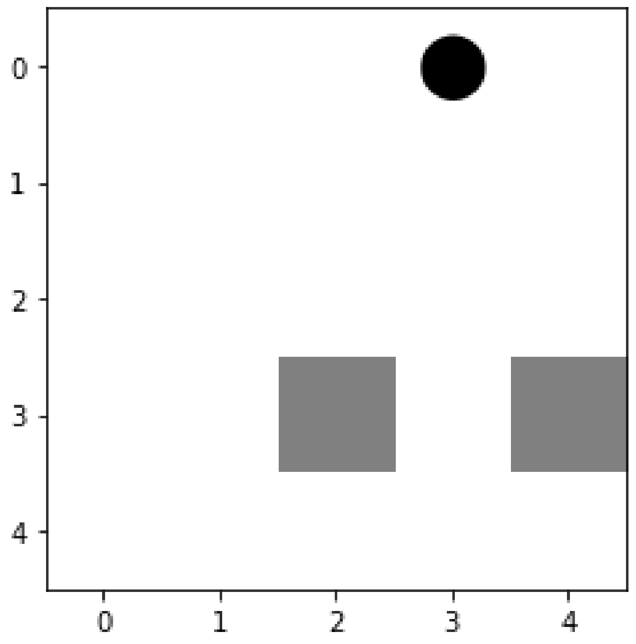
```
-----
step: 12
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



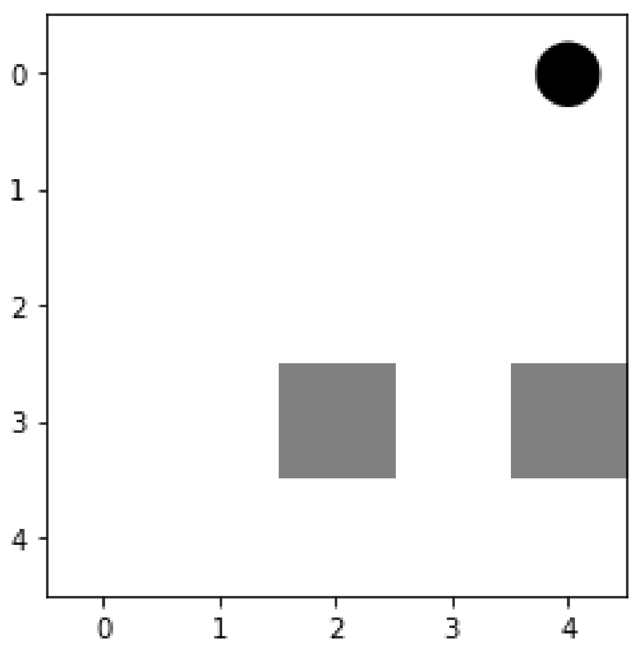
```
-----
step: 13
current position [1, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



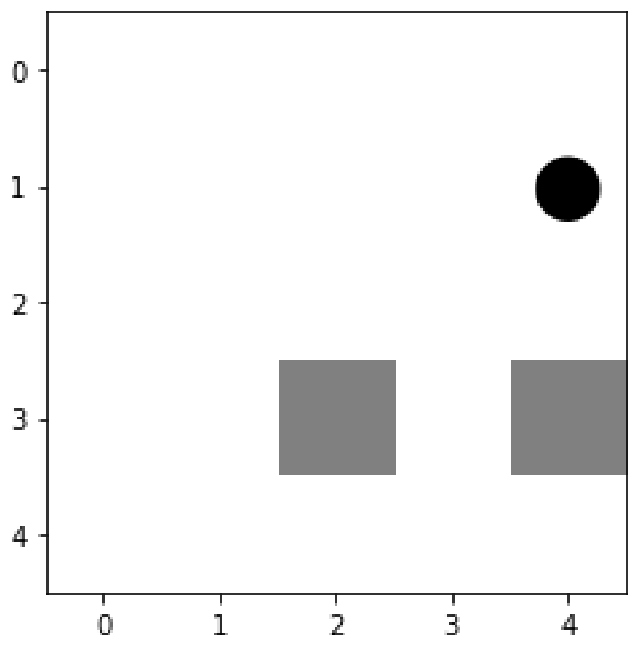
```
-----
step: 14
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



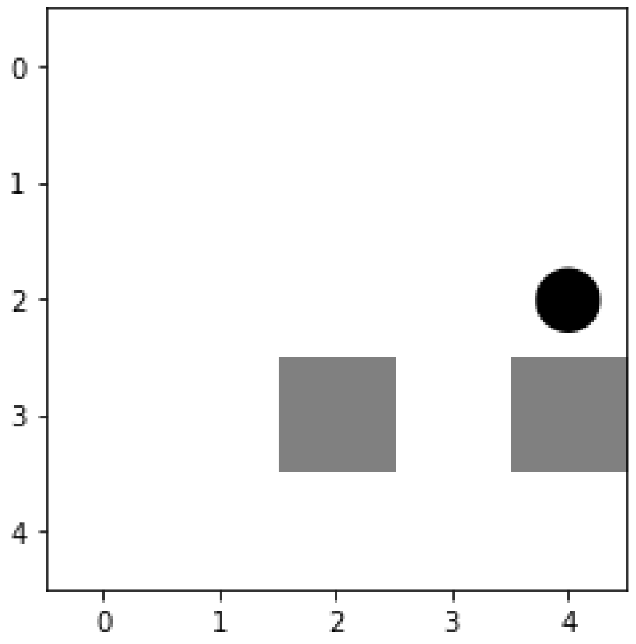
```
-----
step: 15
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



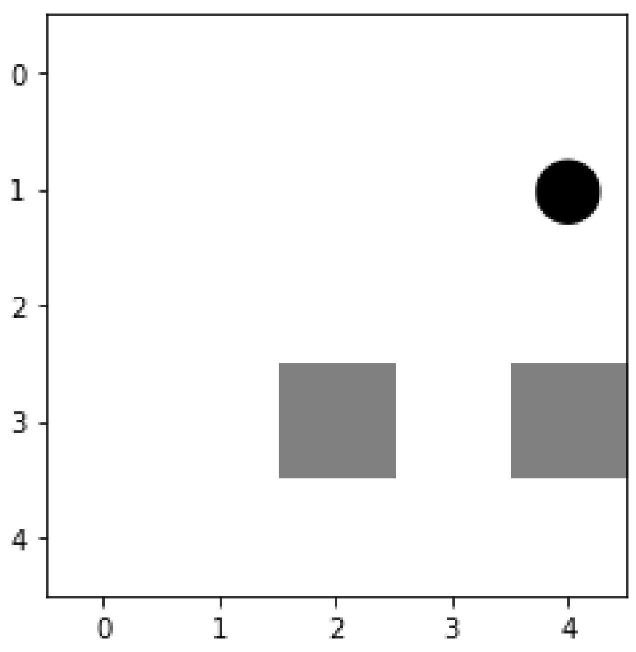
```
-----
step: 16
current position [0, 4]
bumper {'north': True, 'south': False, 'west': False, 'east': True}
dirty False
action south
```



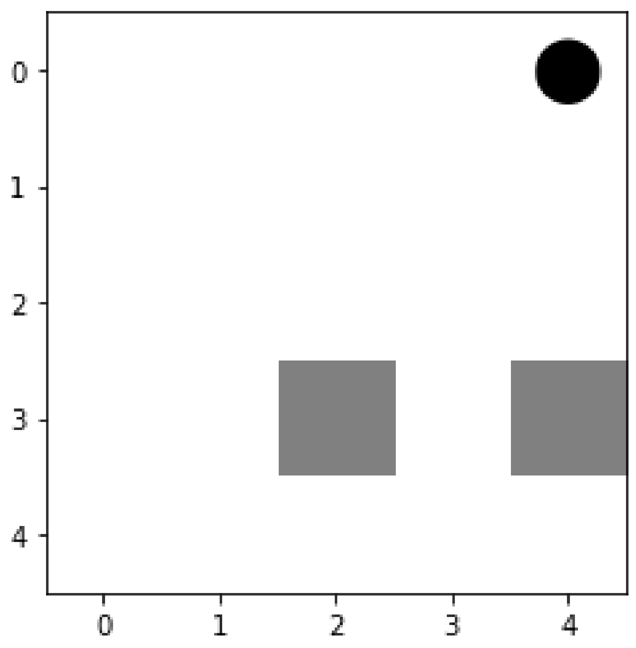
```
-----
step: 17
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action south
```



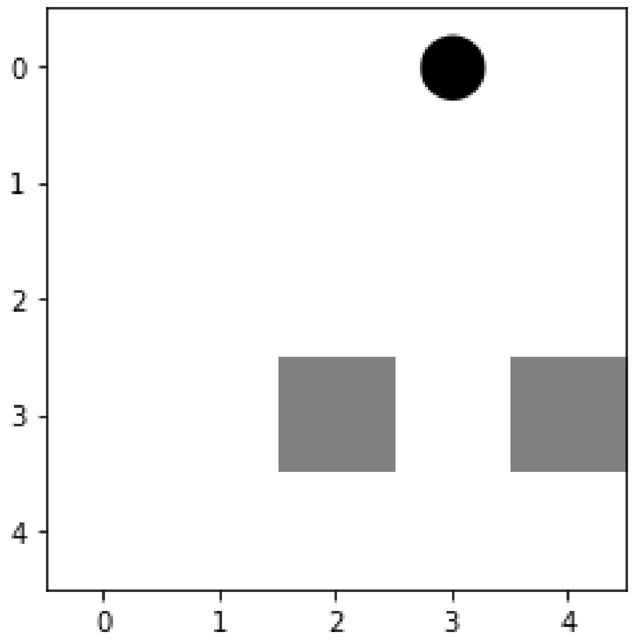
```
-----
step: 18
current position [2, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action north
```



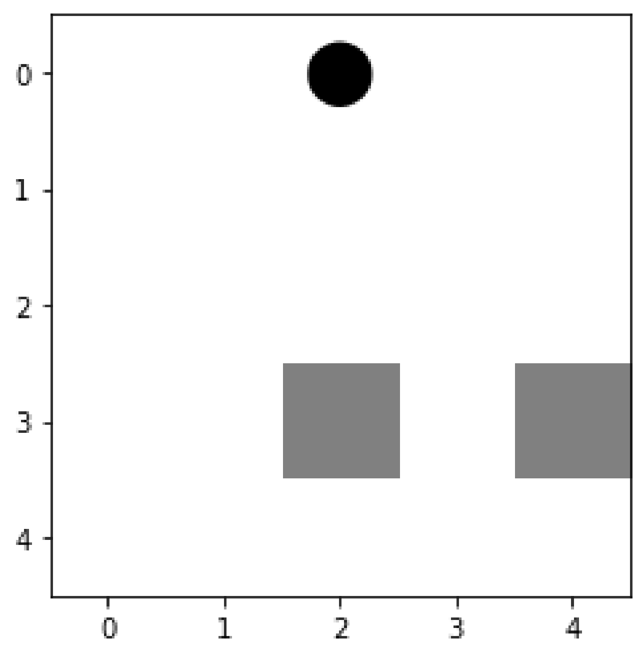
```
-----
step: 19
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action north
```



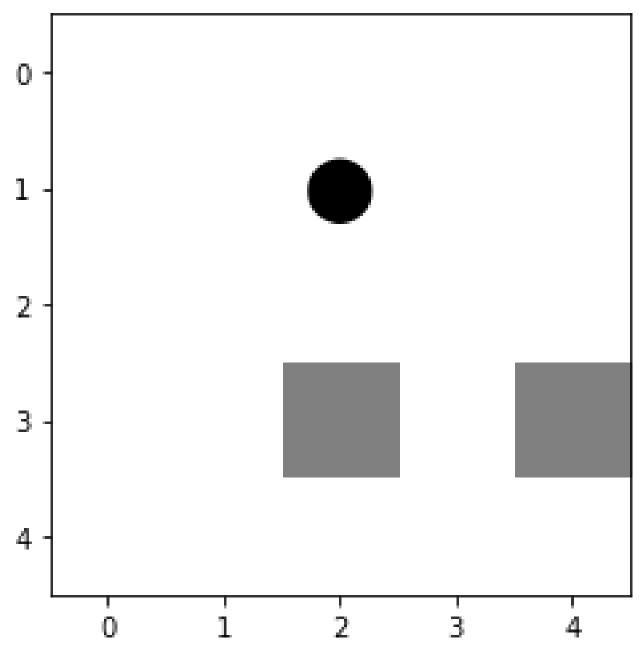
```
-----
step: 20
current position [0, 4]
bumper {'north': True, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



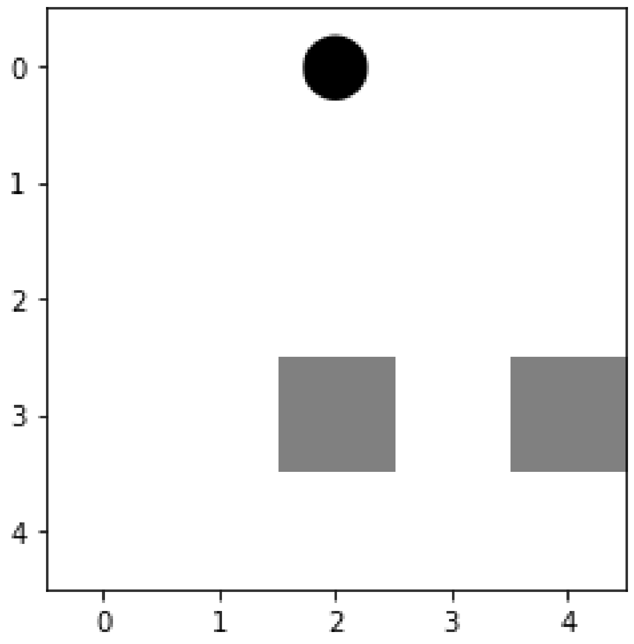
```
-----
step: 21
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



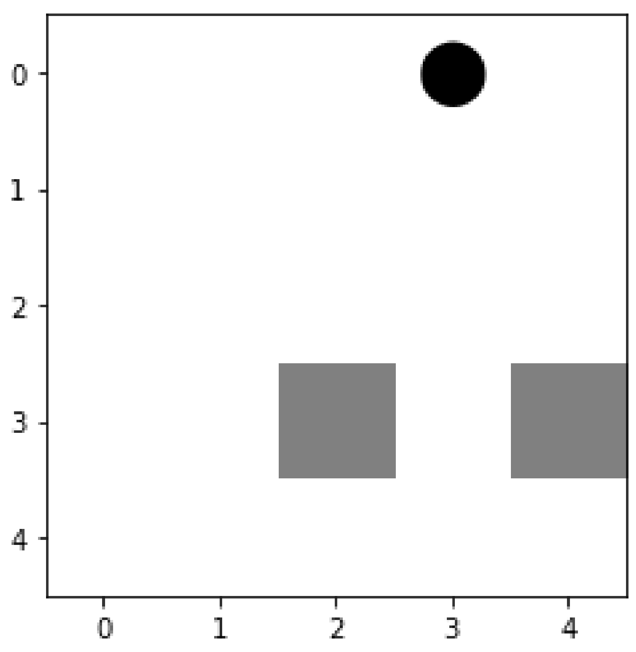
```
-----
step: 22
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



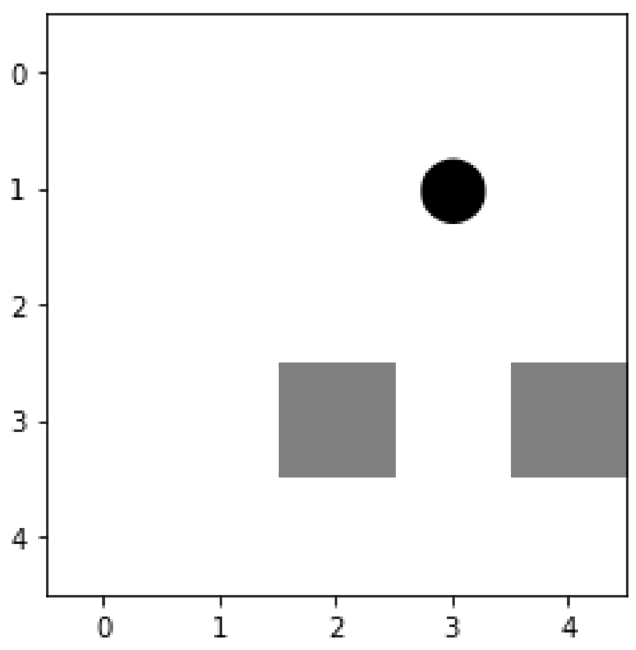
```
-----
step: 23
current position [1, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



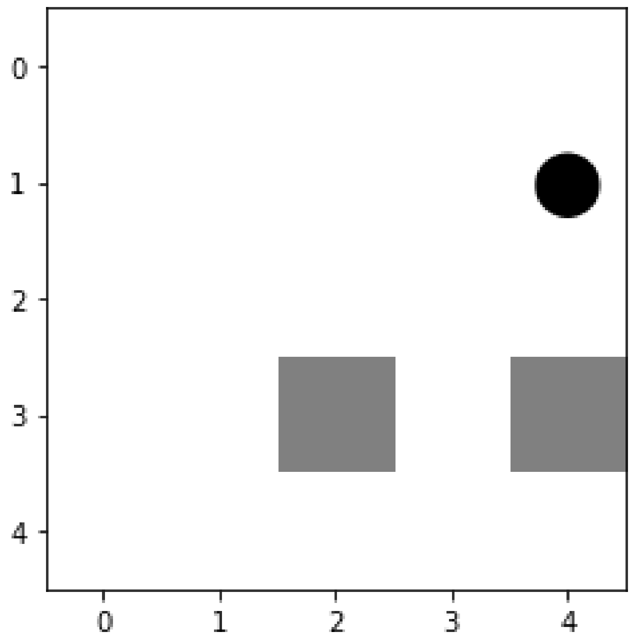
```
-----
step: 24
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



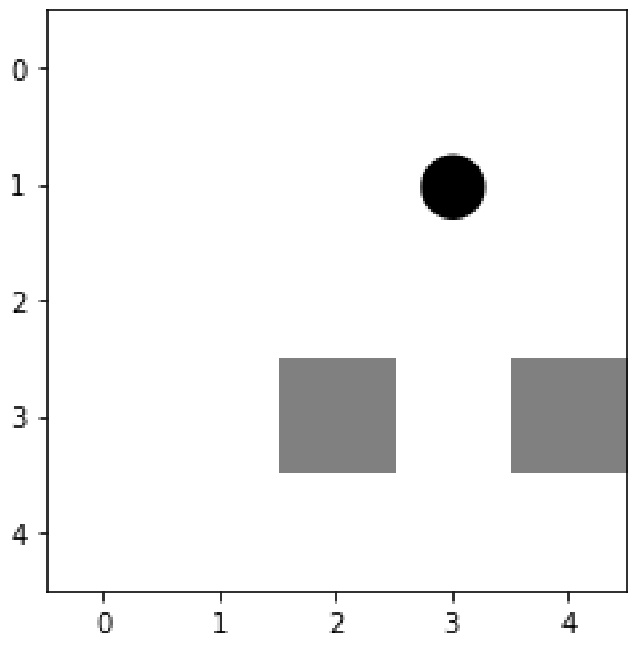
```
-----
step: 25
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



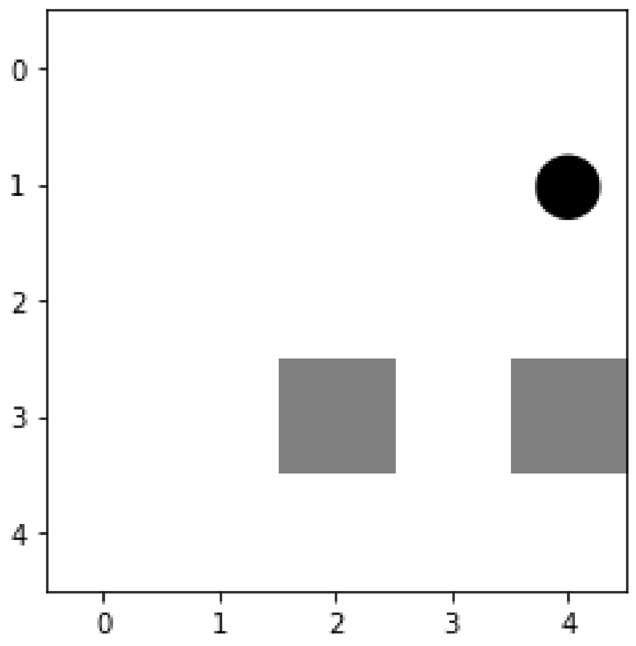
```
-----
step: 26
current position [1, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



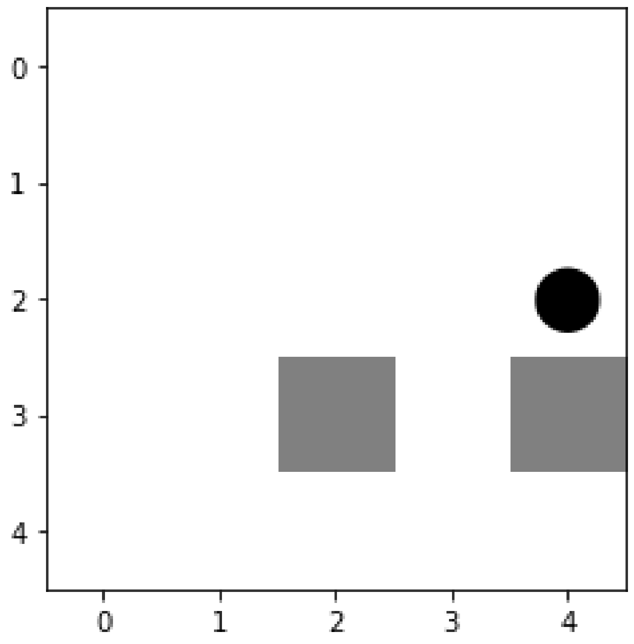
```
-----
step: 27
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



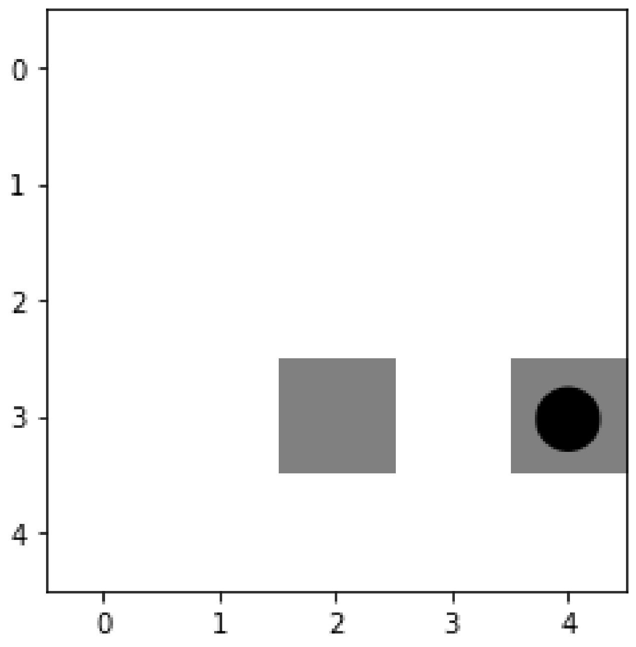
```
-----
step: 28
current position [1, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



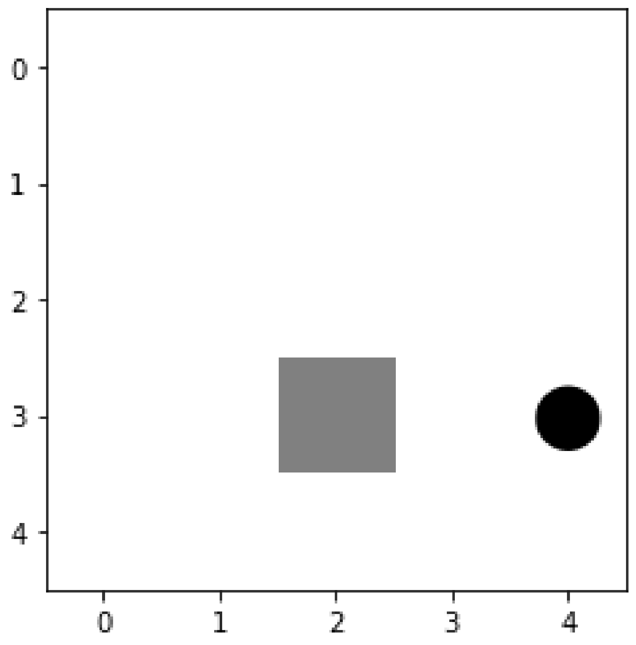
```
-----
step: 29
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action south
```



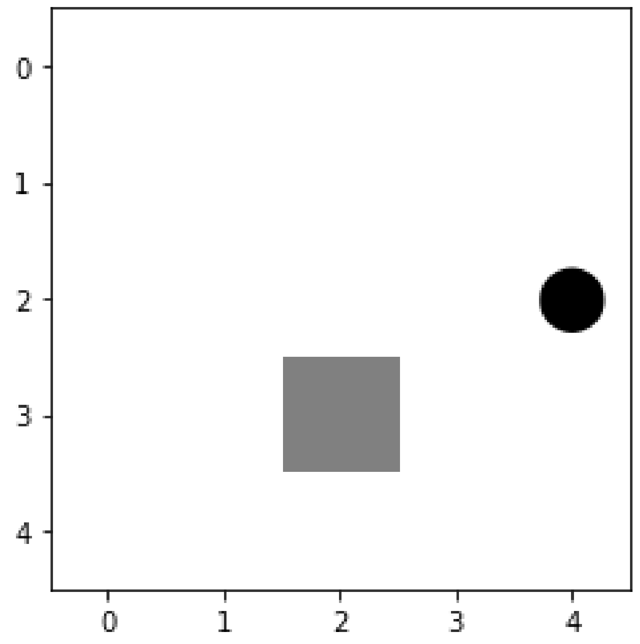
```
-----
step: 30
current position [2, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action south
```



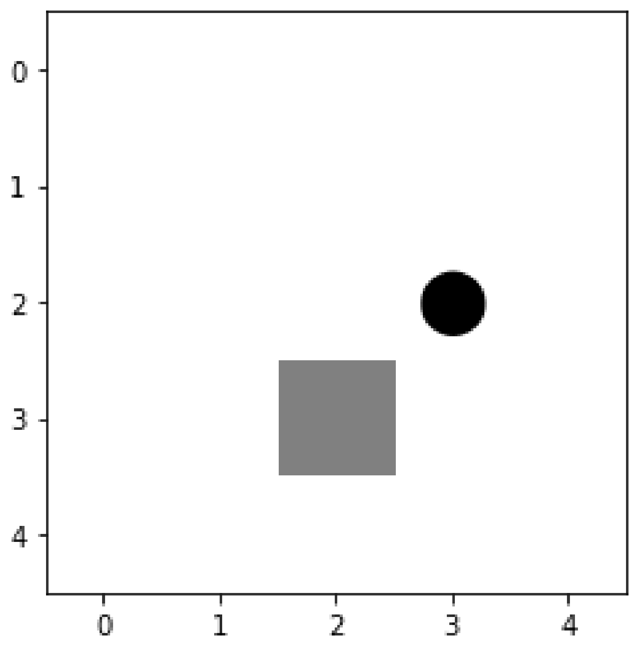
```
-----
step: 31
current position [3, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty True
action suck
```



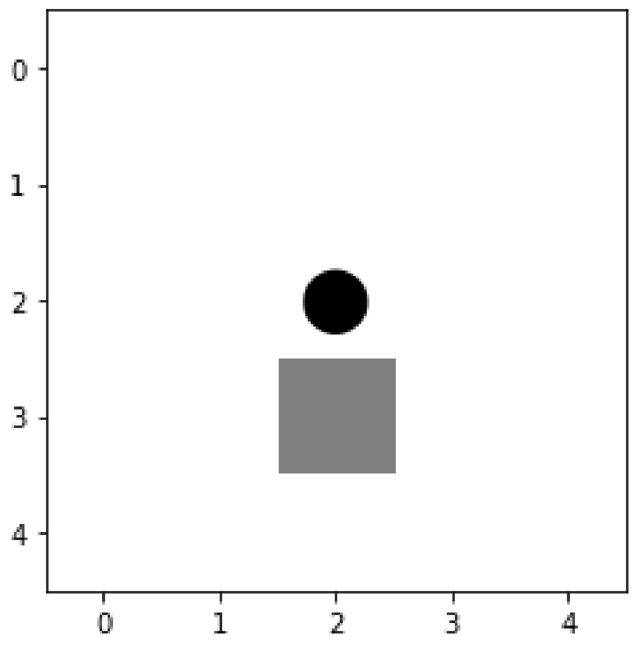
```
-----
step: 32
current position [3, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action north
```



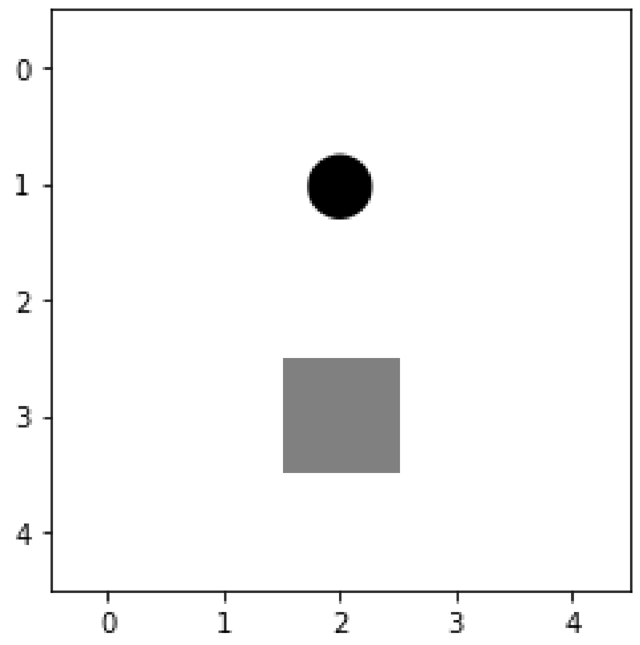
```
-----
step: 33
current position [2, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



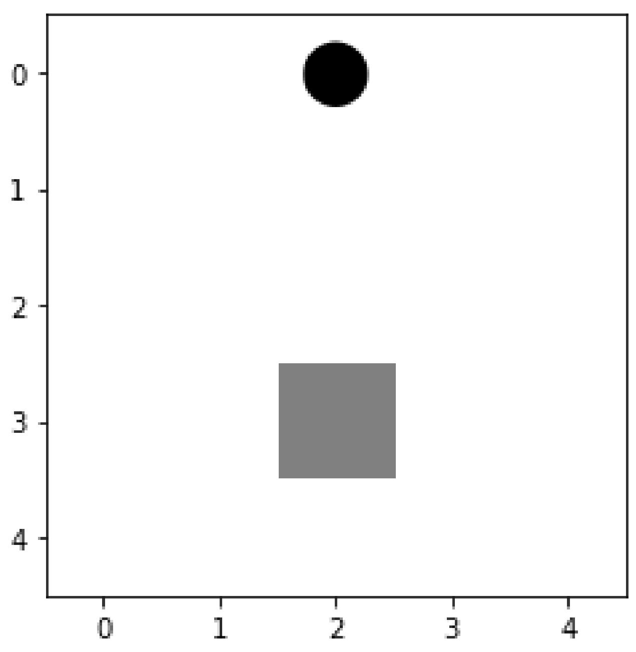
```
-----
step: 34
current position [2, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



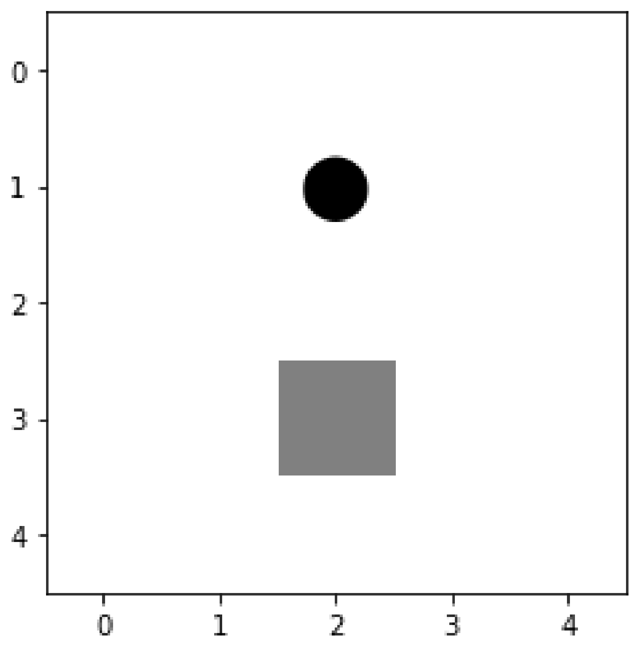
```
-----
step: 35
current position [2, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



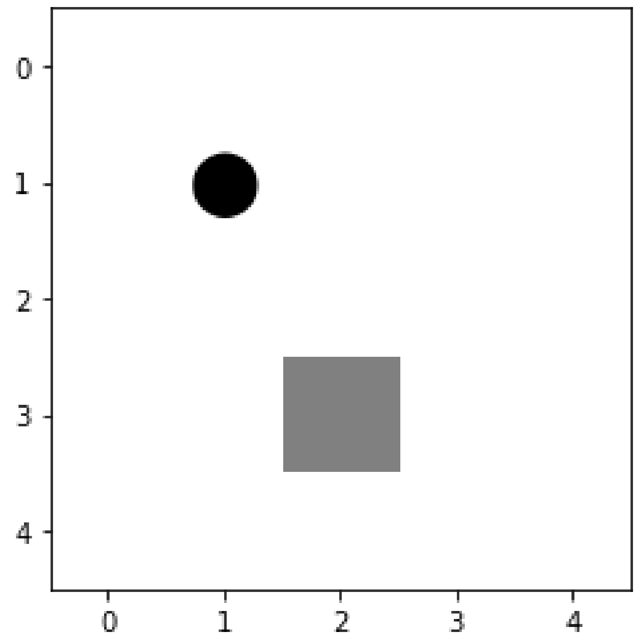
```
-----
step: 36
current position [1, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



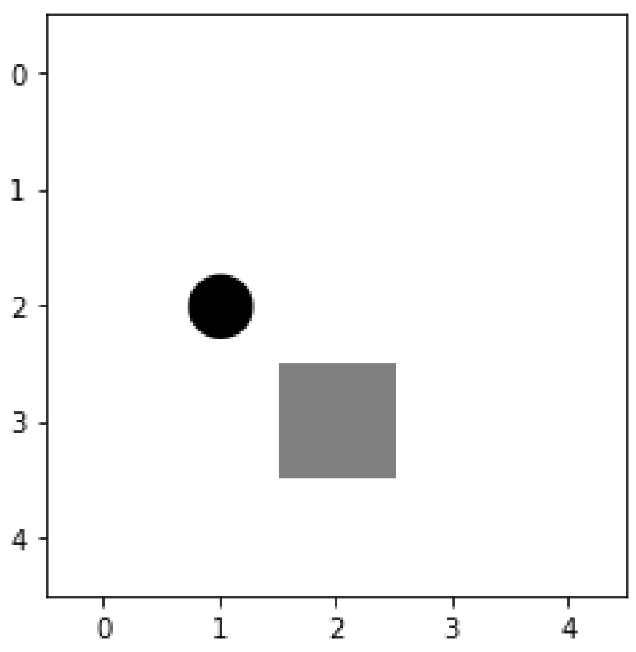
```
-----
step: 37
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



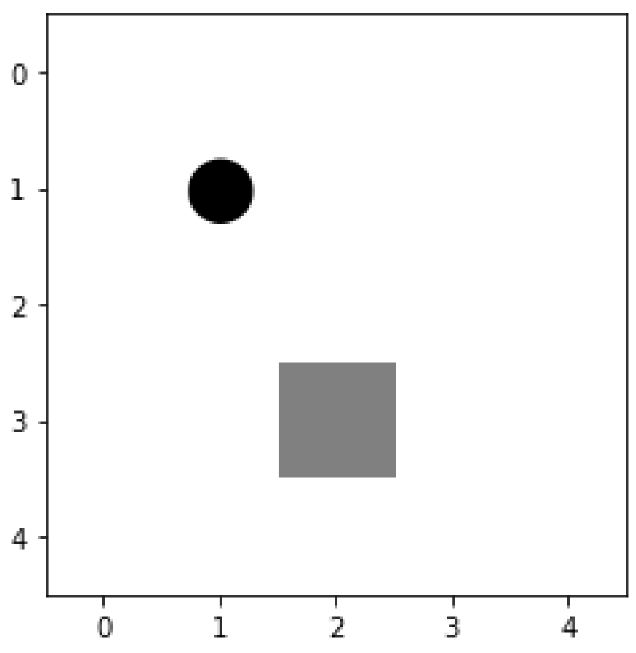
```
-----
step: 38
current position [1, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



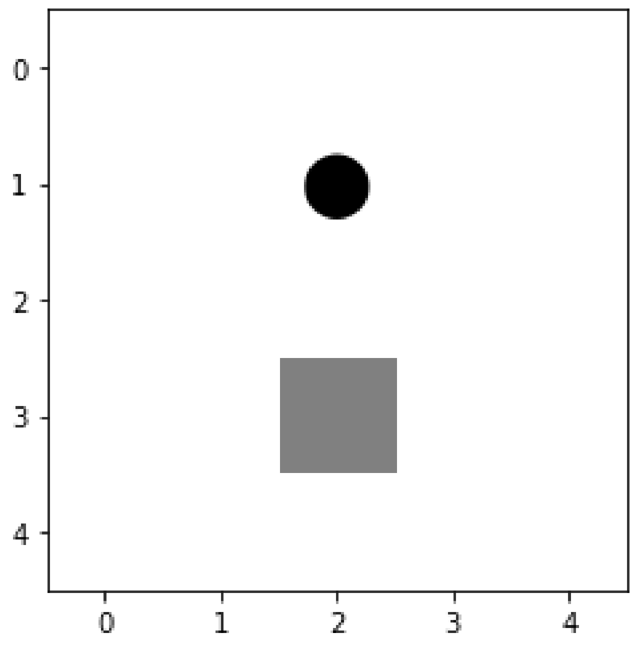
```
-----
step: 39
current position [1, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



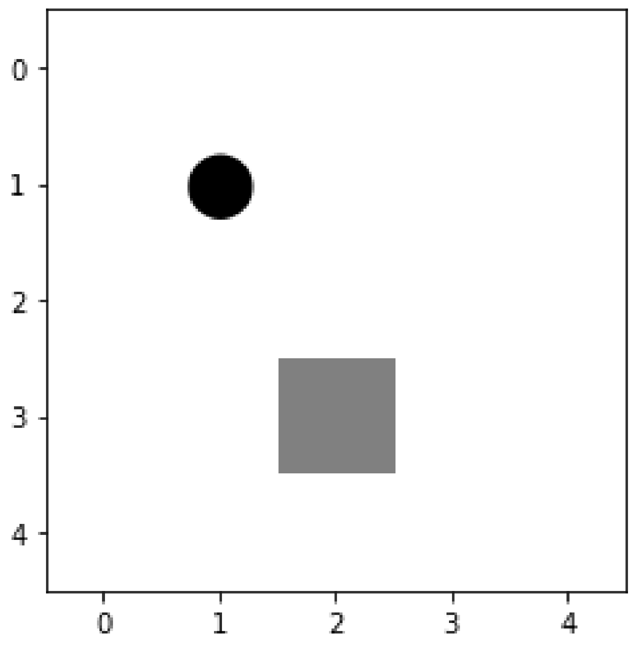
```
-----
step: 40
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



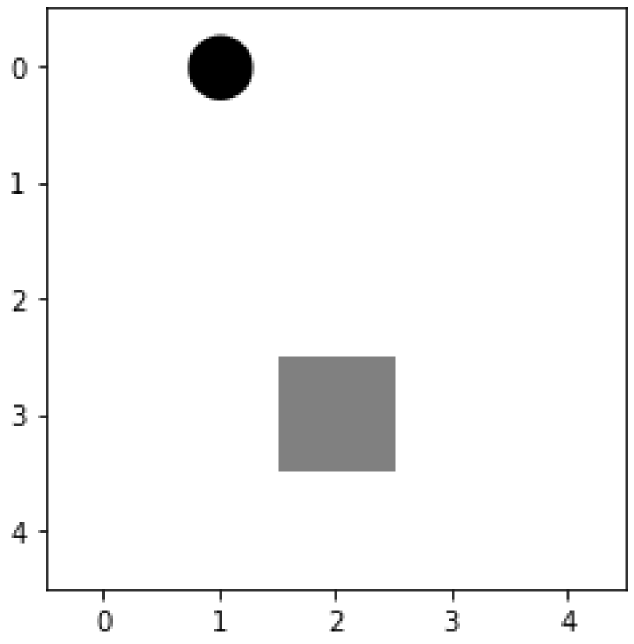
```
-----
step: 41
current position [1, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



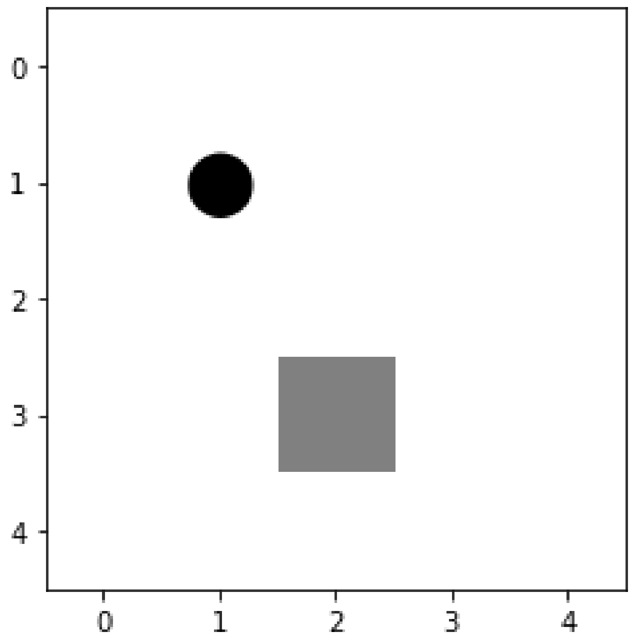
```
-----
step: 42
current position [1, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



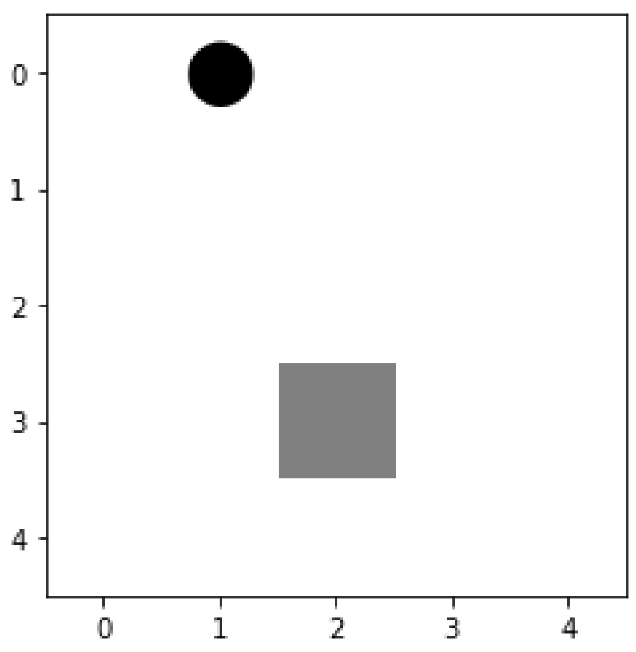
```
-----
step: 43
current position [1, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



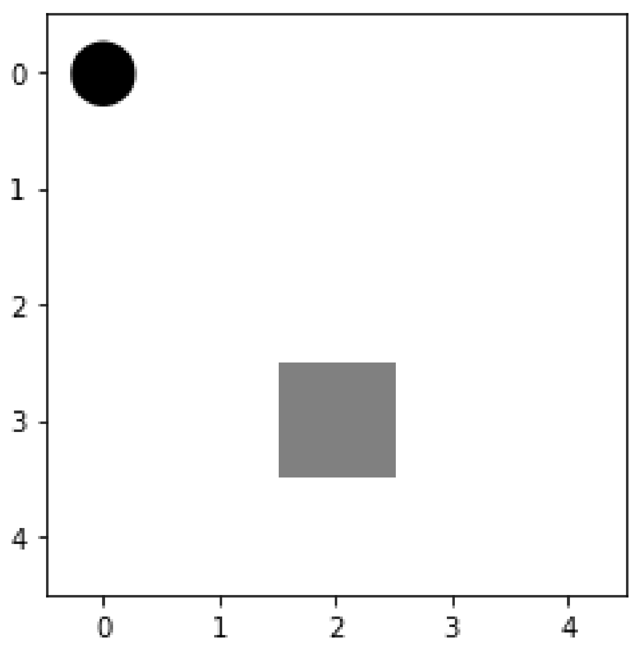
```
-----
step: 44
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



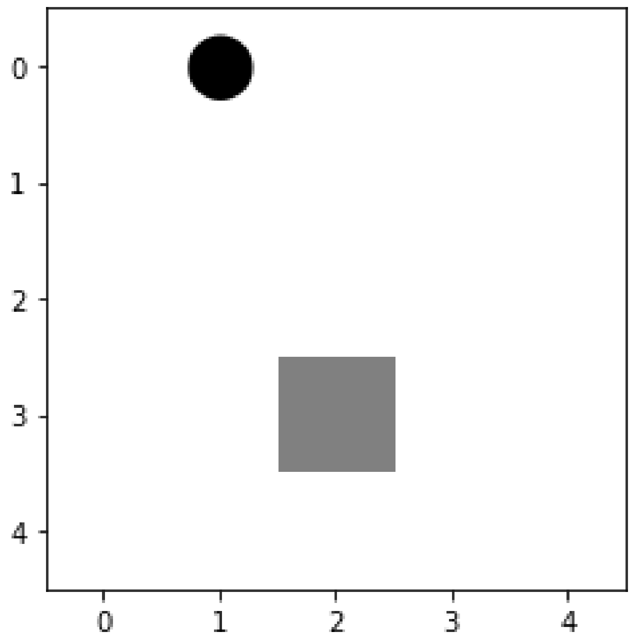
```
-----
step: 45
current position [1, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



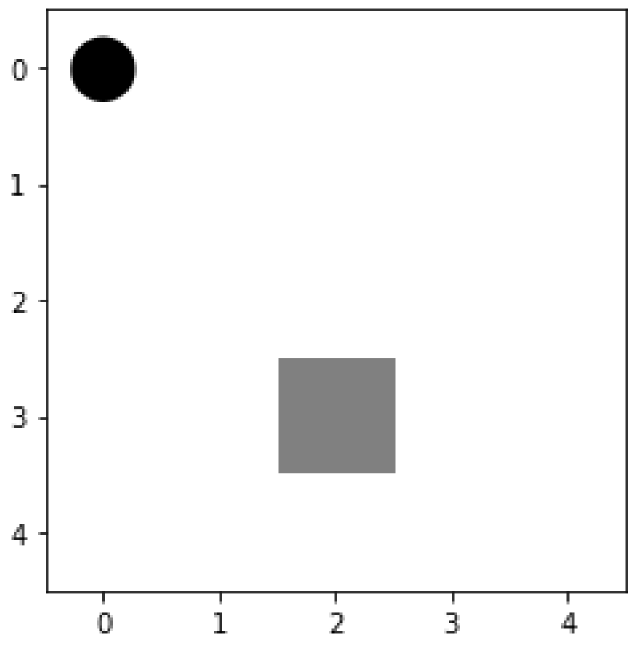
```
-----
step: 46
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



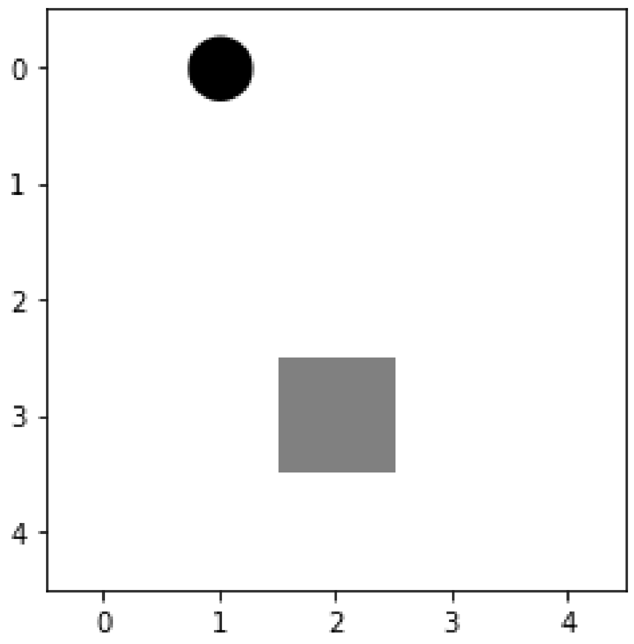
```
-----
step: 47
current position [0, 0]
bumper {'north': True, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



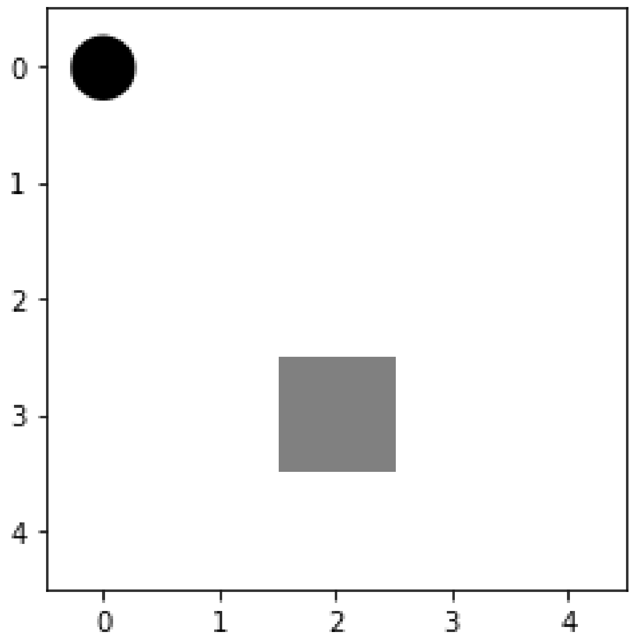
```
-----
step: 48
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



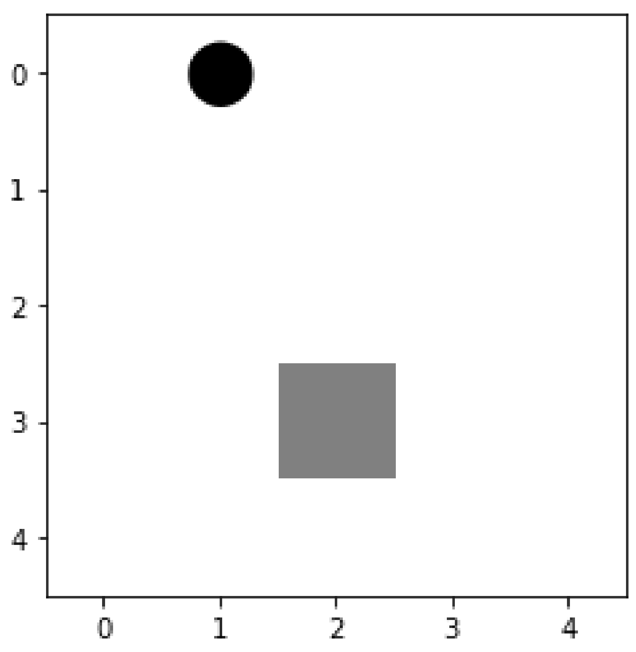
```
-----
step: 49
current position [0, 0]
bumper {'north': True, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



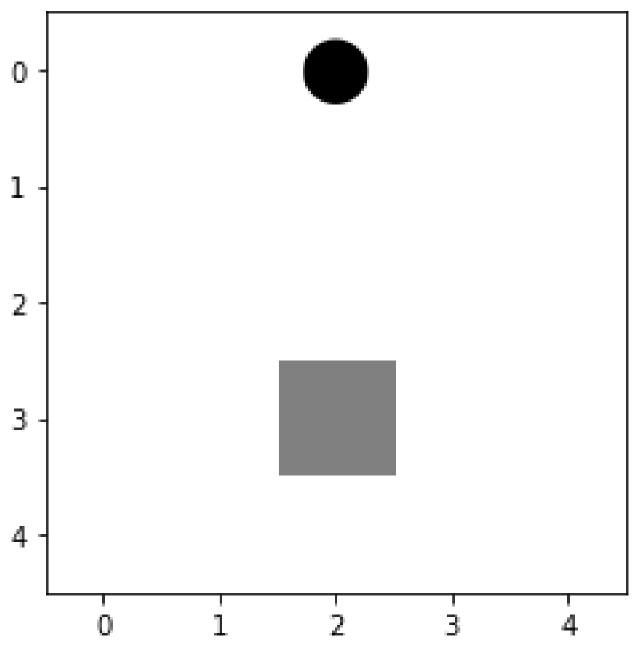
```
-----
step: 50
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



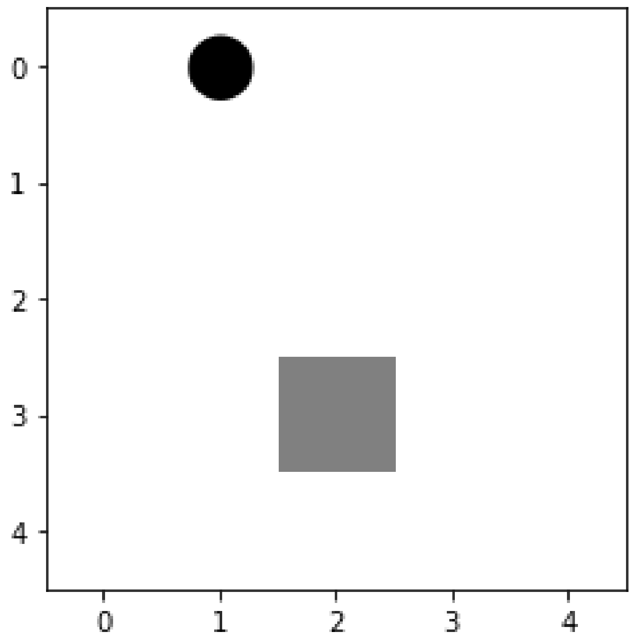
```
-----
step: 51
current position [0, 0]
bumper {'north': True, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



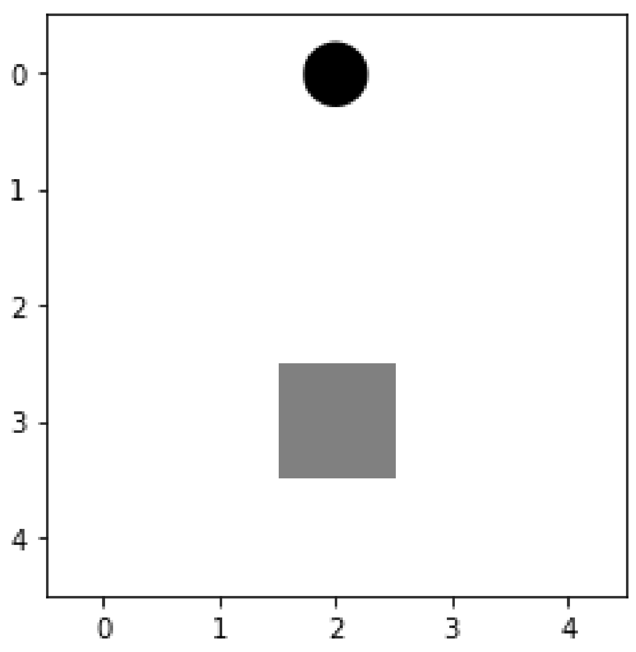
```
-----
step: 52
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



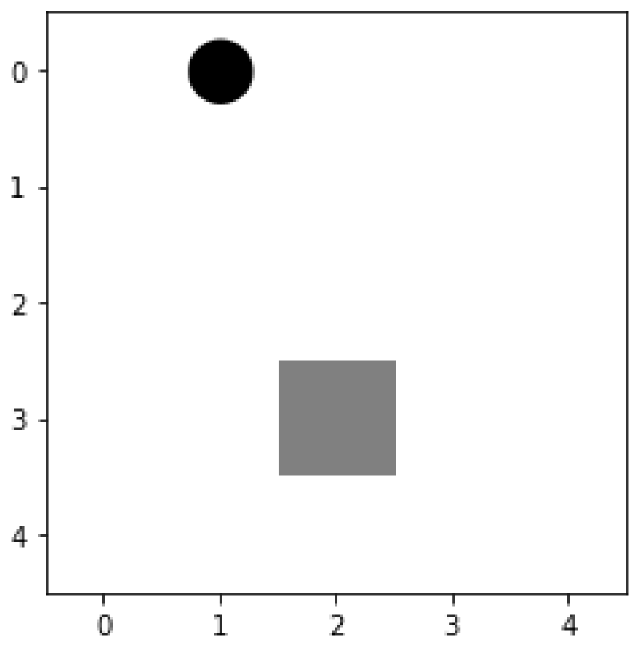
```
-----
step: 53
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



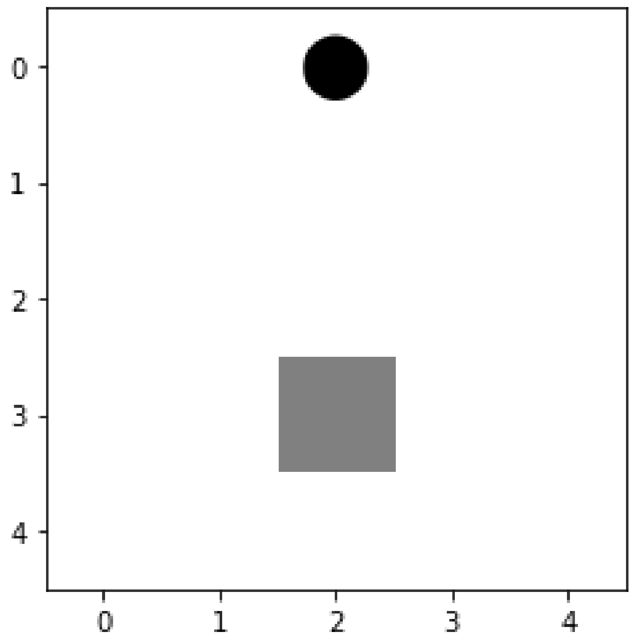
```
-----
step: 54
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



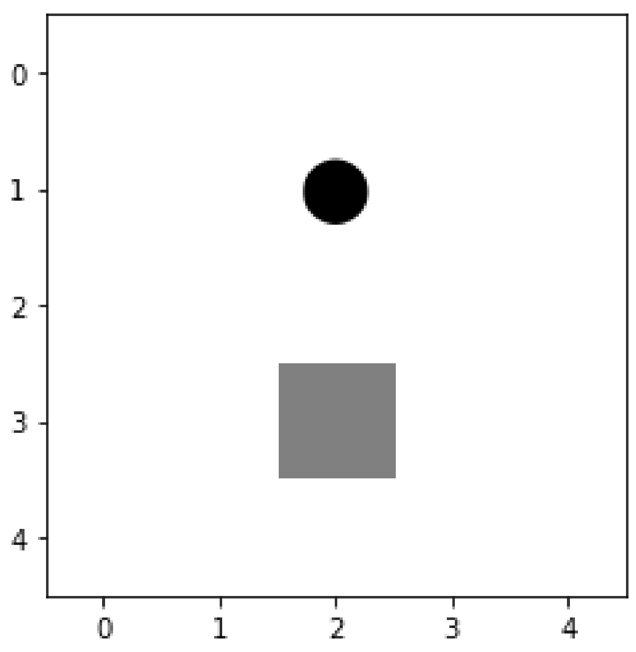
```
-----
step: 55
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



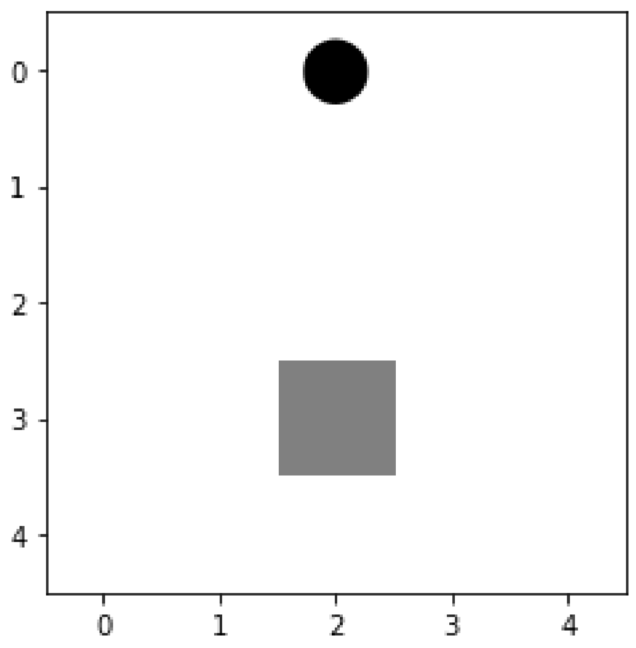
```
-----
step: 56
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



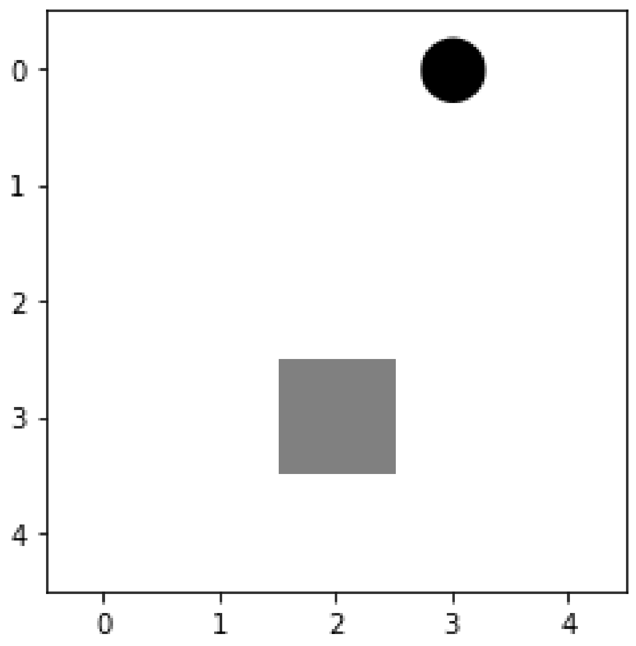
```
-----
step: 57
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



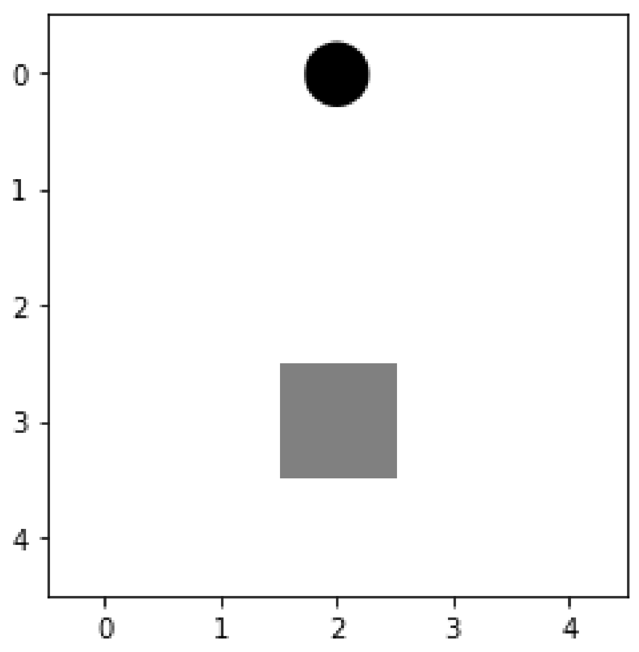
```
-----
step: 58
current position [1, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



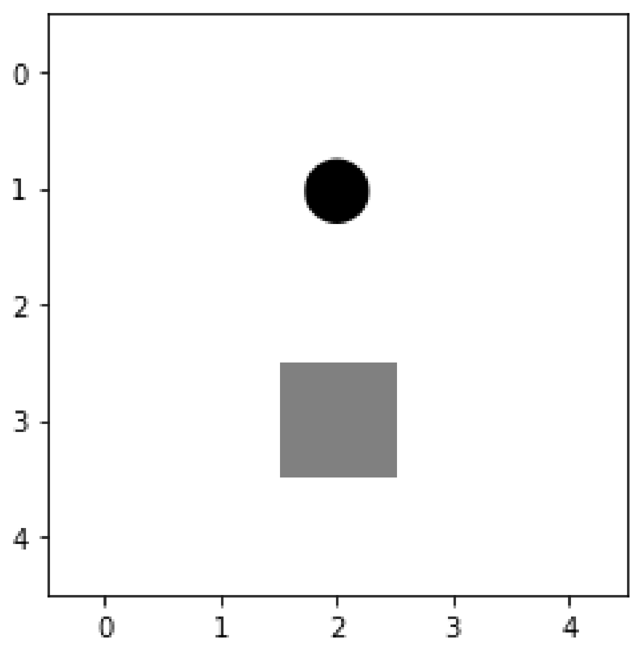
```
-----
step: 59
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



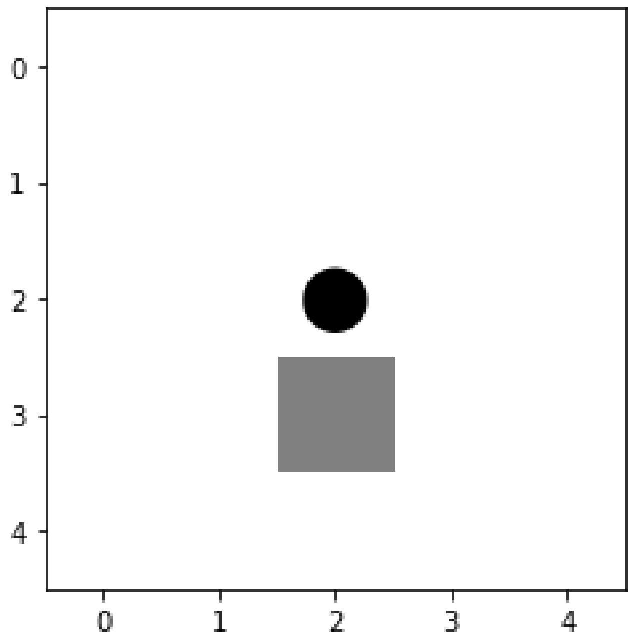
```
-----
step: 60
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



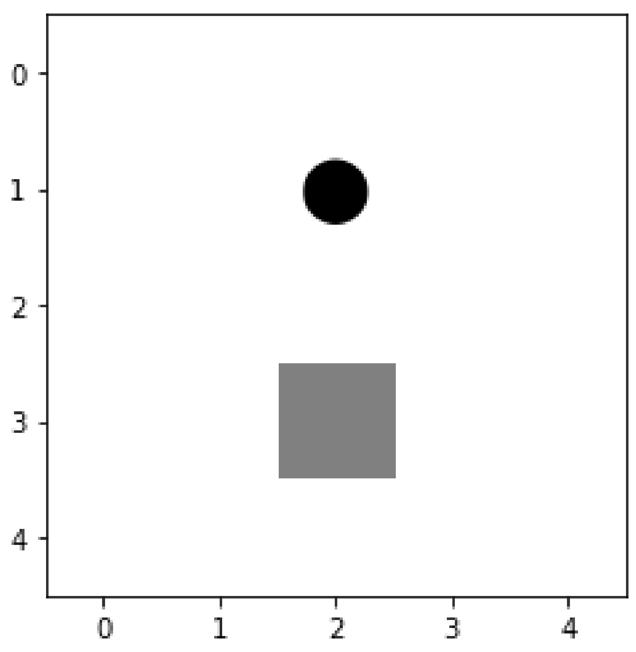
```
-----
step: 61
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



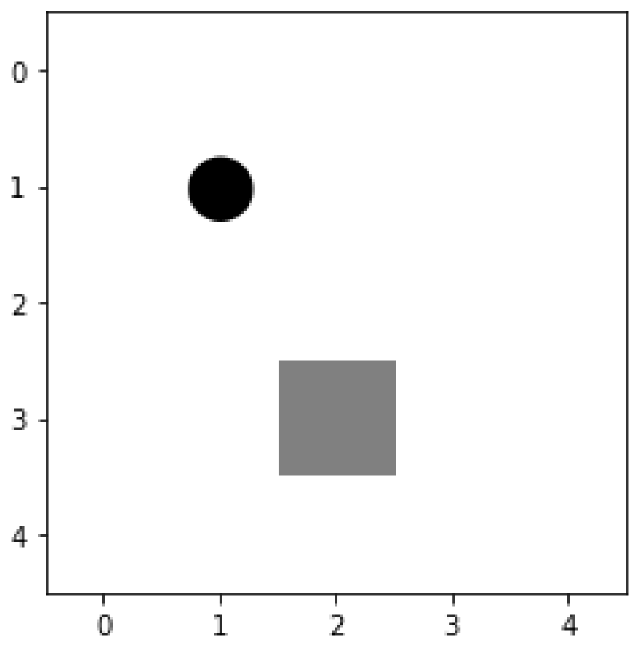
```
-----
step: 62
current position [1, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



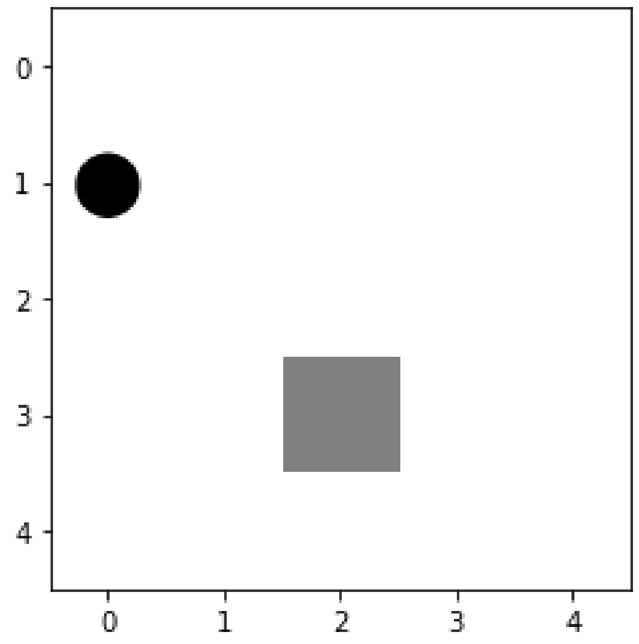
```
-----
step: 63
current position [2, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



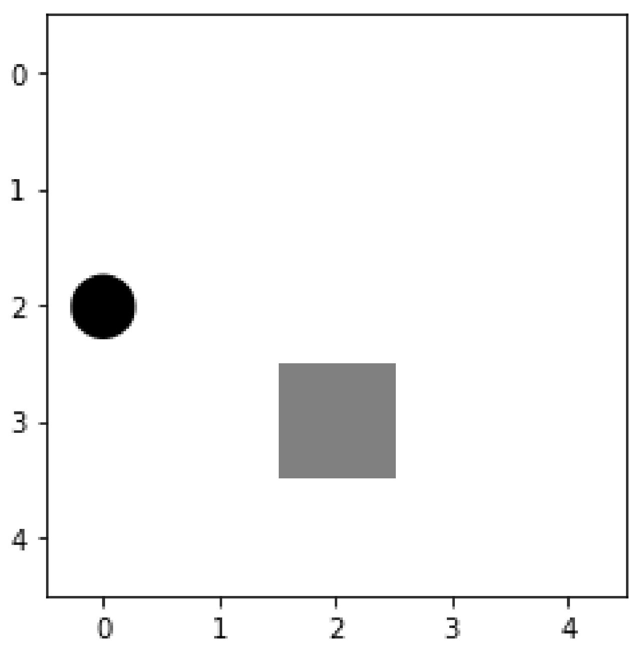
```
-----
step: 64
current position [1, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



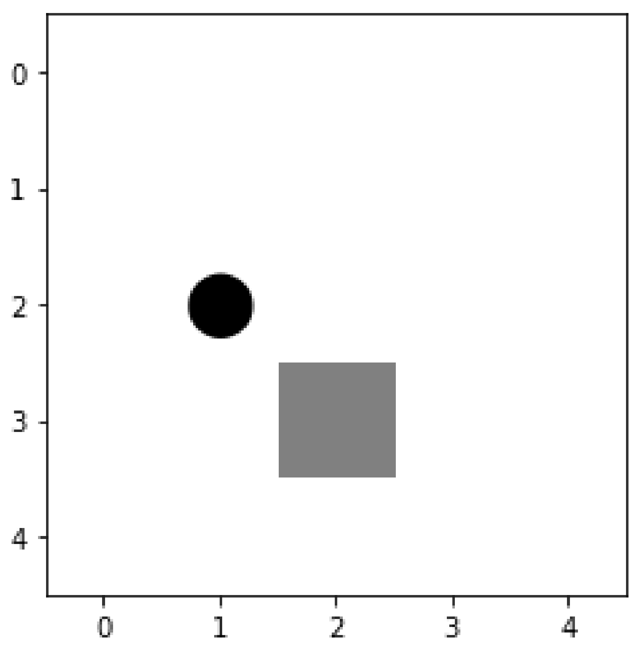
```
-----
step: 65
current position [1, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



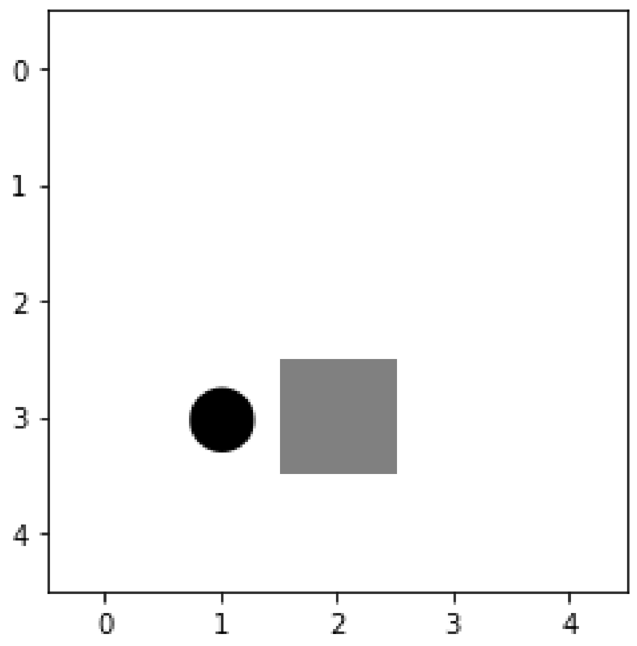
```
-----
step: 66
current position [1, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action south
```



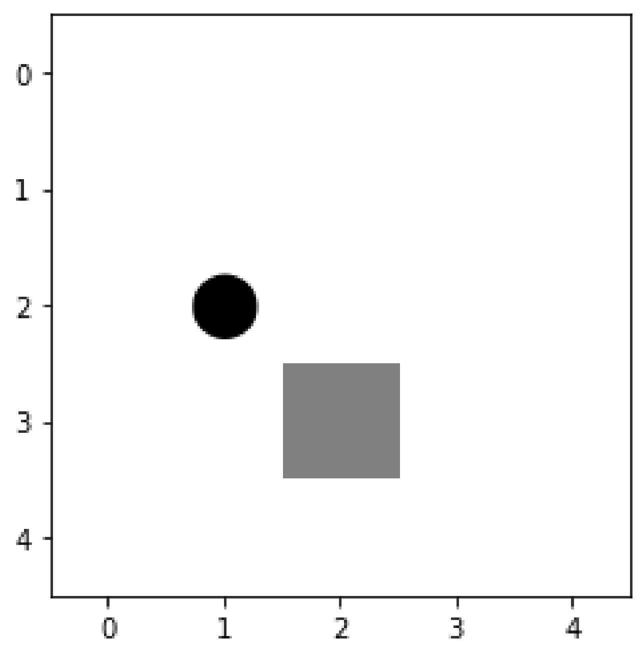
```
-----
step: 67
current position [2, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



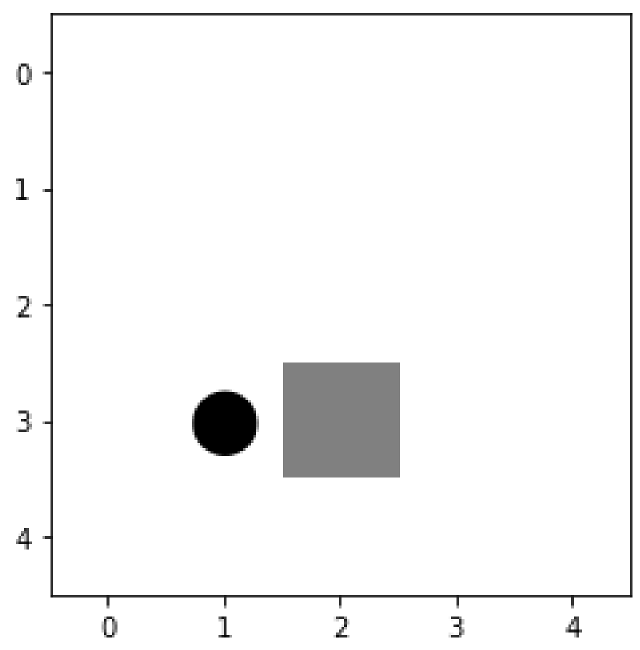
```
-----
step: 68
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



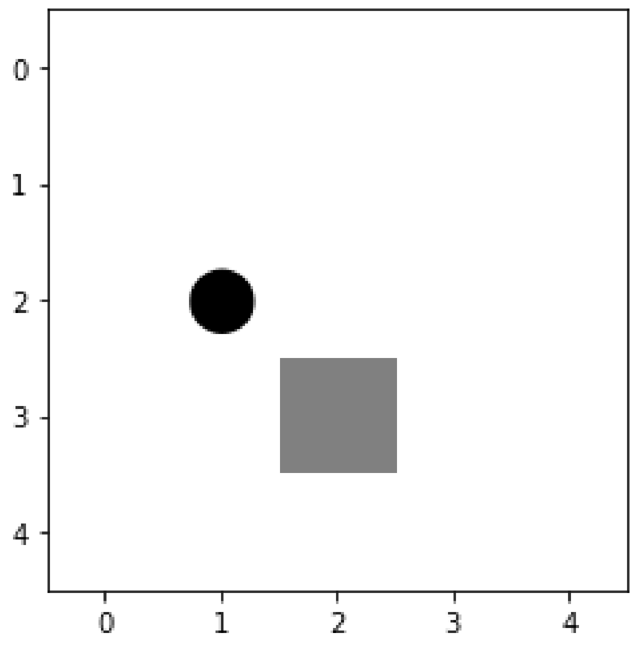
```
-----
step: 69
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



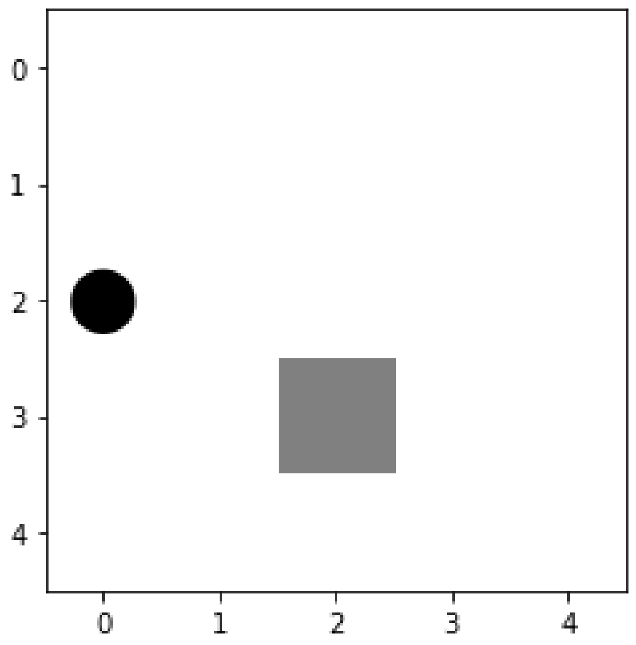
```
-----
step: 70
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



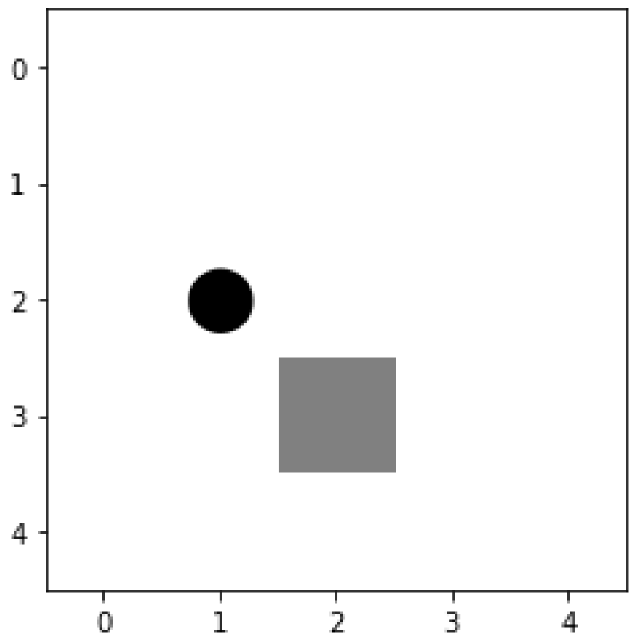
```
-----
step: 71
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action north
```



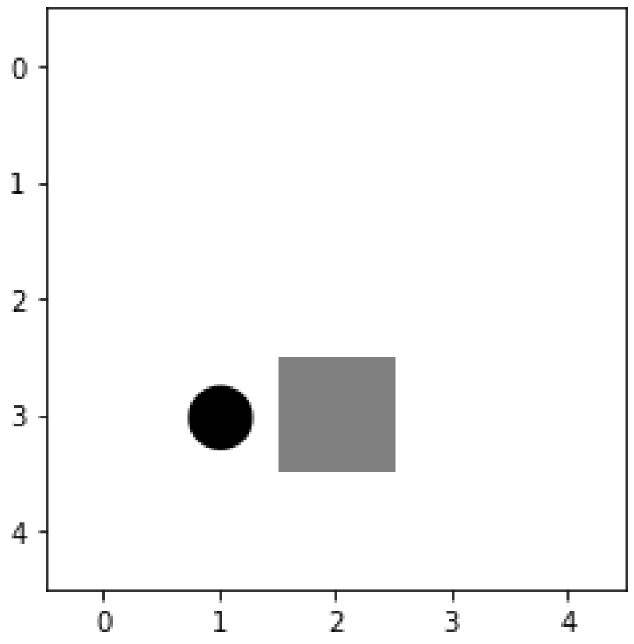
```
-----
step: 72
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



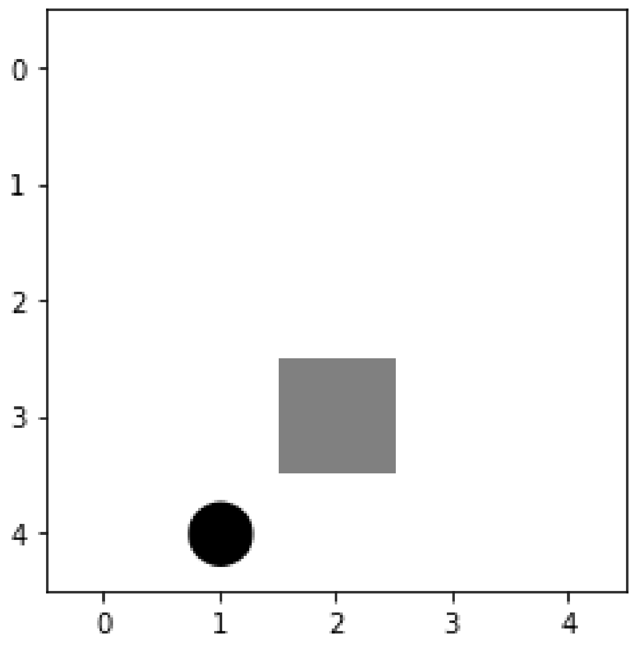
```
-----
step: 73
current position [2, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



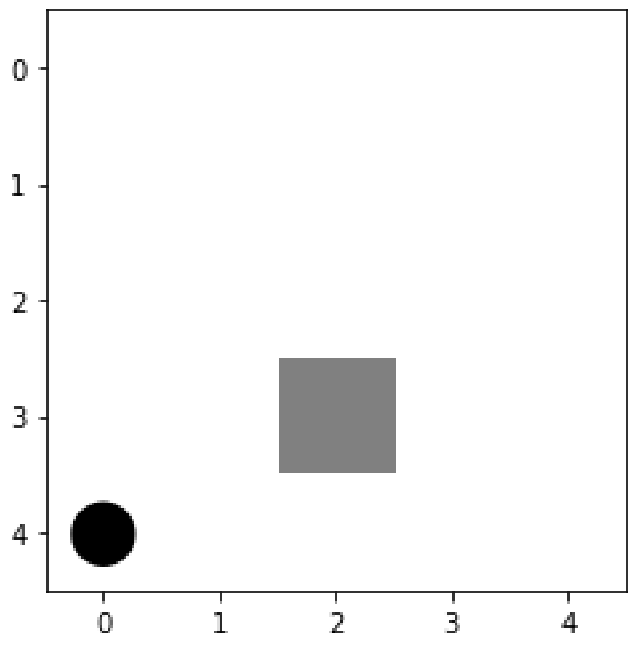
```
-----
step: 74
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



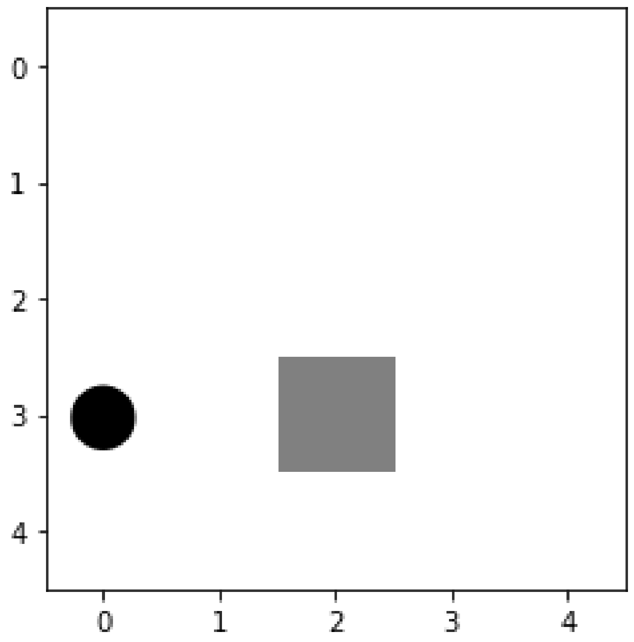
```
-----
step: 75
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



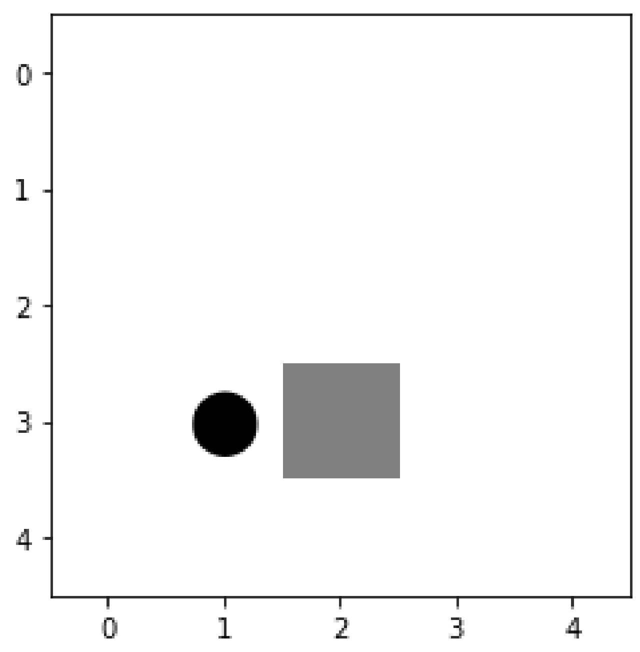
```
-----
step: 76
current position [4, 1]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action west
```



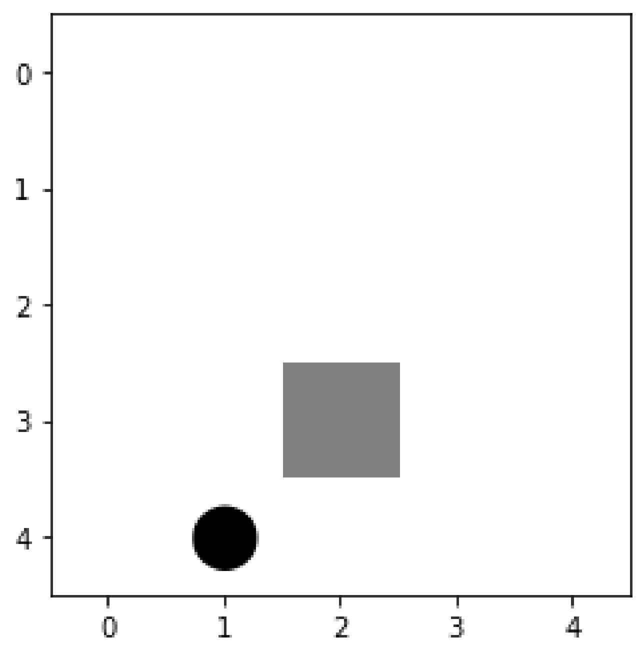
```
-----
step: 77
current position [4, 0]
bumper {'north': False, 'south': True, 'west': True, 'east': False}
dirty False
action north
```



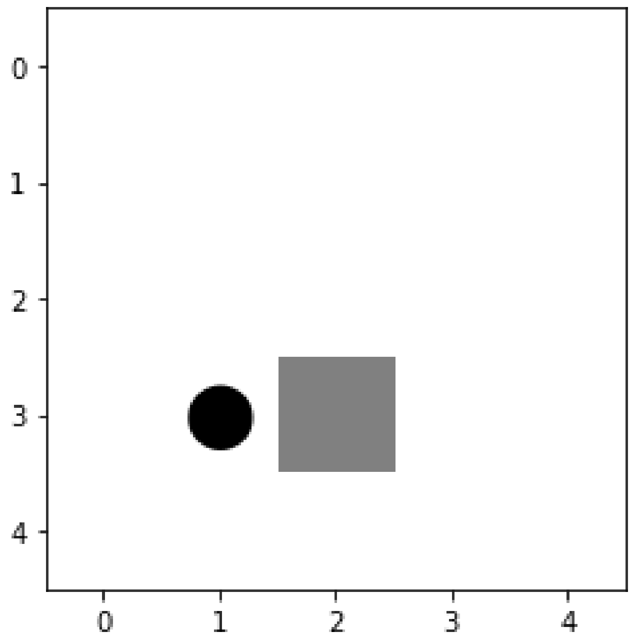
```
-----
step: 78
current position [3, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



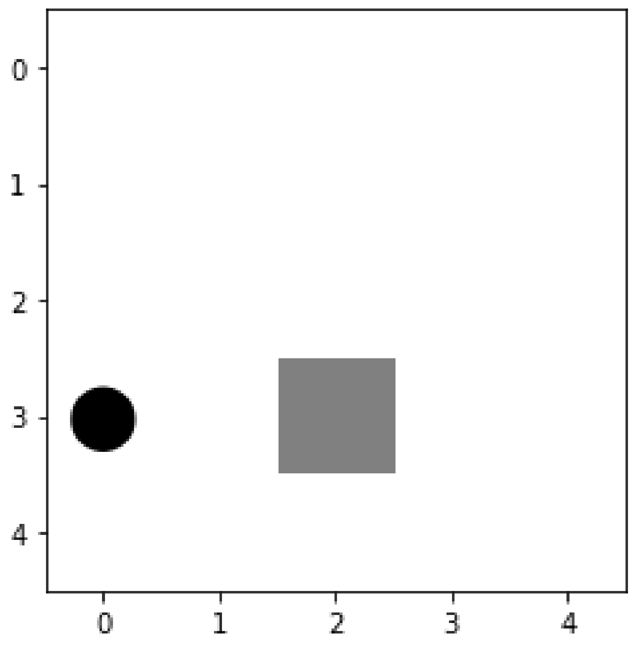
```
-----
step: 79
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action south
```



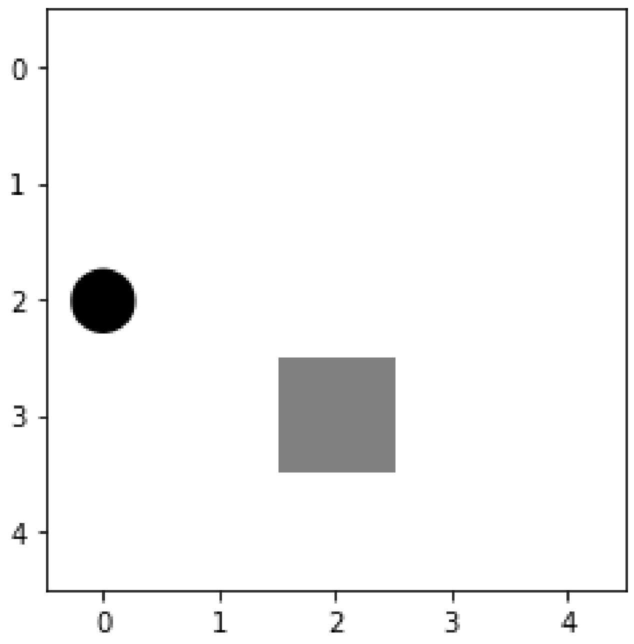
```
-----
step: 80
current position [4, 1]
bumper {'north': False, 'south': True, 'west': False, 'east': False}
dirty False
action north
```



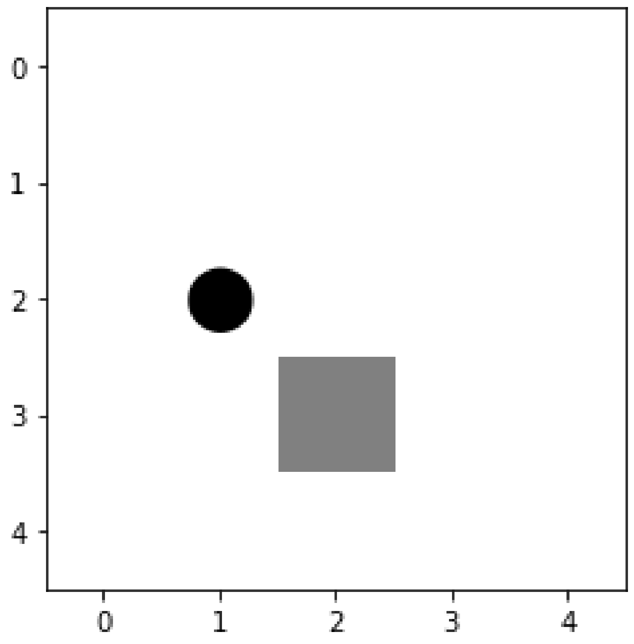
```
-----
step: 81
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



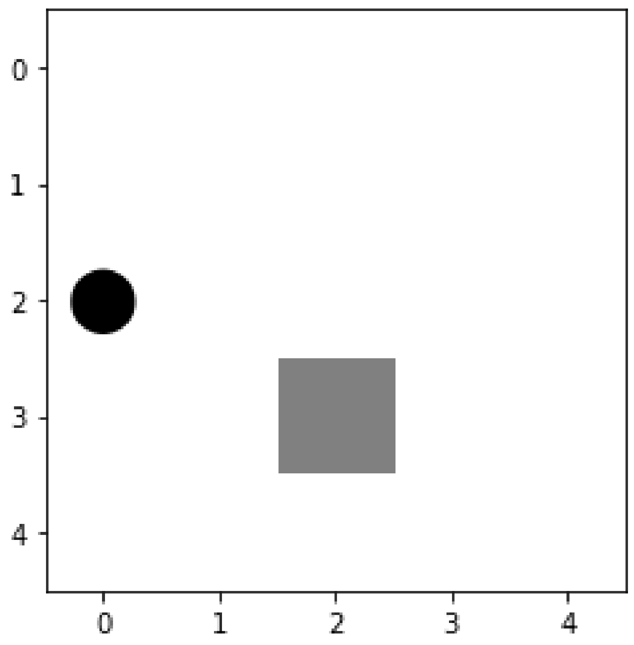
```
-----
step: 82
current position [3, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action north
```



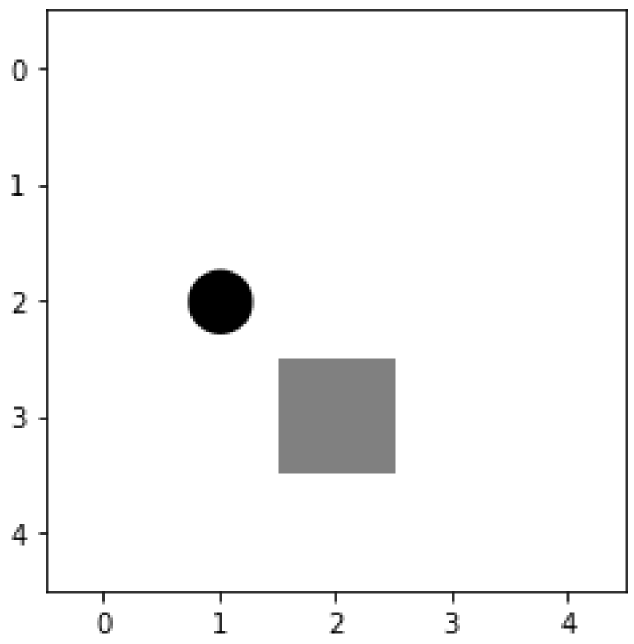
```
-----
step: 83
current position [2, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



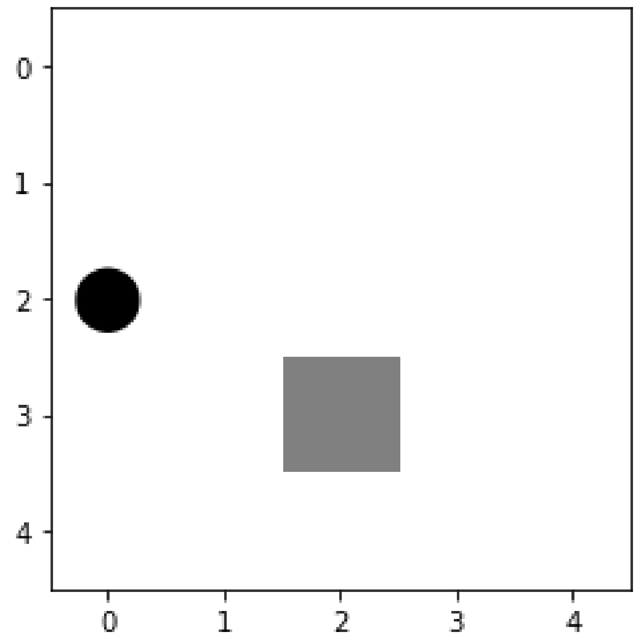
```
-----
step: 84
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



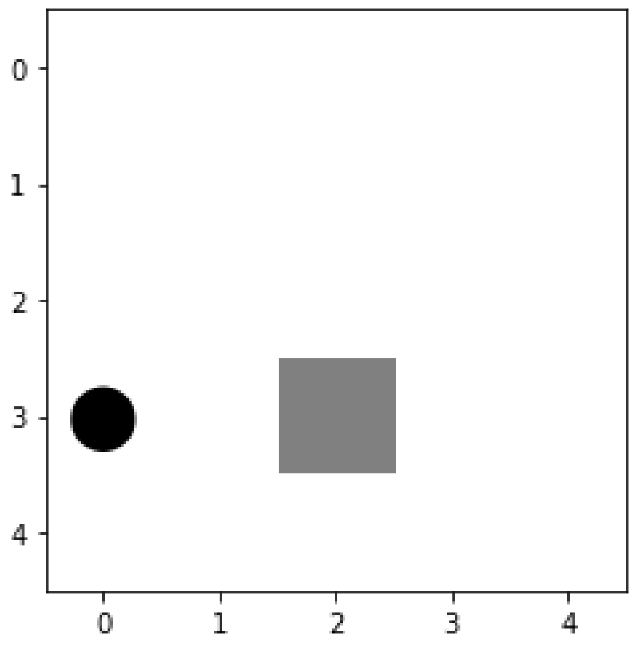
```
-----
step: 85
current position [2, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



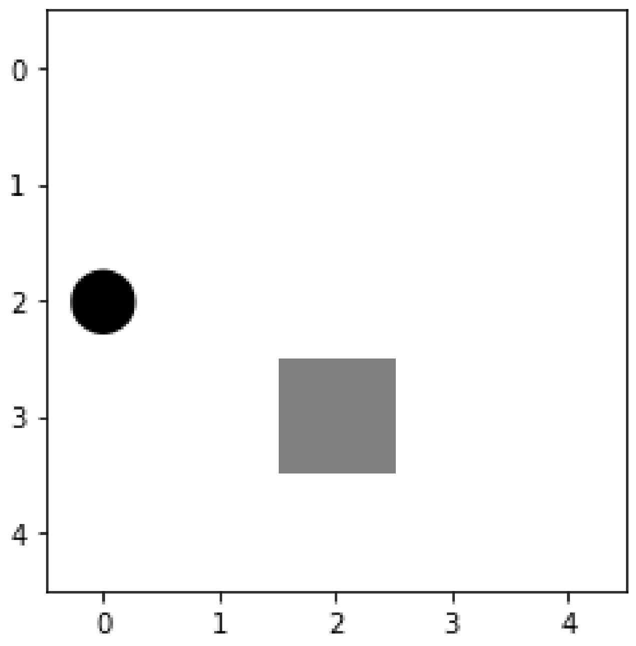
```
-----
step: 86
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



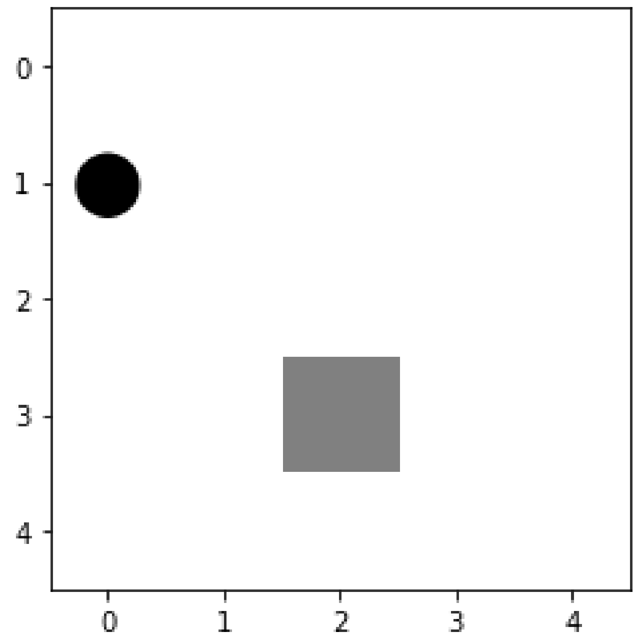
```
-----
step: 87
current position [2, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action south
```



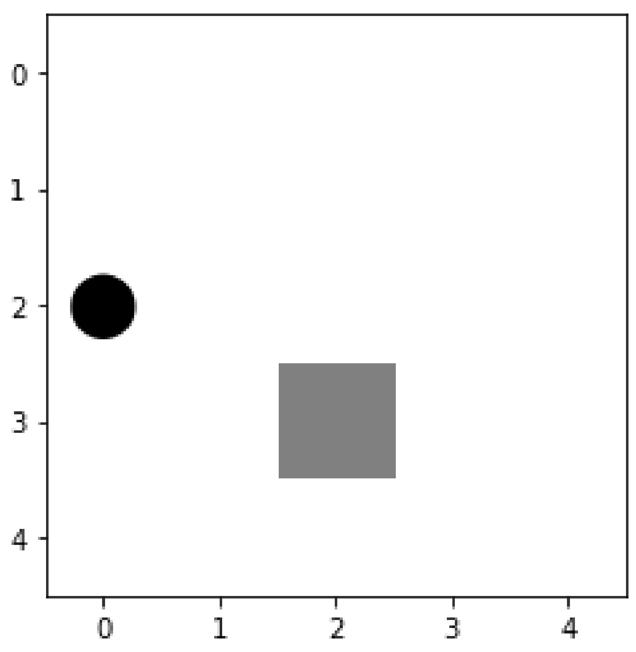
```
-----
step: 88
current position [3, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action north
```



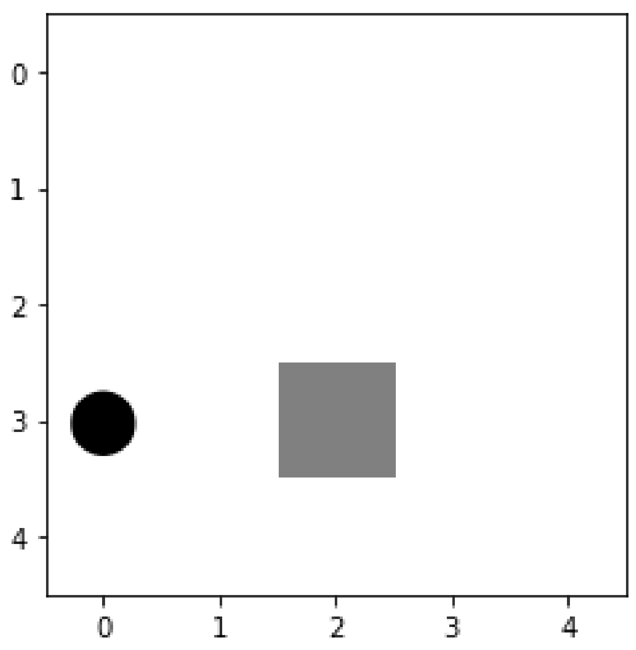
```
-----
step: 89
current position [2, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action north
```



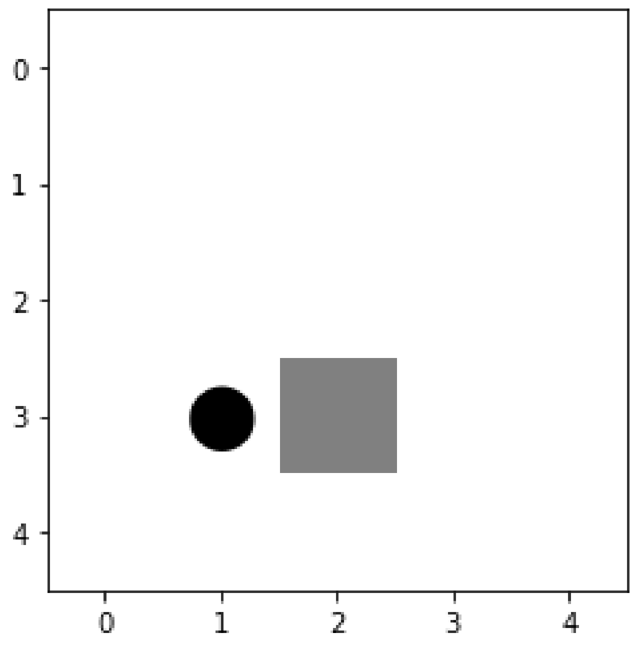
```
-----
step: 90
current position [1, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action south
```



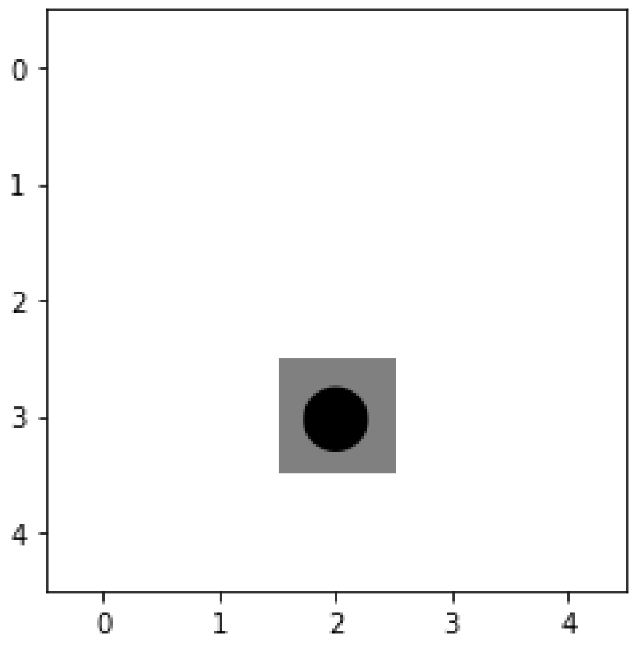
```
-----
step: 91
current position [2, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action south
```



```
-----
step: 92
current position [3, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



```
-----
step: 93
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



```
-----
step: 94
current position [3, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty True
action suck
```

[8]:
```
[2, 94]
```

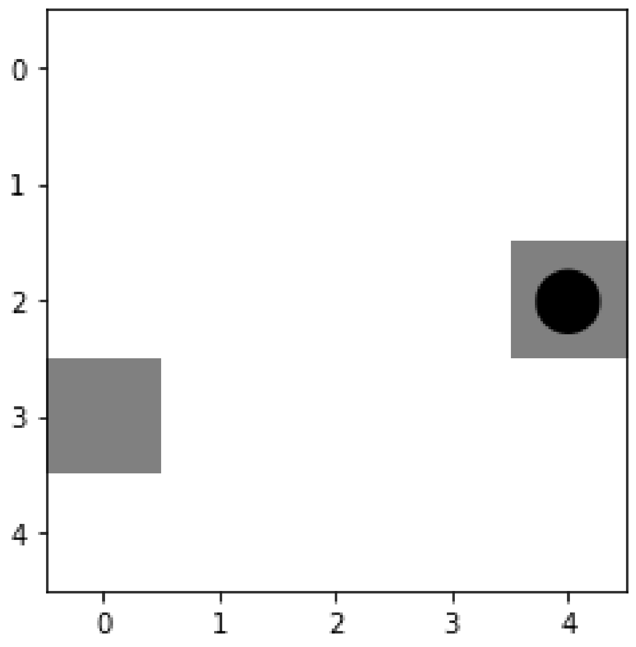
4 Task 3: Implement a model-based reflex agent [3 Point]

This agent keeps track of the location and remembers where it has cleaned. Assume the agent knows how many squares the room has. It can move to a corner to determine its location and then is able to use more advanced navigation.

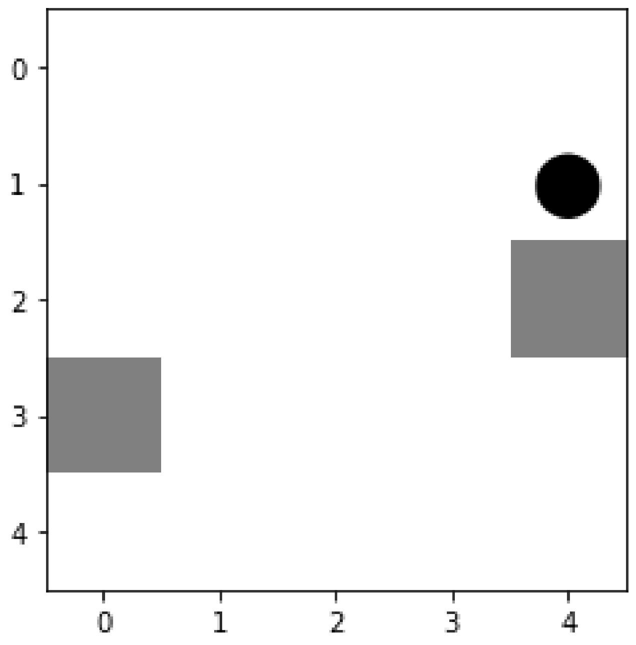
Describe how you define the **agent state** and how your agent works before implementing it. *Note on implementing the state in Python:* Examples

[9]:
```python
class model_based_reflex_agent:

    def __init__(self, name = "An Agent"):

        self.name = name

        self.task = True

        self.clearmemory = []

        self.direction = "east"


    def actions(self, bumpers, dirty):

        if self.task :

            if not bumpers["north"] :


                return "north"

            if not bumpers["west"] :


                return "west"



            self.task = False

            self.position = [0,0]


            return actions


        else :

            if dirty :


                self.clearmemory.append(self.position)


                return "suck"

            else :


                # Checking for bumper.
                if not bumpers[self.direction] :


                    if self.direction == "east":
                        self.position[1] = self.position[1] + 1


                    else :

                        self.position[1] = self.position[1] - 1

                    return self.direction


                else :

                    self.position[0] = self.position[0] + 1

                    if self.direction == "east":

                        self.direction = "west"

                    else:

                        self.direction = "east"

                    return "south"


mreflex = model_based_reflex_agent()
vacuum_environment(mreflex.actions, n = 5, maxsteps = 10000, verbose = True)
```

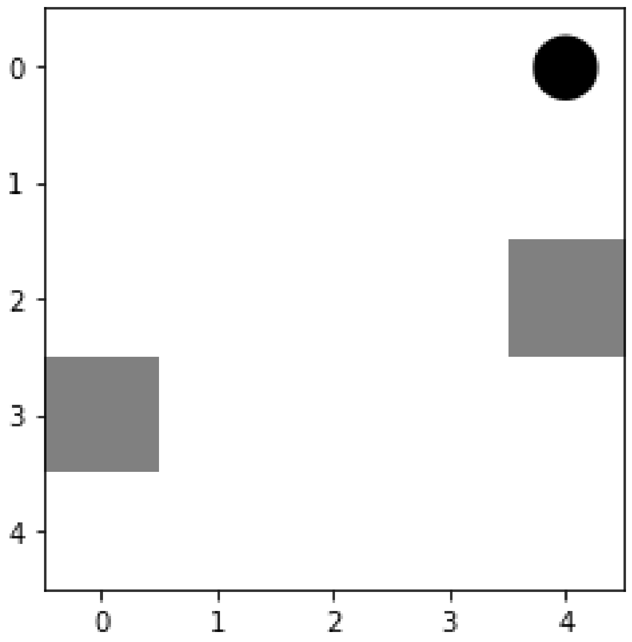
```
room:
 [[False False False False False]
 [False False False False False]
 [False False False False  True]
 [ True False False False False]
 [False False False False False]]
dirty squares: 2

start simulation
```



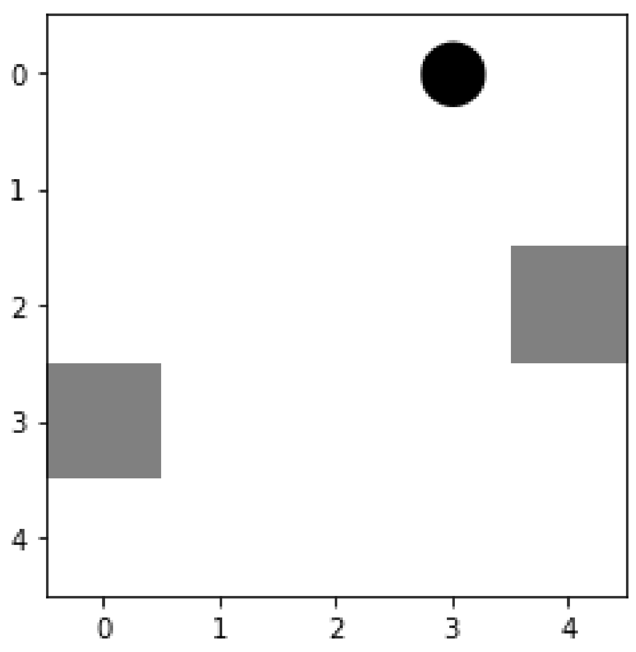
```
-----
step: 1
current position [2, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty True
action north
```



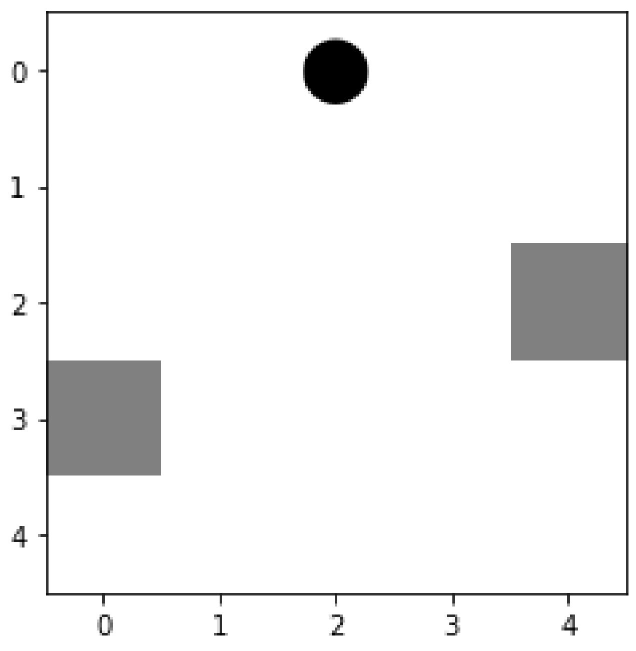
```
-----
step: 2
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action north
```



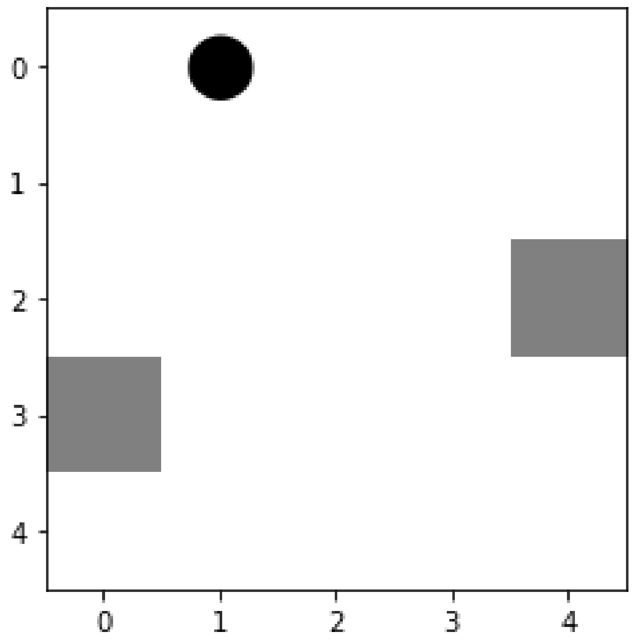
```
-----
step: 3
current position [0, 4]
bumper {'north': True, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



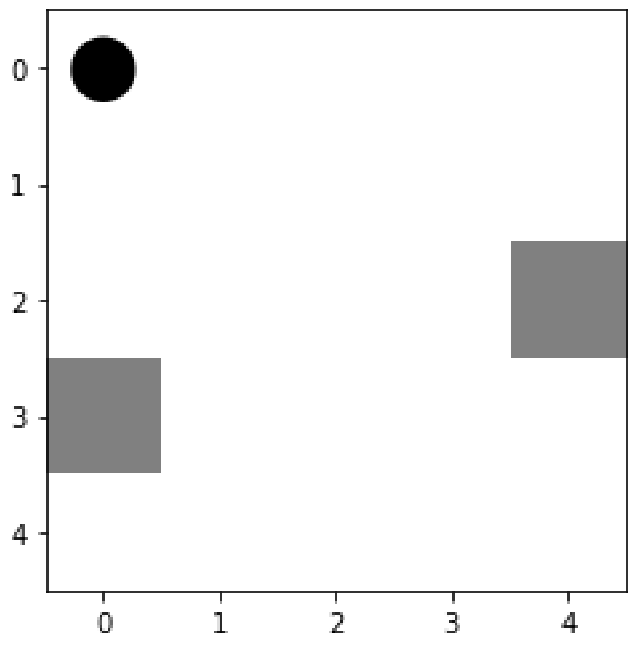
```
-----
step: 4
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



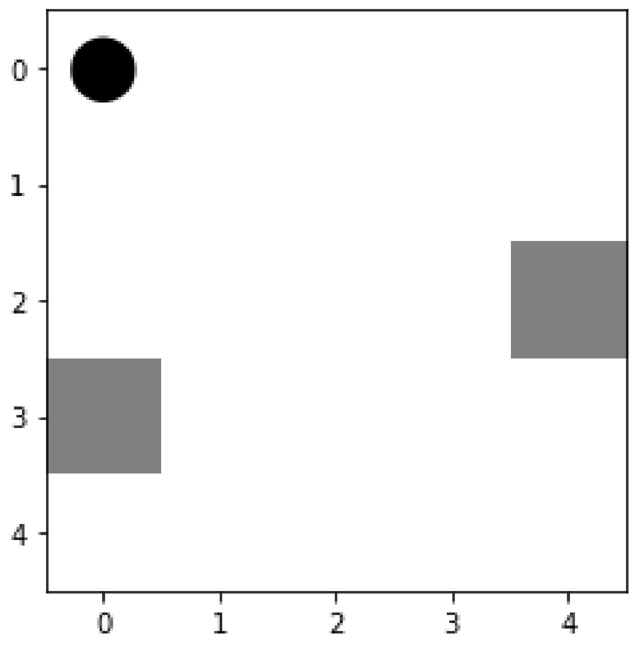
```
-----
step: 5
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



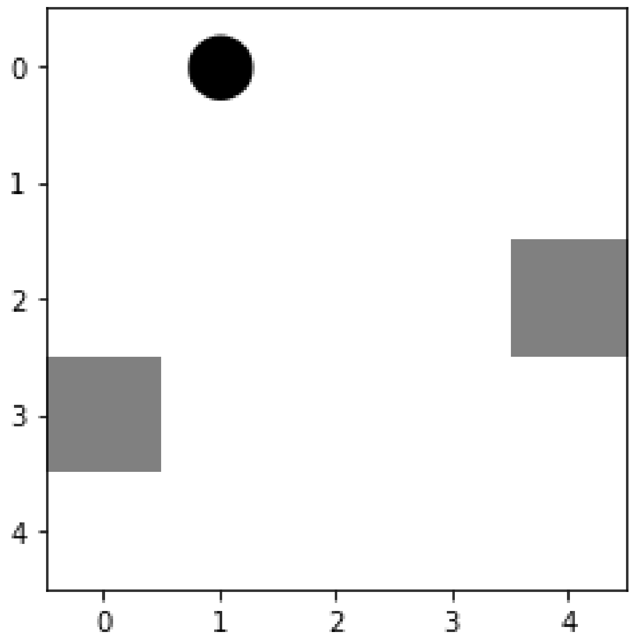
```
-----
step: 6
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



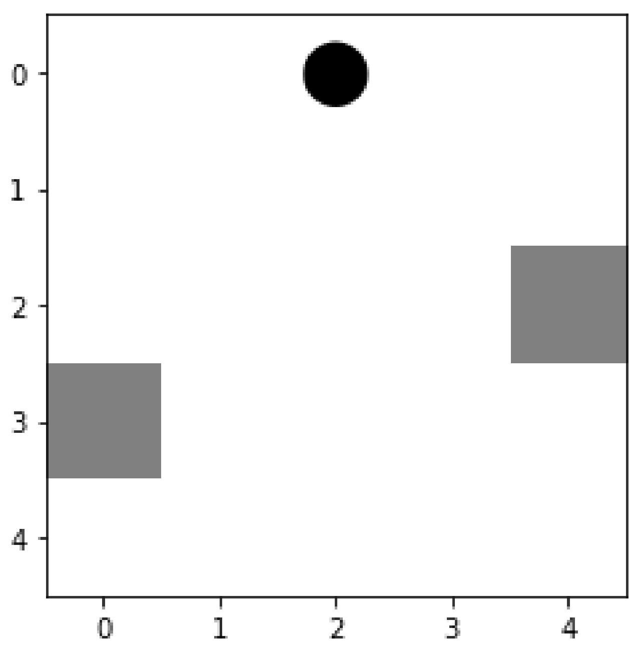
```
-----
step: 7
current position [0, 0]
bumper {'north': True, 'south': False, 'west': True, 'east': False}
dirty False
action ['north', 'east', 'west', 'south', 'suck']
```



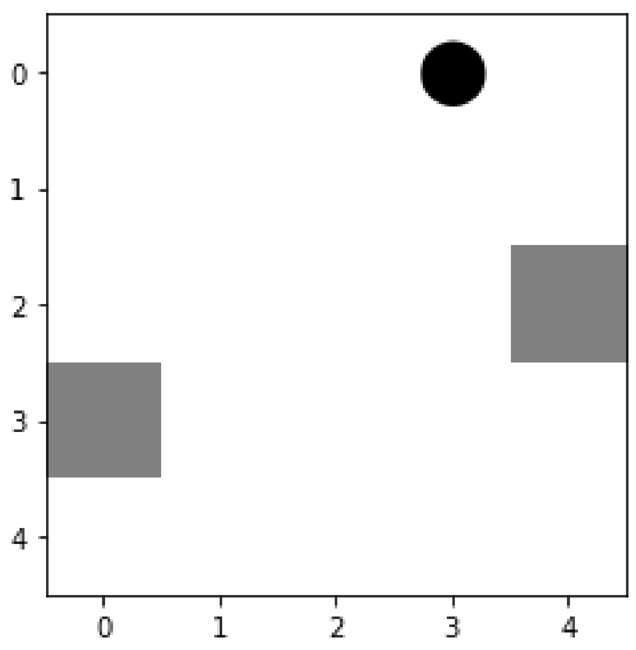
```
-----
step: 8
current position [0, 0]
bumper {'north': True, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



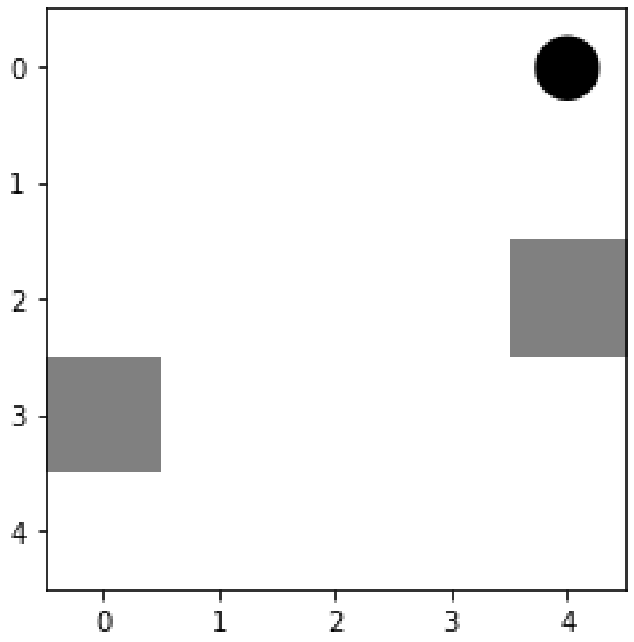
```
-----
step: 9
current position [0, 1]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



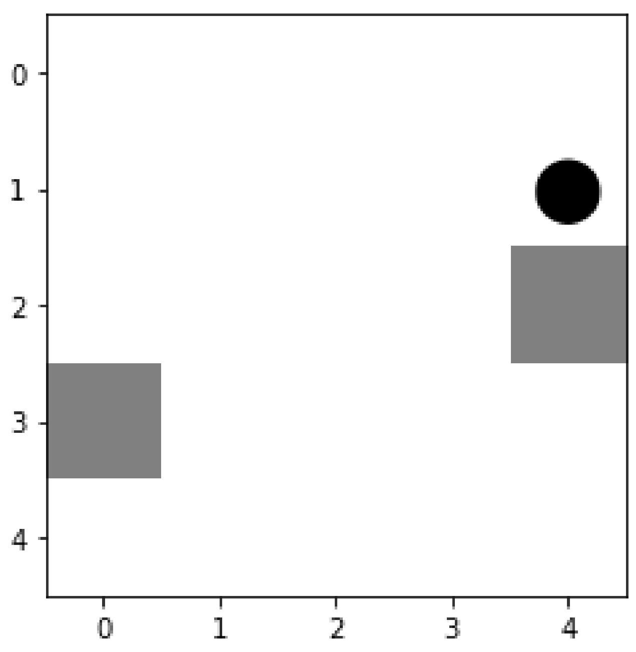
```
-----
step: 10
current position [0, 2]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



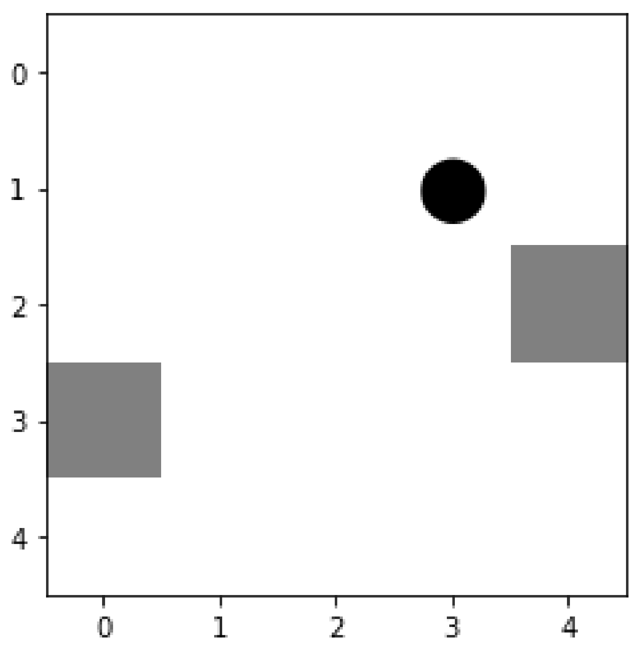
```
-----
step: 11
current position [0, 3]
bumper {'north': True, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



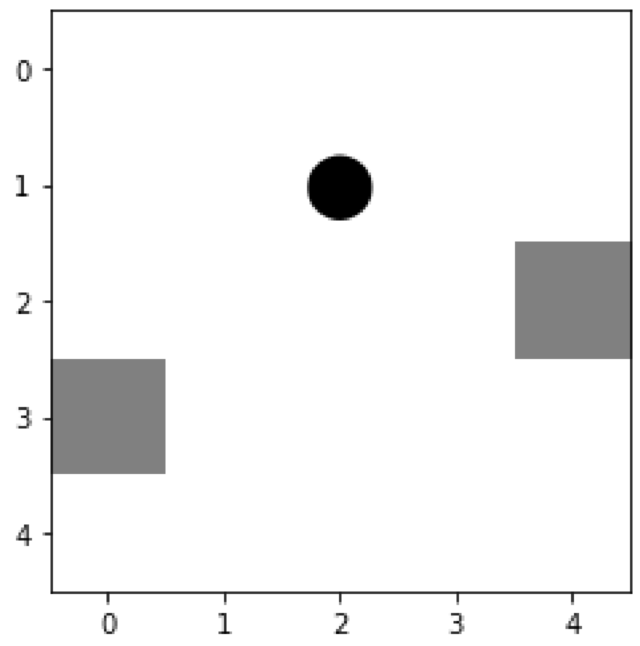
```
-----
step: 12
current position [0, 4]
bumper {'north': True, 'south': False, 'west': False, 'east': True}
dirty False
action south
```



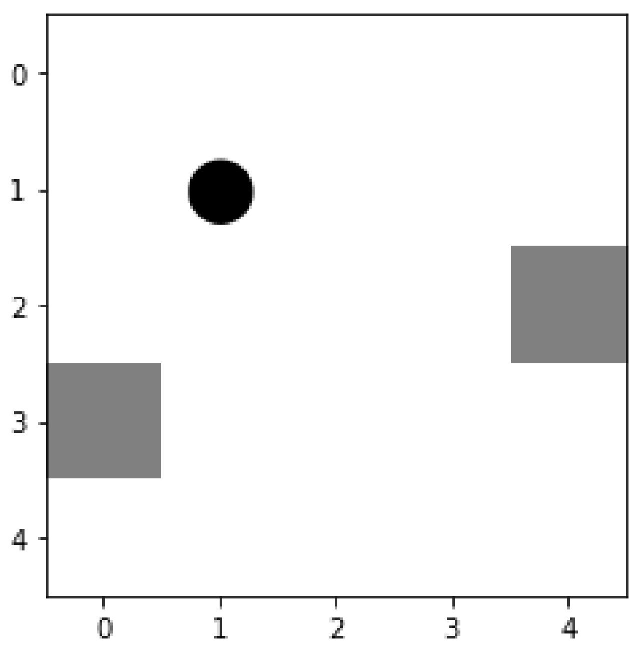
```
-----
step: 13
current position [1, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



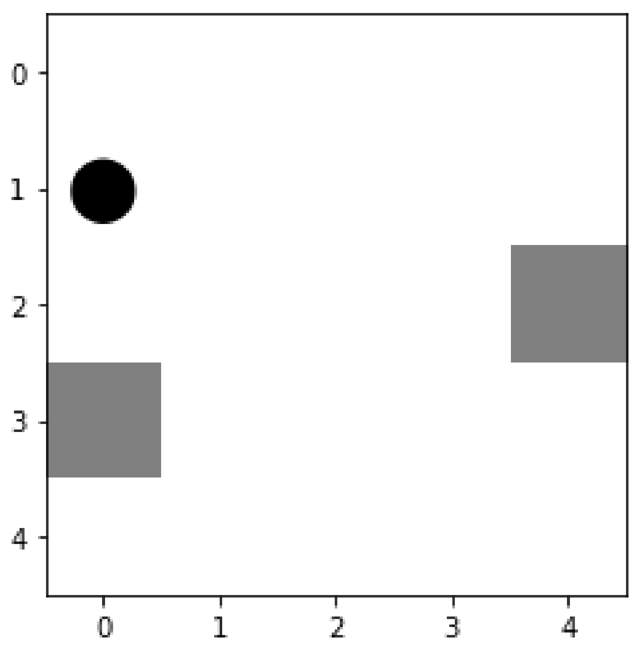
```
-----
step: 14
current position [1, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



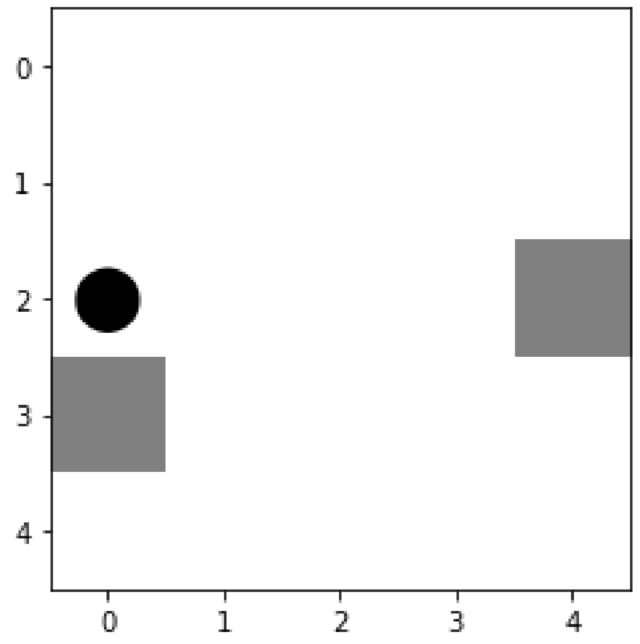
```
-----
step: 15
current position [1, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



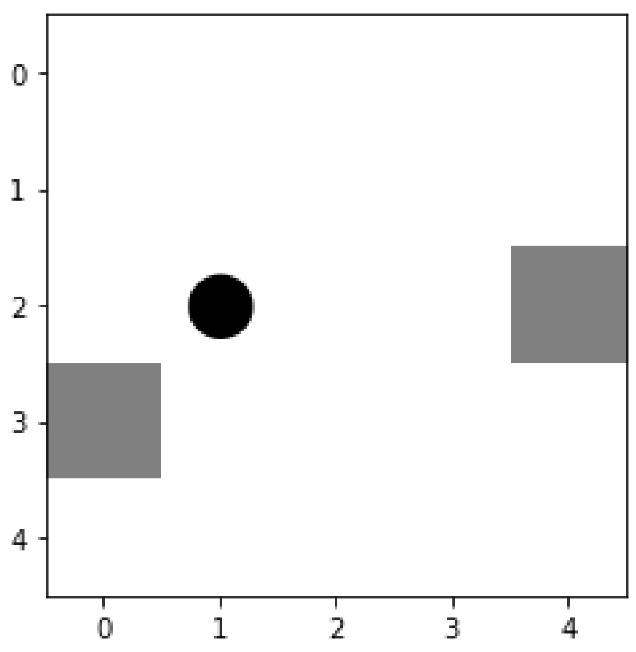
```
-----
step: 16
current position [1, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



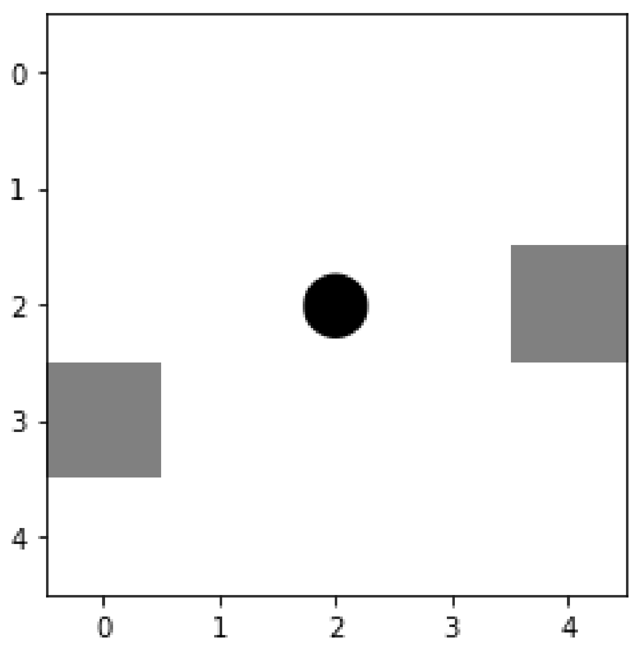
```
-----
step: 17
current position [1, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action south
```



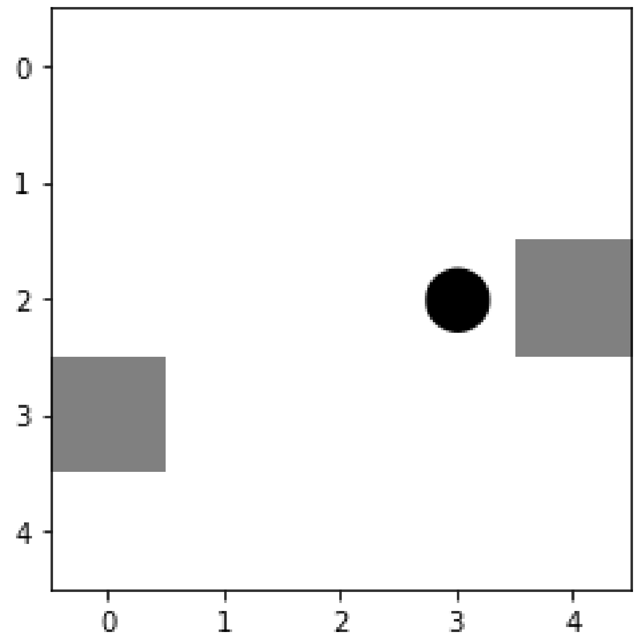
```
-----
step: 18
current position [2, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty False
action east
```



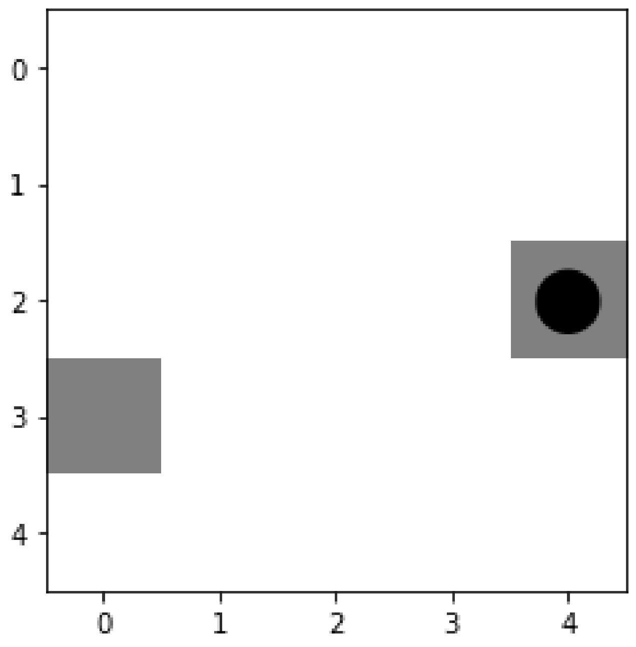
```
-----
step: 19
current position [2, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



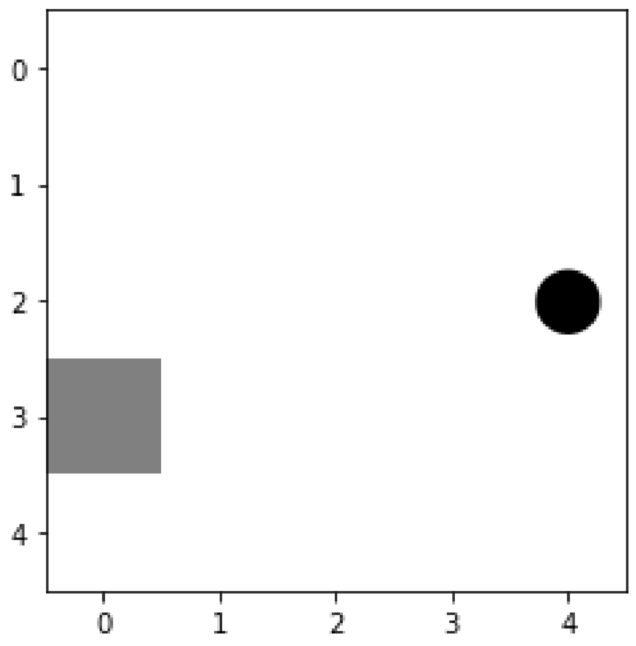
```
-----
step: 20
current position [2, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



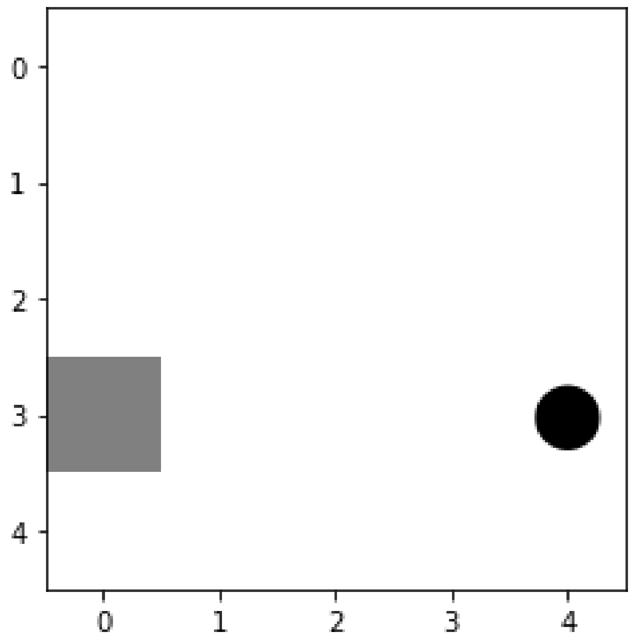
```
-----
step: 21
current position [2, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action east
```



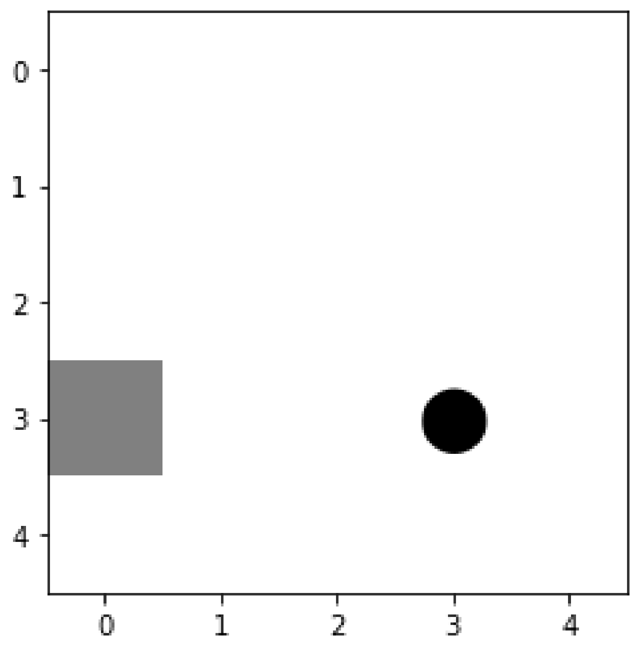
```
-----
step: 22
current position [2, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty True
action suck
```



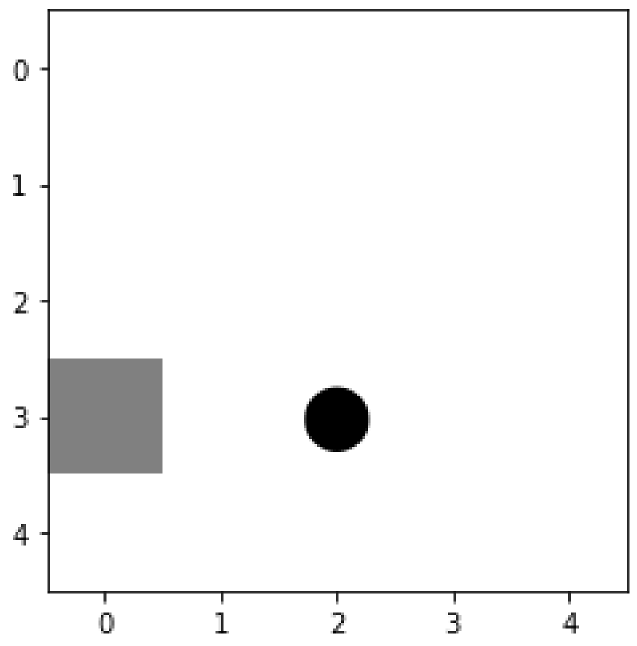
```
-----
step: 23
current position [2, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action south
```



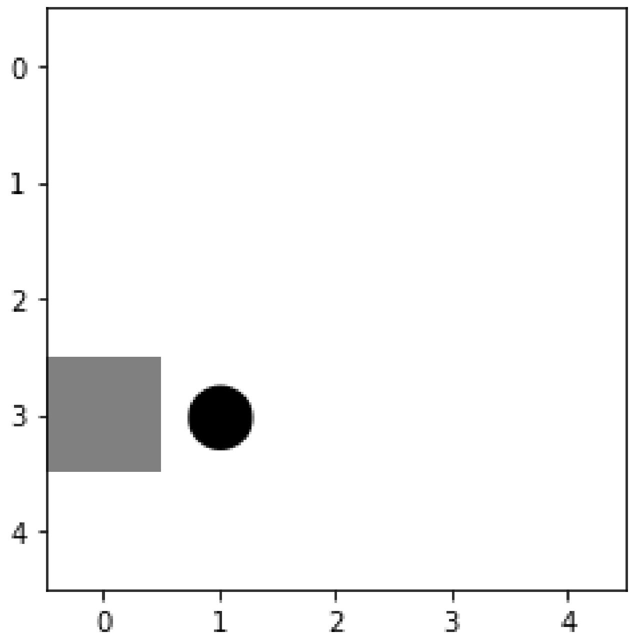
```
-----
step: 24
current position [3, 4]
bumper {'north': False, 'south': False, 'west': False, 'east': True}
dirty False
action west
```



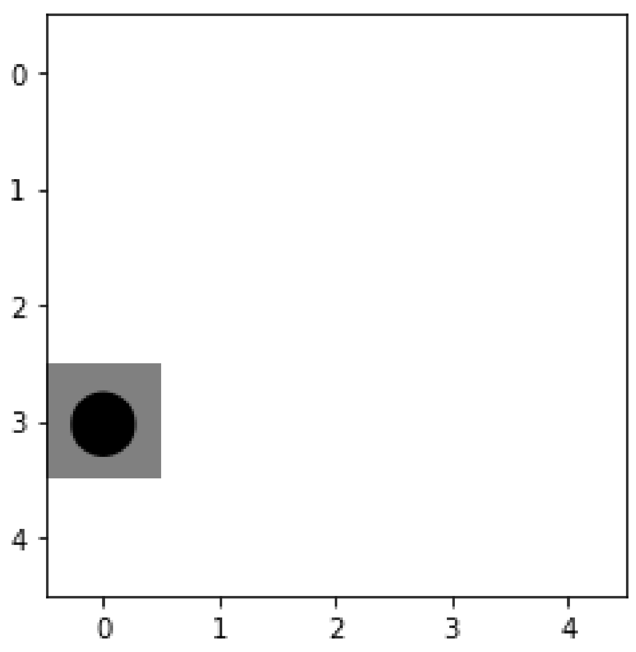
```
-----
step: 25
current position [3, 3]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



```
-----
step: 26
current position [3, 2]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



```
-----
step: 27
current position [3, 1]
bumper {'north': False, 'south': False, 'west': False, 'east': False}
dirty False
action west
```



```
-----
step: 28
current position [3, 0]
bumper {'north': False, 'south': False, 'west': True, 'east': False}
dirty True
action suck
```

[9]:
```
[2, 28]
```

4.1 Task 4: Simulation study [3 Points]

Compare the performance (the performance measure is defined in the PEAS description above) of the agents using environments of different size. E.g., 5×5, 10×10 and 100×100. Use 100 random runs for each. Present the results in a suitable format (tables, graphs) and discuss the differences.

Here is some help with charts and tables.

[10]:
```python
# Your code goes here
import matplotlib.pyplot as plt

# add a grid to the plots
import seaborn as sns
sns.set(style="whitegrid")
from pylab import rcParams
rcParams['figure.figsize']=10,5
```

[ ]:
```

```

```
[11]: N = 100
agent1_5 = np.repeat(0,N)
agent2_5 = np.repeat(0,N)
agent3_5 = np.repeat(0,N)
agent1_10 = np.repeat(0,N)
agent2_10 = np.repeat(0,N)
agent3_10 = np.repeat(0,N)
agent1_100= np.repeat(0,N)
agent2_100= np.repeat(0,N)
agent3_100= np.repeat(0,N)


# Calculating the energy expended to clean a dirty block in each run for 5x5␣
 ↪room for i in range(N):
for i in range(N):

    mreflex = model_based_reflex_agent()
    # a15 contains [dirty, energy expended] pair
    agent1_5[i]= vacuum_environment(simple_randomized_agent, n = 5, maxsteps =␣
 ↪1000, verbose = False)[1]



    agent2_5[i]= vacuum_environment(simple_reflex_agent, n = 5, maxsteps =␣
 ↪1000, verbose = False)[1]


    agent3_5[i]= vacuum_environment(mreflex.actions, n = 5, maxsteps = 1000,␣
 ↪verbose = False)[1]



# Calculating the energy expended to clean a dirty block in each run for 10x10␣
 ↪room for i in range(N):
for i in range(N):

    mreflex = model_based_reflex_agent()

    agent1_10[i]= vacuum_environment(simple_randomized_agent, n = 10, maxsteps␣
 ↪= 1000, verbose = False)[1]



    agent2_10[i]= vacuum_environment(simple_reflex_agent, n = 10, maxsteps =␣
 ↪1000, verbose = False)[1]
```

```
    agent3_10[i]= vacuum_environment(mreflex.actions, n = 10, maxsteps = 1000,␣
 ↪verbose = False)[1]


    # Calculating the energy expended to clean a dirty block in each run for␣
 ↪100x100 room for i in range(N):

for i in range(N):

    mreflex = model_based_reflex_agent()

    agent1_100[i]= vacuum_environment(simple_randomized_agent, n = 100,␣
 ↪maxsteps = 100000, verbose = False)[1]



    agent2_100[i]= vacuum_environment(simple_reflex_agent, n = 100, maxsteps =␣
 ↪100000, verbose = False)[1]


    agent3_100[i]= vacuum_environment(mreflex.actions, n = 100, maxsteps =␣
 ↪100000, verbose = False)[1]
```

Fill out the following table with the average performance measure for 100 random runs (you may also create this table with code):

Size	Randomized Agent	Simple Reflex Agent	Model-based Reflex Agent
5x5			
10x10			
100x100			

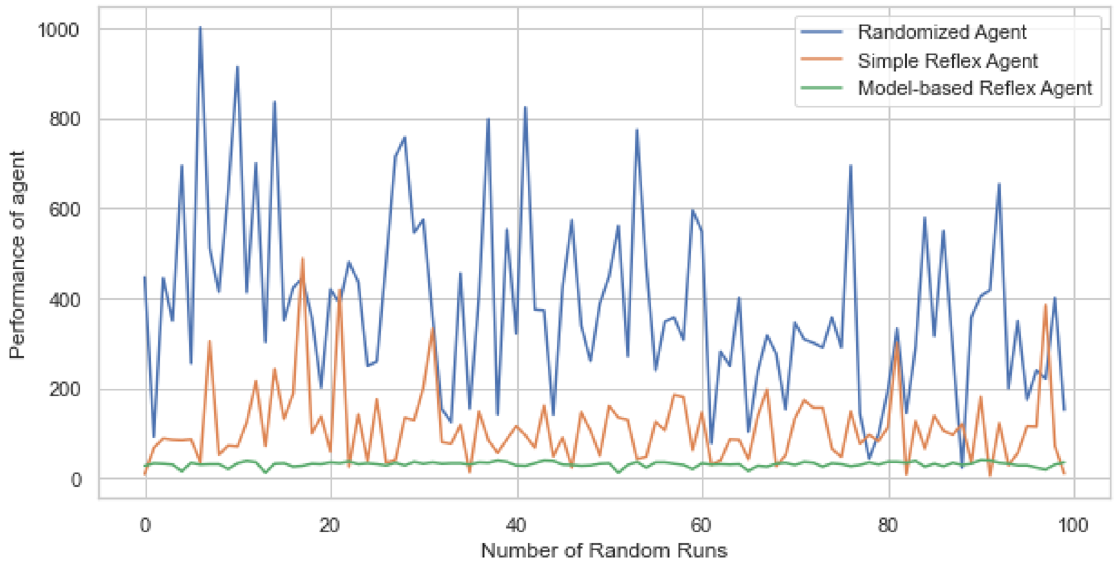
```
[12]: #Below table will display the average of the performance of the  3 agents for␣
 ↪diffrent room size for 100 random runs
import numpy as np

import pandas as pd

data = np.array([['5x5', np.average(agent1_5), np.average(agent2_5), np.
 ↪average(agent3_5)], ['10x10', np.average(agent1_10) ,np.average(agent2_10),␣
 ↪np.average(agent3_10)], ['100x100', np.average(agent1_100), np.
 ↪average(agent2_100), np.average(agent3_100)]])



df = pd.DataFrame(data, columns=["Size", "Randomized Agent", "Simple Reflex␣
 ↪Agent", "Model-based Reflex Agent"])
```

```
df
```

```
[12]:      Size  Randomized Agent  Simple Reflex Agent  Model-based Reflex Agent
      0     5x5            383.74               112.64                     30.68
      1   10x10            1000.0               801.29                    123.02
      2 100x100          100000.0             100000.0                  12090.58
```

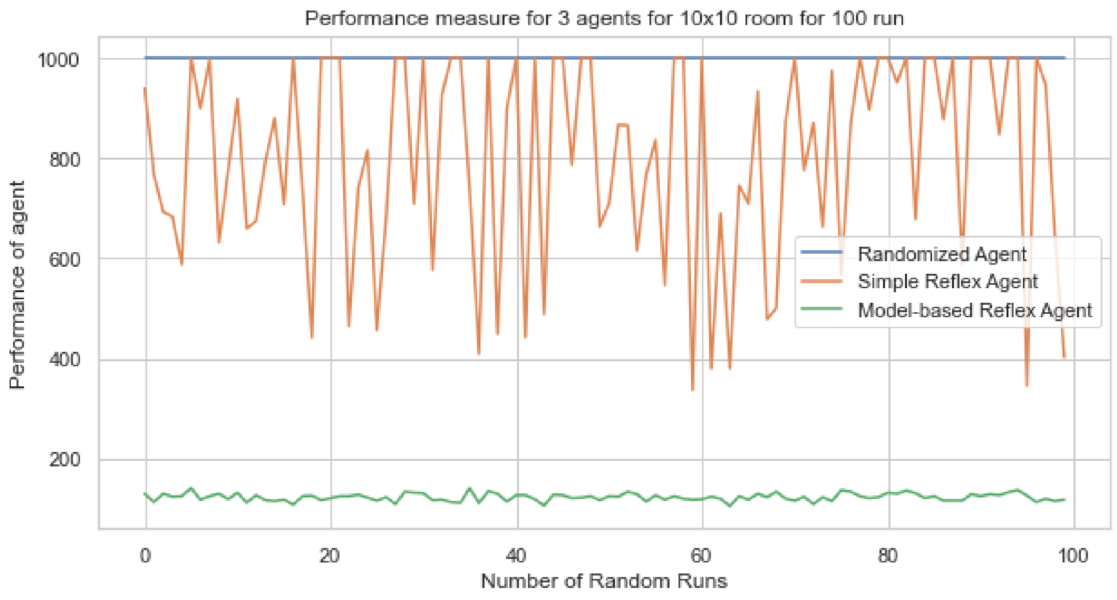
From the above table we can see that model based reflex agent has the least energy expended since it is more intelligent agent than randomized and simple reflex agent because it stores the memory when it cleans the area from dirty to clean whereas other two agent does not. That is why the energy expended by model based reflex agent is less than other two agent

```
[13]: rcParams['figure.figsize'] = 10, 5

x = range(N)

plt.plot(x, agent1_5, label = "Randomized Agent")

plt.plot(x, agent2_5, label = "Simple Reflex Agent")

plt.plot(x, agent3_5, label = "Model-based Reflex Agent")

plt.xlabel("Number of Random Runs")

plt.ylabel("Performance of agent")

plt.legend()

plt.show()
```



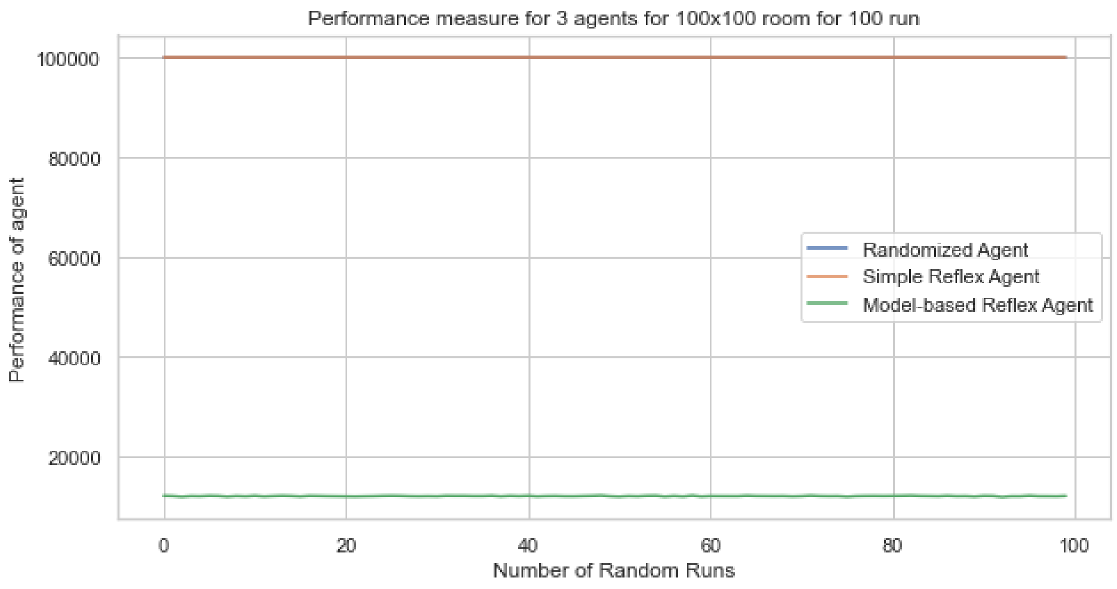
From the above graph we can see that green coloured value which is model based reflex agent requires less cost (i.e performance measure)since it is more intelligent agent.

[14]:
```
rcParams['figure.figsize'] = 10, 5

x = range(N)

plt.plot(x, agent1_10, label = "Randomized Agent")

plt.plot(x, agent2_10, label = "Simple Reflex Agent")

plt.plot(x, agent3_10, label = "Model-based Reflex Agent")

plt.title("Performance measure for 3 agents for 10x10 room for 100 run")

plt.xlabel("Number of Random Runs")

plt.ylabel("Performance of agent")

plt.legend()

plt.show()
```



From the above graph we can see that green coloured value which is model based reflex agent requires less cost (i.e performance measure)since it is more intelligent agent.

[15]:
```python
rcParams['figure.figsize'] = 10, 5

x = range(N)

plt.plot(x, agent1_100, label = "Randomized Agent")

plt.plot(x, agent2_100, label = "Simple Reflex Agent")

plt.plot(x, agent3_100, label = "Model-based Reflex Agent")

plt.title("Performance measure for 3 agents for 100x100 room for 100 run")

plt.xlabel("Number of Random Runs")

plt.ylabel("Performance of agent")

plt.legend()

plt.show()
```



In the 100x100 graph we can see that randomized agent and simple reflex agent will take infinite number of steps compared to model based agent

4.2 Task 5: Robustness of the agent implementations [1 Point]

Describe how your agent implementations will perform

- if it is put into a rectangular room with unknown size,
- if the cleaning area can have an iregular shape (e.g., a hallway connecting two rooms), or

- if the room contains obstacles (i.e., squares that it cannot pass through and trigger the bumper sensors).

5 #Answer goes here

1.If we put our robot in a rectangular room with unkonwn size then the **randomized agent** will go on till it battery dies and we cannot achieve the goal that is clean the whole room as it may also stuck somewhere and may clean the spot for numerous times. In **simple reflex agent** it knows which action to perform when a certain task is done but has no memory so it will also move the whole till the area is clean **Model based agent** has memory it will clean and store the memory that the area is clean and will operate till all the area is cleaned.

2.If the cleaning area can have an irregular shape When the area is known to the machine it can have action where it can move and start cleaning the hallway once the room is done cleaning.So when the **simple reflex and model based reflex** will act upon the reflex where once it hits the wall it can have an action where it turns into the direction of the hallway.For eg suppose the robot os cleaning the living room and then we want it to go to other room when it hits the last spot and start hitting the wall where the agent know to its right is the hallway and left is the the room which is cleaned.

3.If the room contains obstacles Both the agent can work through this if there is an obstacle where it cannot pass through then it can trigger bumper sensor and move into another direction.

5.1 Graduate student advanced task: Obstacles [1 Point]

Undergraduate students: This is a bonus task you can attempt if you like [+1 Bonus point].

1. Change your simulation environment tor run experiments for the following problem: Add random obstacle squares that also trigger the bumper sensor. The agent does not know where the obstacles are. Observe how this changes the performance of the three implementations.
2. Describe what would need to be done to perform better with obstacles. Add code if you can.

Answer: To perform better with obstacles we need to implement function where it defines what kind of obstacles a robot can face while operating and trigger a bumper sensor and call for action what the robot must do to move past these obstacles and start operating like before.

5.2 More advanced tasks to think about

You can think about these:

- **Unknown environment with obstacles:** Implement an agent for an environment where the agent does not know how large the environment is (we assume it is rectangular), where it starts or where the obstacles are. An option would be to always move to the closest unchecked/uncleaned square.
- **Utility-based agent:** Change the environment, so each square has a fixed probability of getting dirty again. We assume the agent has learned this information over time. For the implementation, we give this information to the agent as a 2-dimensional array of probabilities Cleaning one dirty square produces a utility of 1. Implement a utility-based agent that maximizes the expected utility over one full charge which lasts for 10000 time steps. This is very tricky!

[16]:
```
# Your ideas/code
```