VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

**ARTIFICIAL INTELLIGENCE**

**IT159IU**

# REPORT LAB 1

**Instructor:**

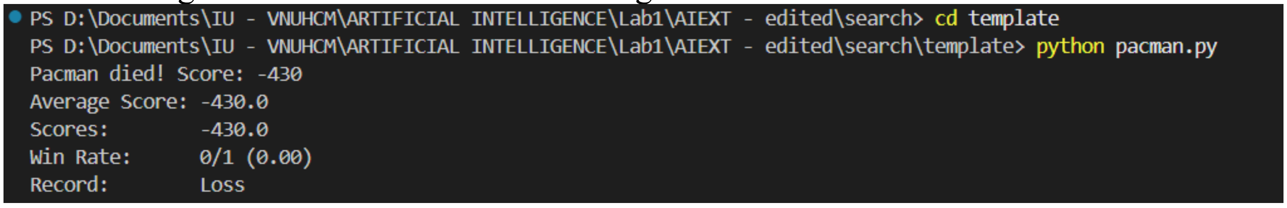
**Dr. Nguyen Trung Ky**

**Dr. Ly Tu Nga**

**Nguyen Huynh Ngan Anh - ITDSIU23003**

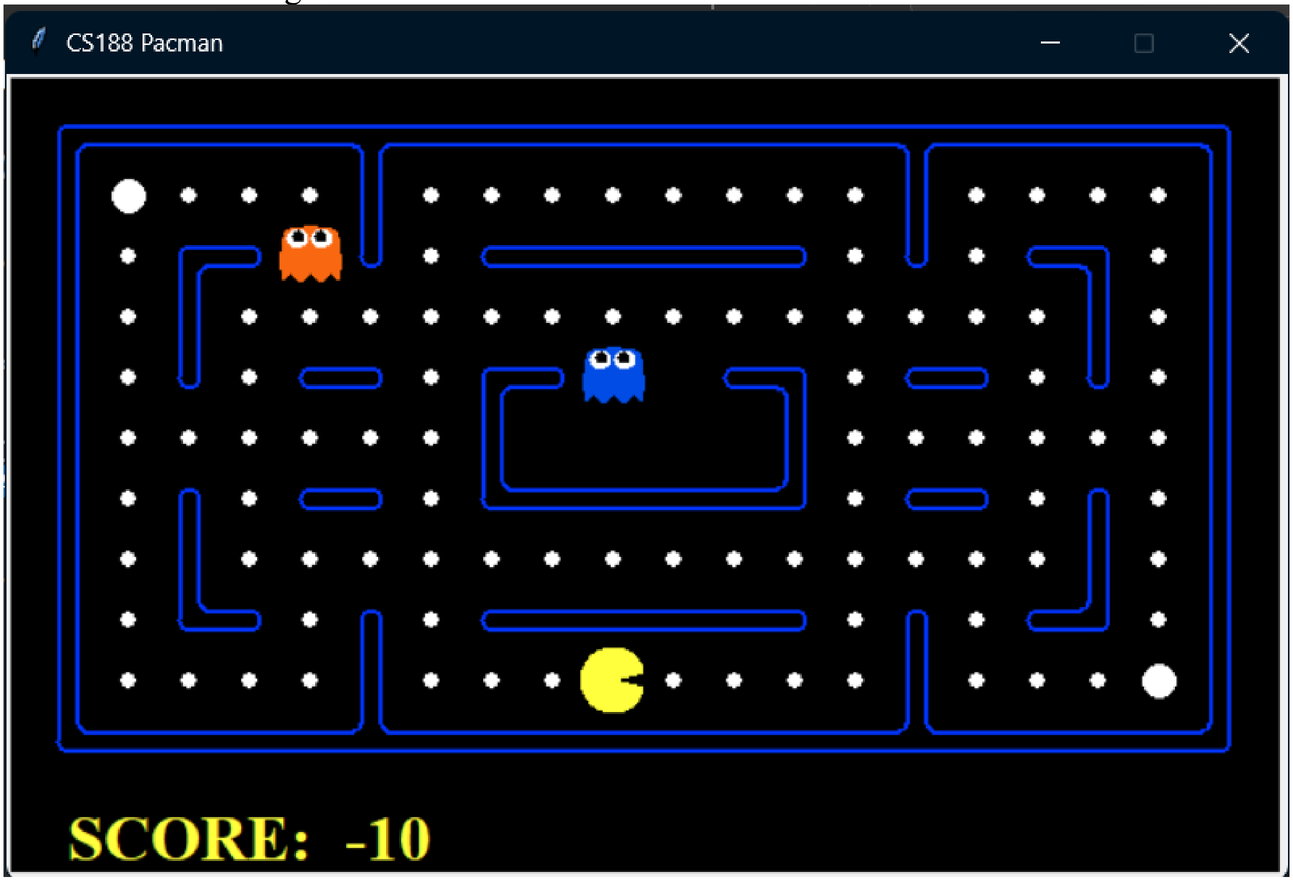
## 0. Setup:

Tried running the below command to run the game



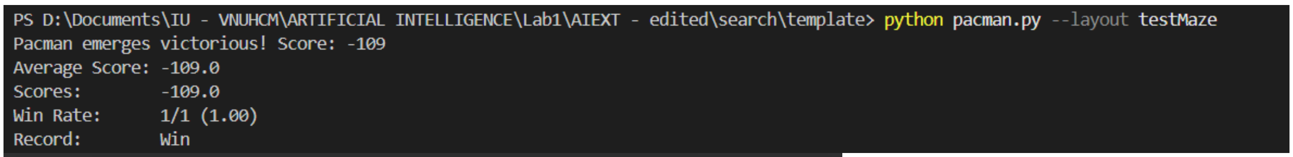
```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search> cd template
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py
Pacman died! Score: -430
Average Score: -430.0
Scores:        -430.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

The interface of the game

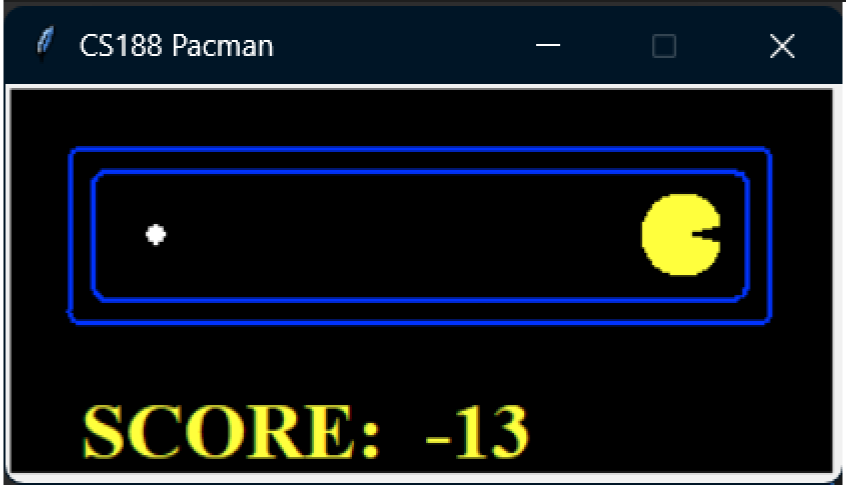


Tried running different layout of the maze to view their interface:

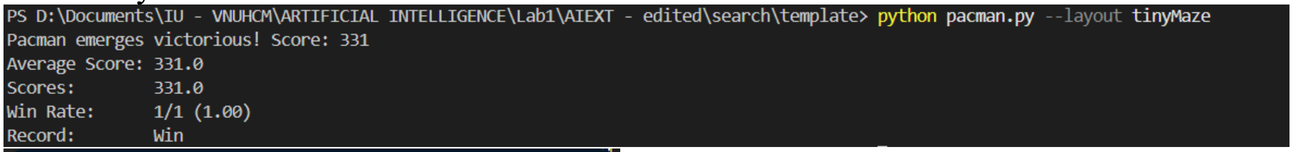
- testMaze:



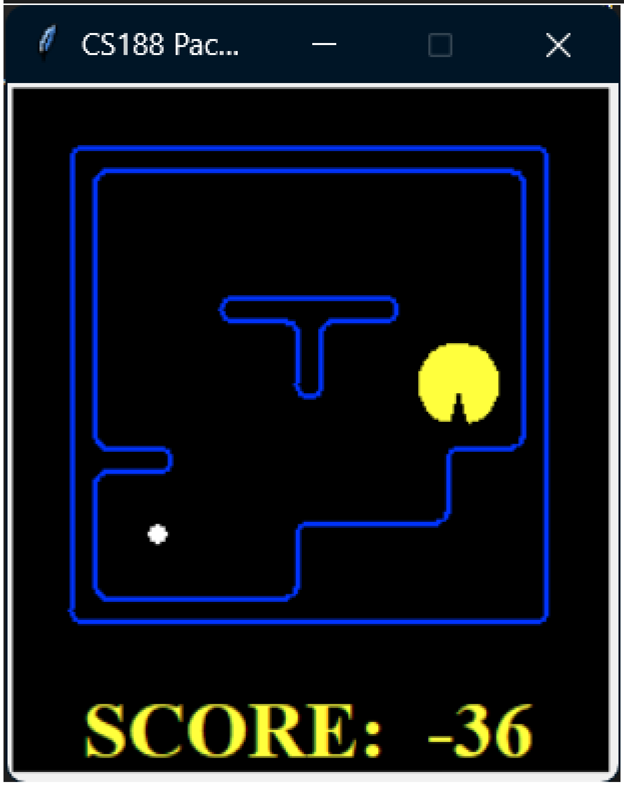
```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py --layout testMaze
Pacman emerges victorious! Score: -109
Average Score: -109.0
Scores:        -109.0
Win Rate:      1/1 (1.00)
Record:        Win
```



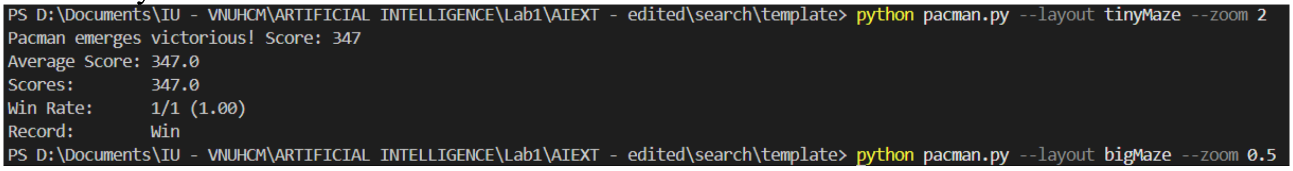
- tinyMaze:



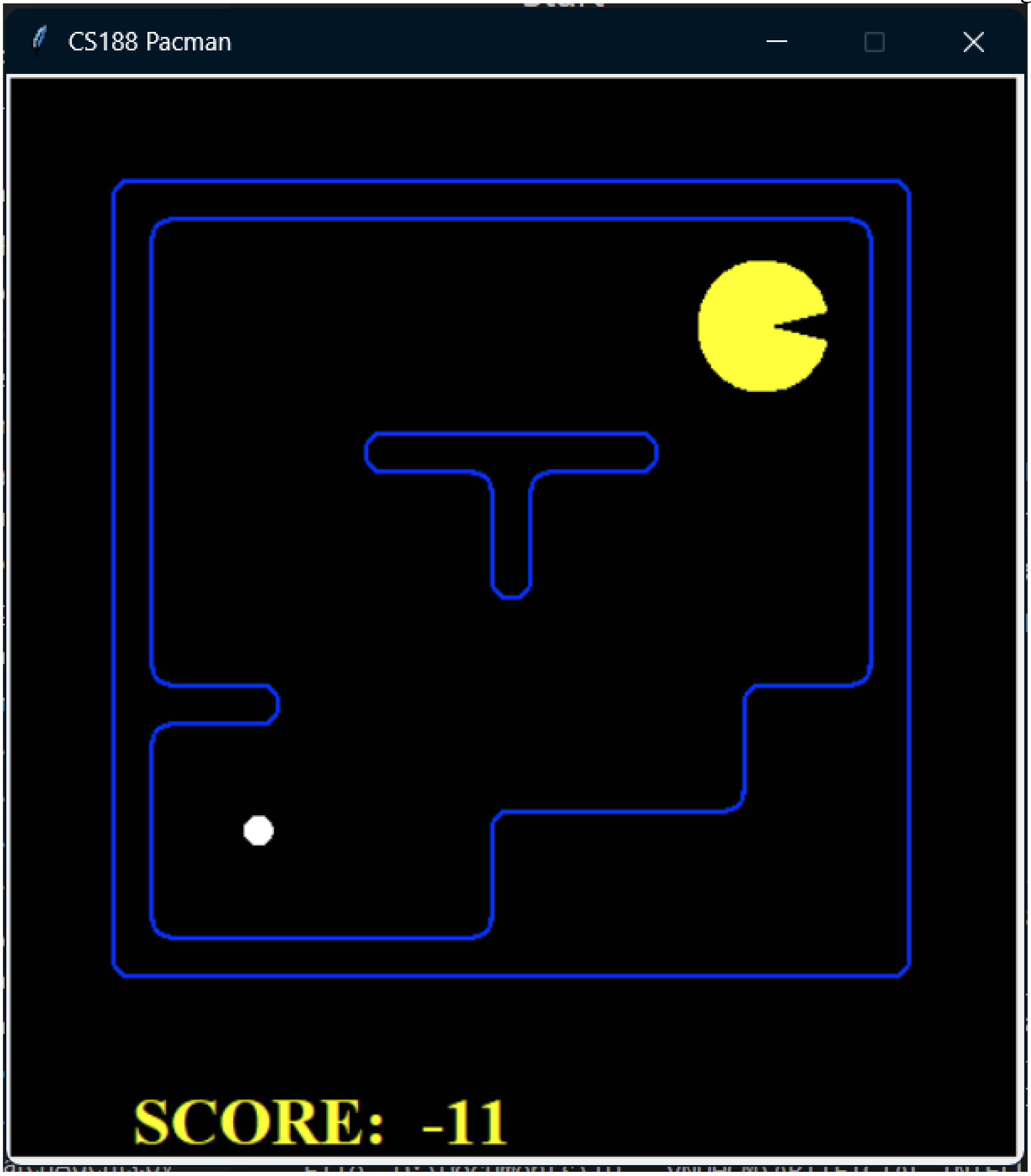
```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py --layout tinyMaze
Pacman emerges victorious! Score: 331
Average Score: 331.0
Scores:        331.0
Win Rate:      1/1 (1.00)
Record:        Win
```



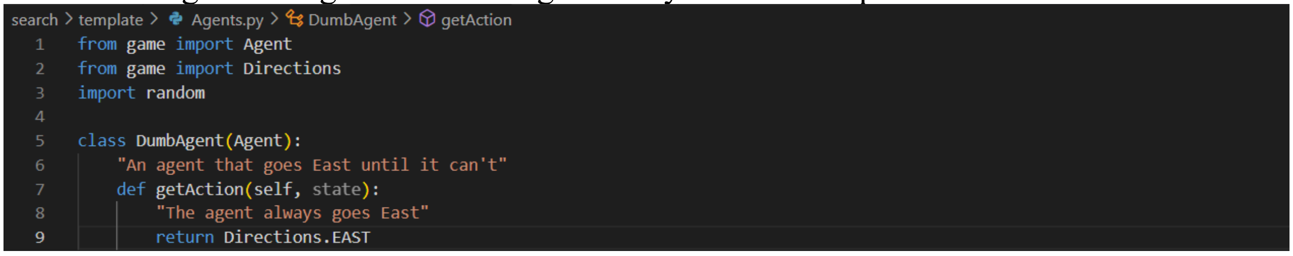
- tinyMaze that zoom x2:



```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py --layout tinyMaze --zoom 2
Pacman emerges victorious! Score: 347
Average Score: 347.0
Scores:        347.0
Win Rate:      1/1 (1.00)
Record:        Win
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py --layout bigMaze --zoom 0.5
```



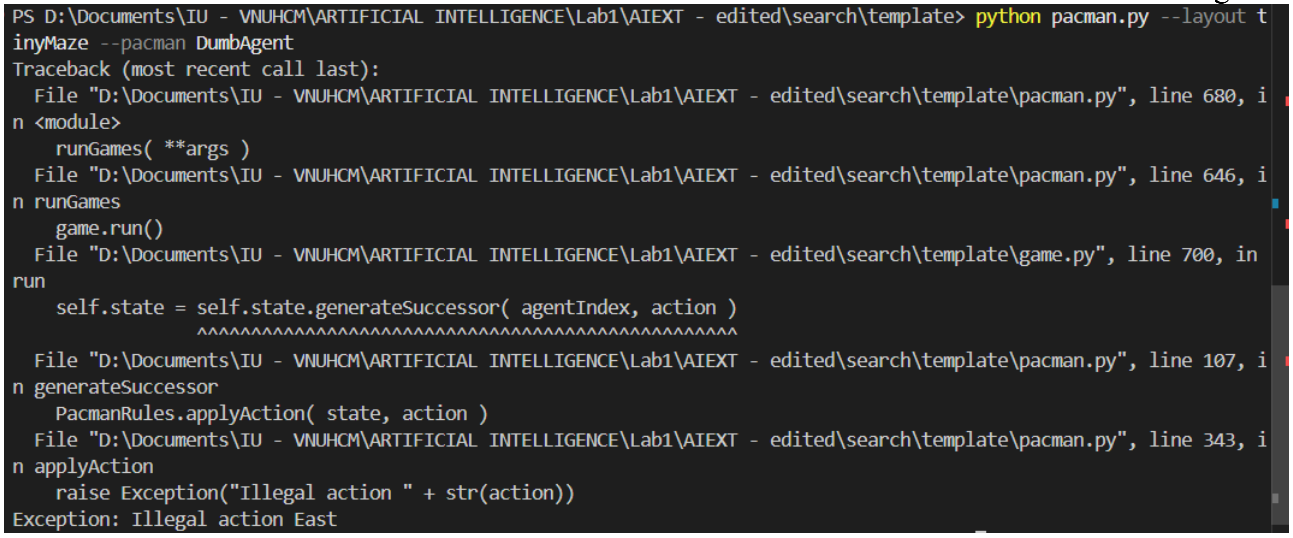
Tried creating DumbAgent and running it in tinyMaze to test pacman's move



search > template > Agents.py > DumbAgent > getAction

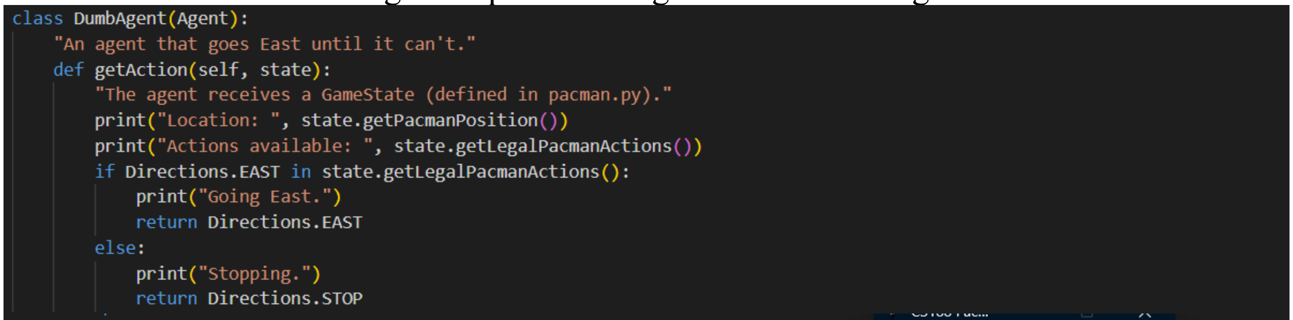
```python
from game import Agent
from game import Directions
import random

class DumbAgent(Agent):
    "An agent that goes East until it can't"
    def getAction(self, state):
        "The agent always goes East"
        return Directions.EAST
```

The game crashed with an 'Illegal Action East'. The error message indicated that the DumbAgent was trying to perform an illegal action, specifically moving East when it was not allowed.

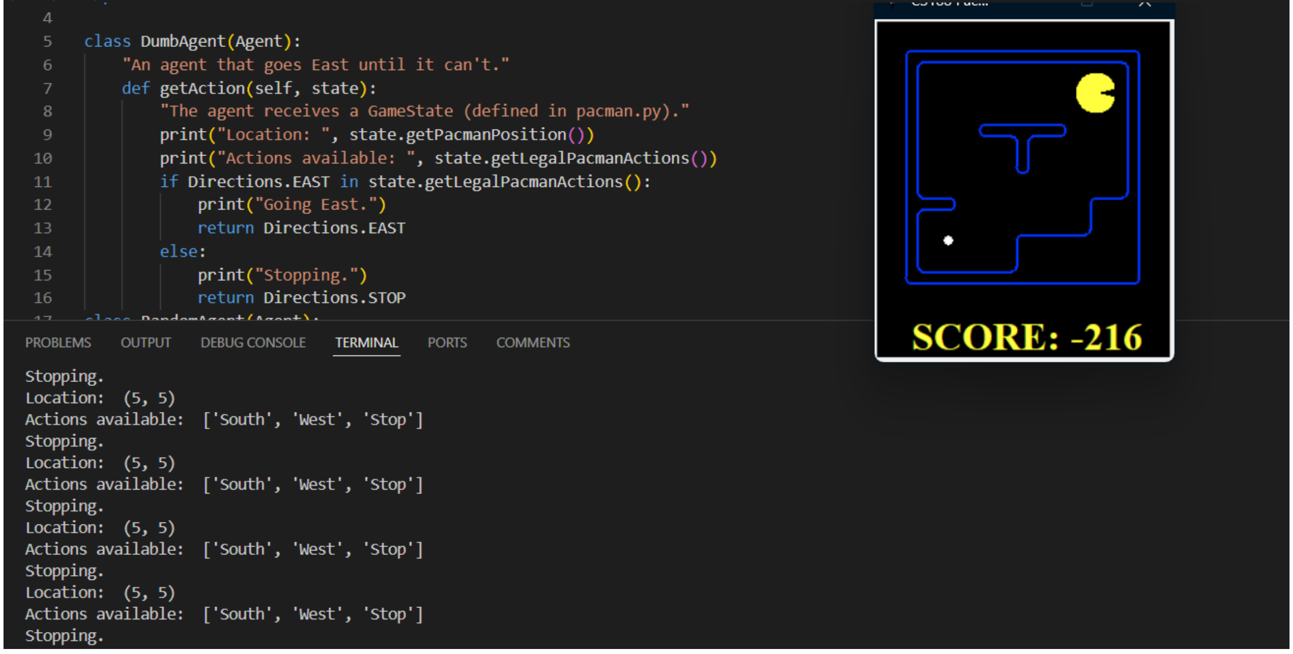


```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py --layout t
inyMaze --pacman DumbAgent
Traceback (most recent call last):
  File "D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template\pacman.py", line 680, i
n <module>
    runGames( **args )
  File "D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template\pacman.py", line 646, i
n runGames
    game.run()
  File "D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template\game.py", line 700, in
run
    self.state = self.state.generateSuccessor( agentIndex, action )
                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template\pacman.py", line 107, i
n generateSuccessor
    PacmanRules.applyAction( state, action )
  File "D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template\pacman.py", line 343, i
n applyAction
    raise Exception("Illegal action " + str(action))
Exception: Illegal action East
```

Tried modified the DumbAgent to prevent the game from crashing



```python
class DumbAgent(Agent):
    "An agent that goes East until it can't."
    def getAction(self, state):
        "The agent receives a GameState (defined in pacman.py)."
        print("Location: ", state.getPacmanPosition())
        print("Actions available: ", state.getLegalPacmanActions())
        if Directions.EAST in state.getLegalPacmanActions():
            print("Going East.")
            return Directions.EAST
        else:
            print("Stopping.")
            return Directions.STOP
```

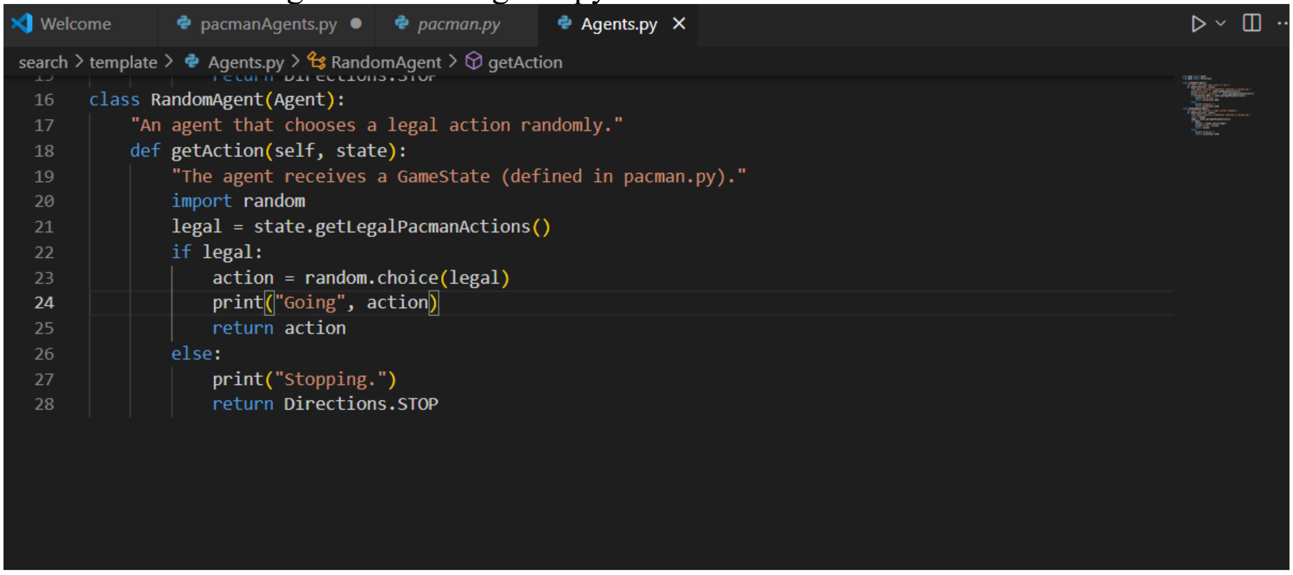


```python
class DumbAgent(Agent):
    "An agent that goes East until it can't."
    def getAction(self, state):
        "The agent receives a GameState (defined in pacman.py)."
        print("Location: ", state.getPacmanPosition())
        print("Actions available: ", state.getLegalPacmanActions())
        if Directions.EAST in state.getLegalPacmanActions():
            print("Going East.")
            return Directions.EAST
        else:
            print("Stopping.")
            return Directions.STOP
```

```
Stopping.
Location:  (5, 5)
Actions available:  ['South', 'West', 'Stop']
Stopping.
Location:  (5, 5)
Actions available:  ['South', 'West', 'Stop']
Stopping.
Location:  (5, 5)
Actions available:  ['South', 'West', 'Stop']
Stopping.
Location:  (5, 5)
Actions available:  ['South', 'West', 'Stop']
Stopping.
```

## 1. Exercise 1:

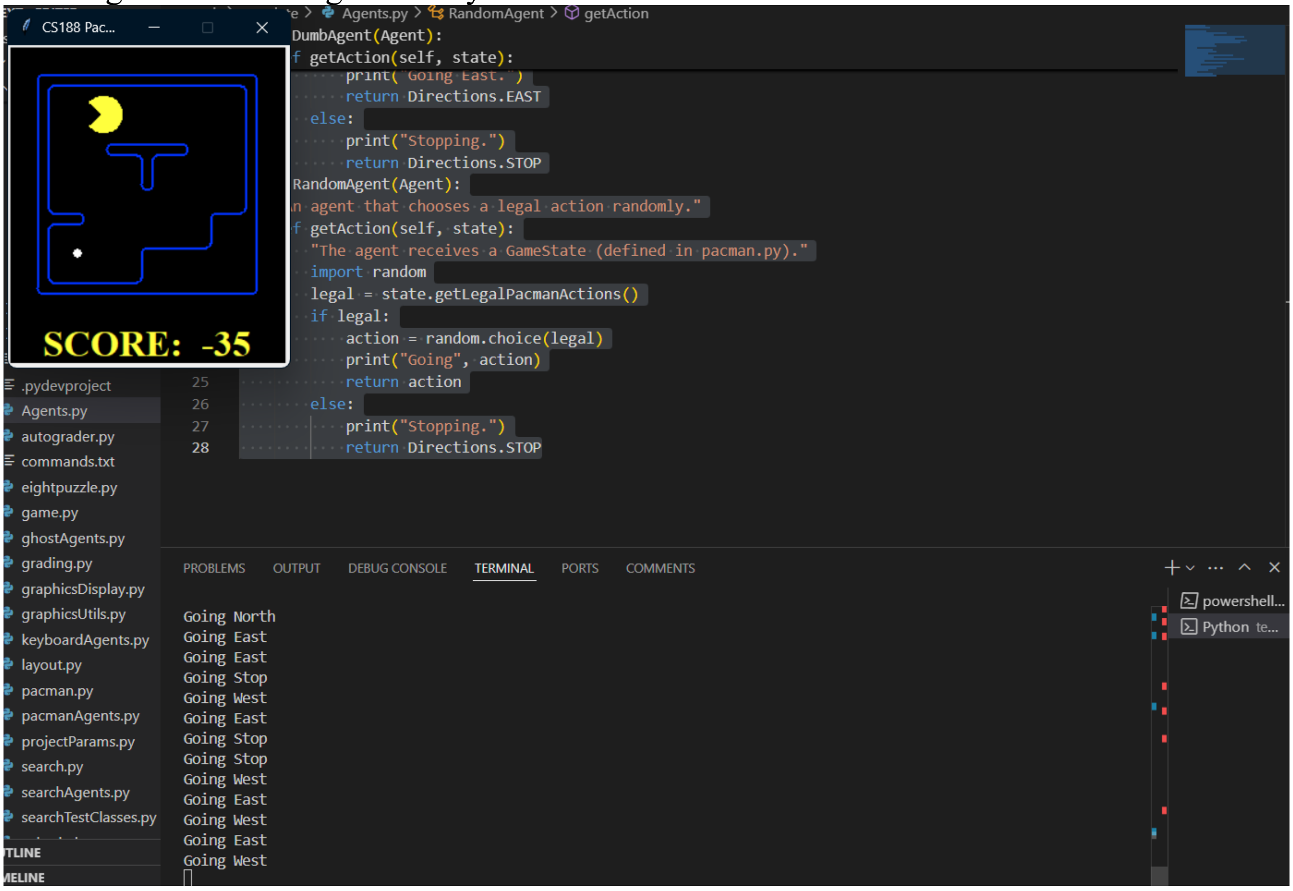
Created a RandomAgent class in Agents.py



```
class RandomAgent(Agent):
    "An agent that chooses a legal action randomly."
    def getAction(self, state):
        "The agent receives a GameState (defined in pacman.py)."
        import random
        legal = state.getLegalPacmanActions()
        if legal:
            action = random.choice(legal)
            print("Going", action)
            return action
        else:
            print("Stopping.")
            return Directions.STOP
```

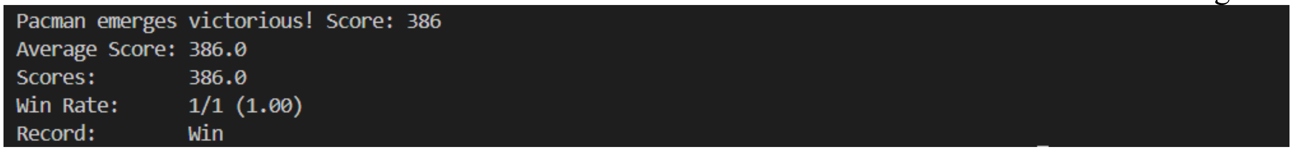
Since it was random, it always got the food ultimately, whether it did so right away or took a while. Because it only pursued legal action, it didn't collapse. The agent moved in numerous directions in a short time; it could be looped again with the previous step or go to the new position. Perhaps, it took time for an agent to reach the food position, so the result was not quite good, and if there were enemies, there was a high probability of losing because the agent could not shirk the danger. RandomAgent moved in a random direction, including stopping. It sometimes stopped on the wall.

Running the RandomAgent in tinyMaze



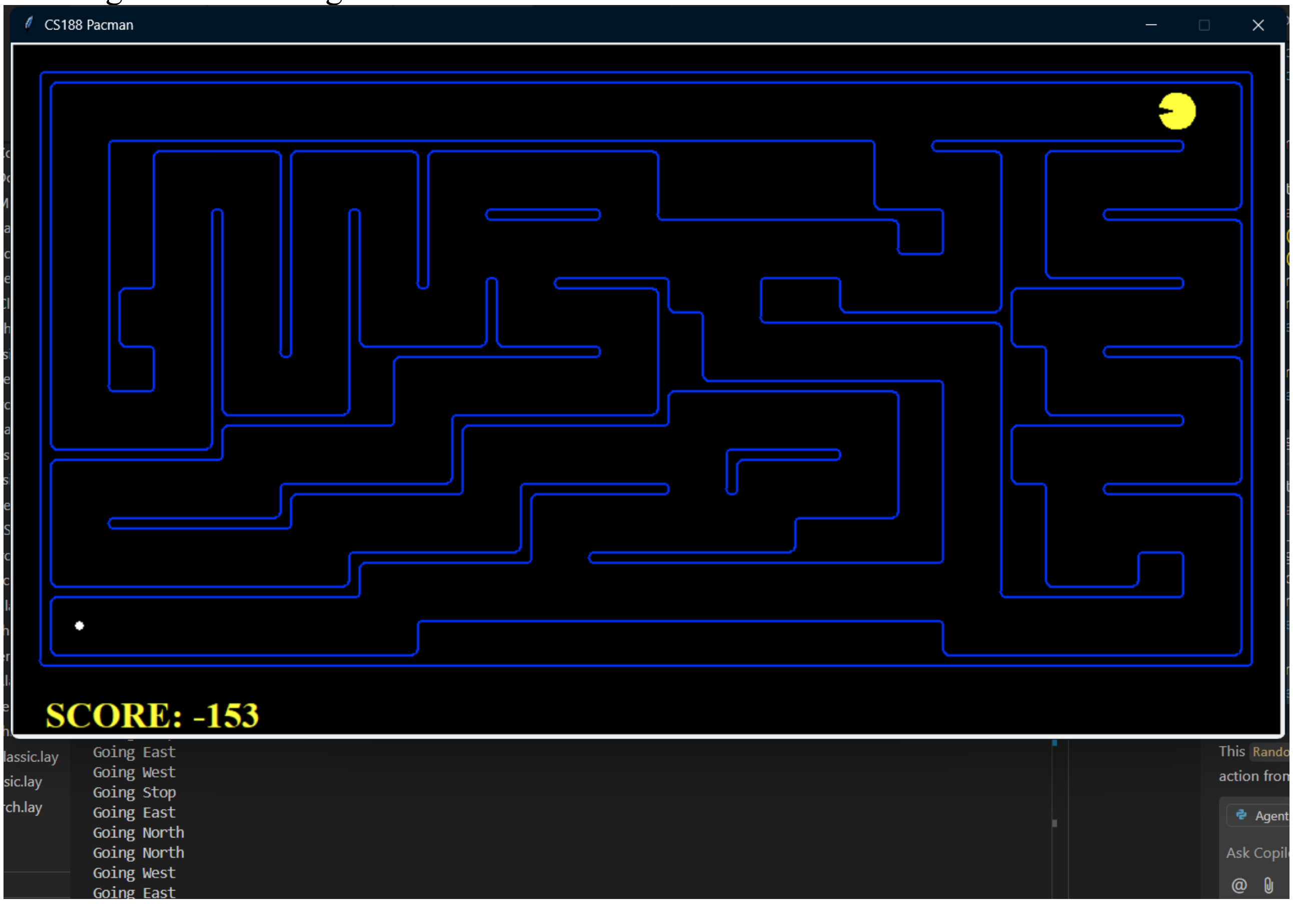
```
DumbAgent(Agent):
f getAction(self, state):
        print("Going East.")
        return Directions.EAST
    else:
        print("Stopping.")
        return Directions.STOP
RandomAgent(Agent):
n agent that chooses a legal action randomly."
f getAction(self, state):
    "The agent receives a GameState (defined in pacman.py)."
    import random
    legal = state.getLegalPacmanActions()
    if legal:
        action = random.choice(legal)
        print("Going", action)
        return action
    else:
        print("Stopping.")
        return Directions.STOP
```

```
Going North
Going East
Going East
Going Stop
Going West
Going East
Going Stop
Going Stop
Going West
Going East
Going West
Going East
Going West
```



```
Pacman emerges victorious! Score: 386
Average Score: 386.0
Scores:        386.0
Win Rate:      1/1 (1.00)
Record:        Win
```

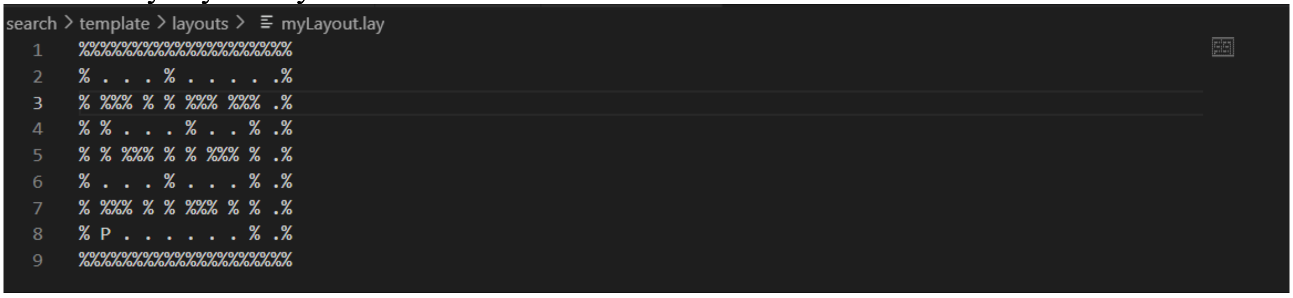
Running the RandomAgent in mediumMaze



```
Going East
Going West
Going Stop
Going East
Going North
Going North
Going West
Going East
```

## 2. Exercise 2:

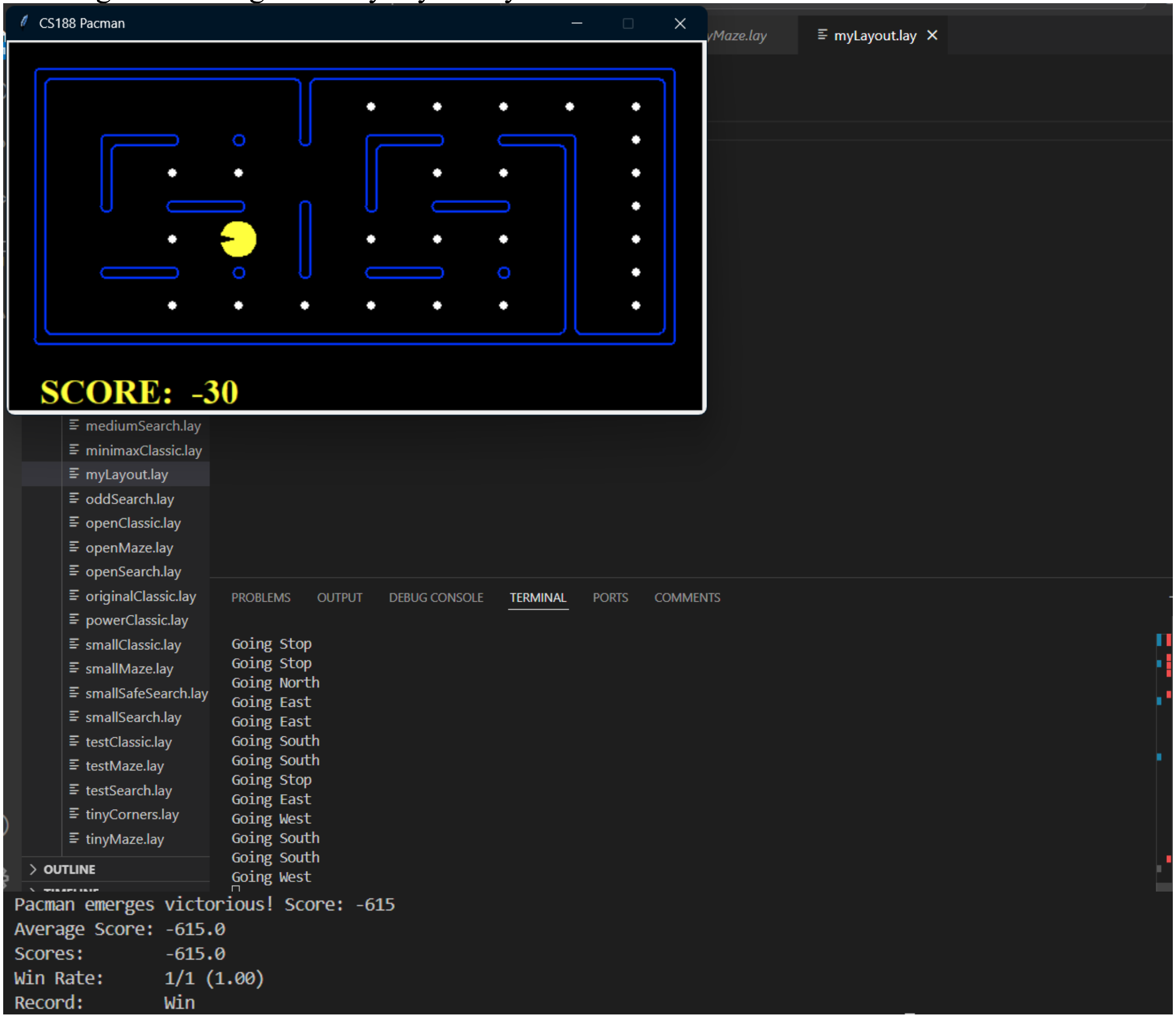
Created myLayout.lay



```
search > template > layouts > myLayout.lay
%%%%%%%%%%%%%%%%%%%%
% . . . % . . . . .%
% %%% % % %%% %%% .%
% % . . . % . . % .%
% % %%% % % %%% % .%
% . . . % . . . % .%
% %%% % % %%% % % .%
% P . . . . . . % .%
%%%%%%%%%%%%%%%%%%%%
```

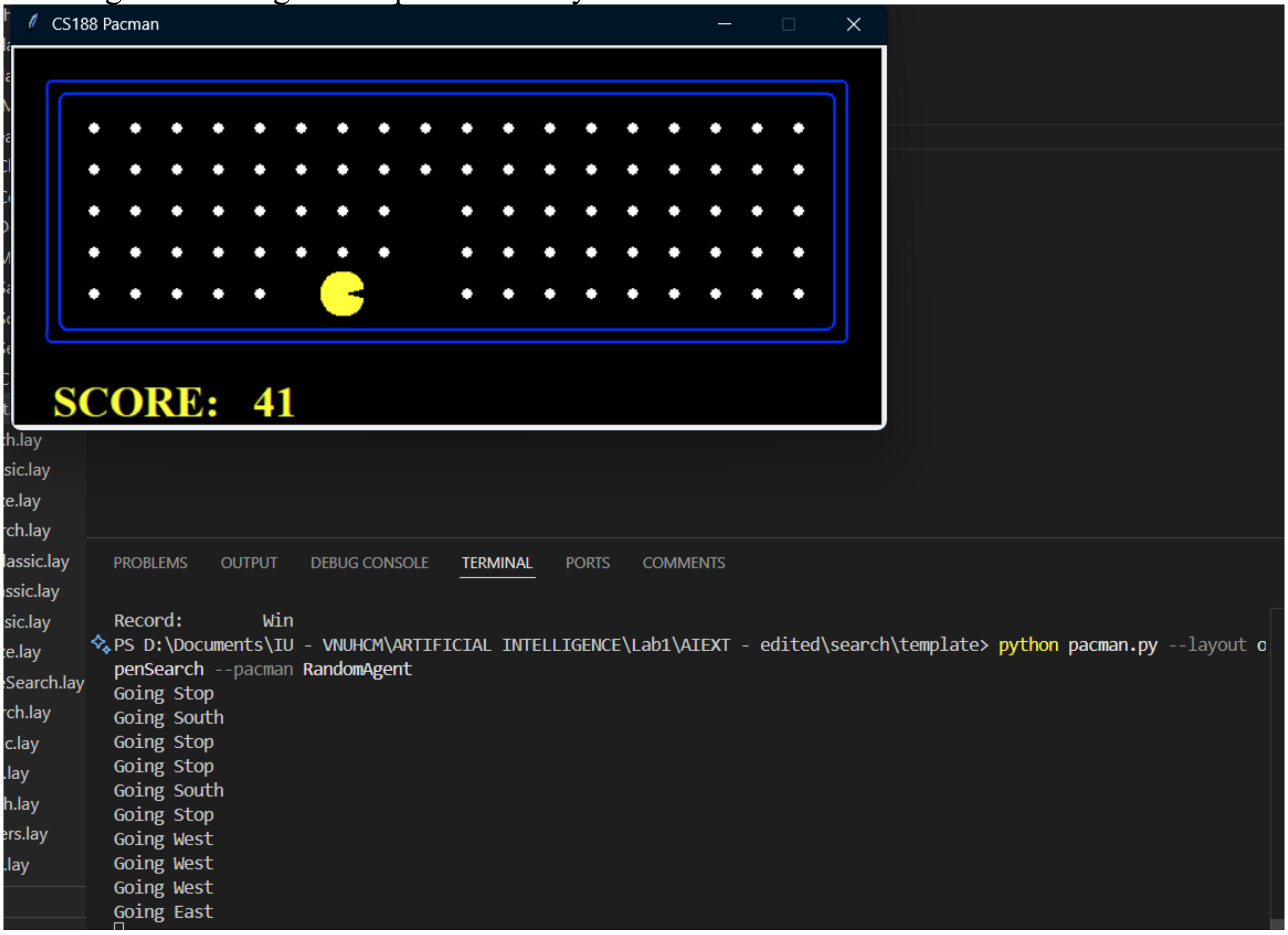
- For the myLayout.lay, each "%" defined the wall for the game rendering. They could have many different shapes, but the prerequisite requirement was that the map had to be wrapped by "%". According to this, the game system worked smoothly without error.
- Each "." defined the food that the agent could eat to win the game. If there was no "." inside the layout, the map described it as a space, which the agent could move across.
- The "P" defined the initial position of the agent (Pac-Man).

Running RandomAgent in myLayout.lay



```
Going Stop
Going Stop
Going North
Going East
Going East
Going South
Going South
Going Stop
Going East
Going West
Going South
Going South
Going West
Pacman emerges victorious! Score: -615
Average Score: -615.0
Scores:        -615.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Running RandomAgent in openSearch.lay

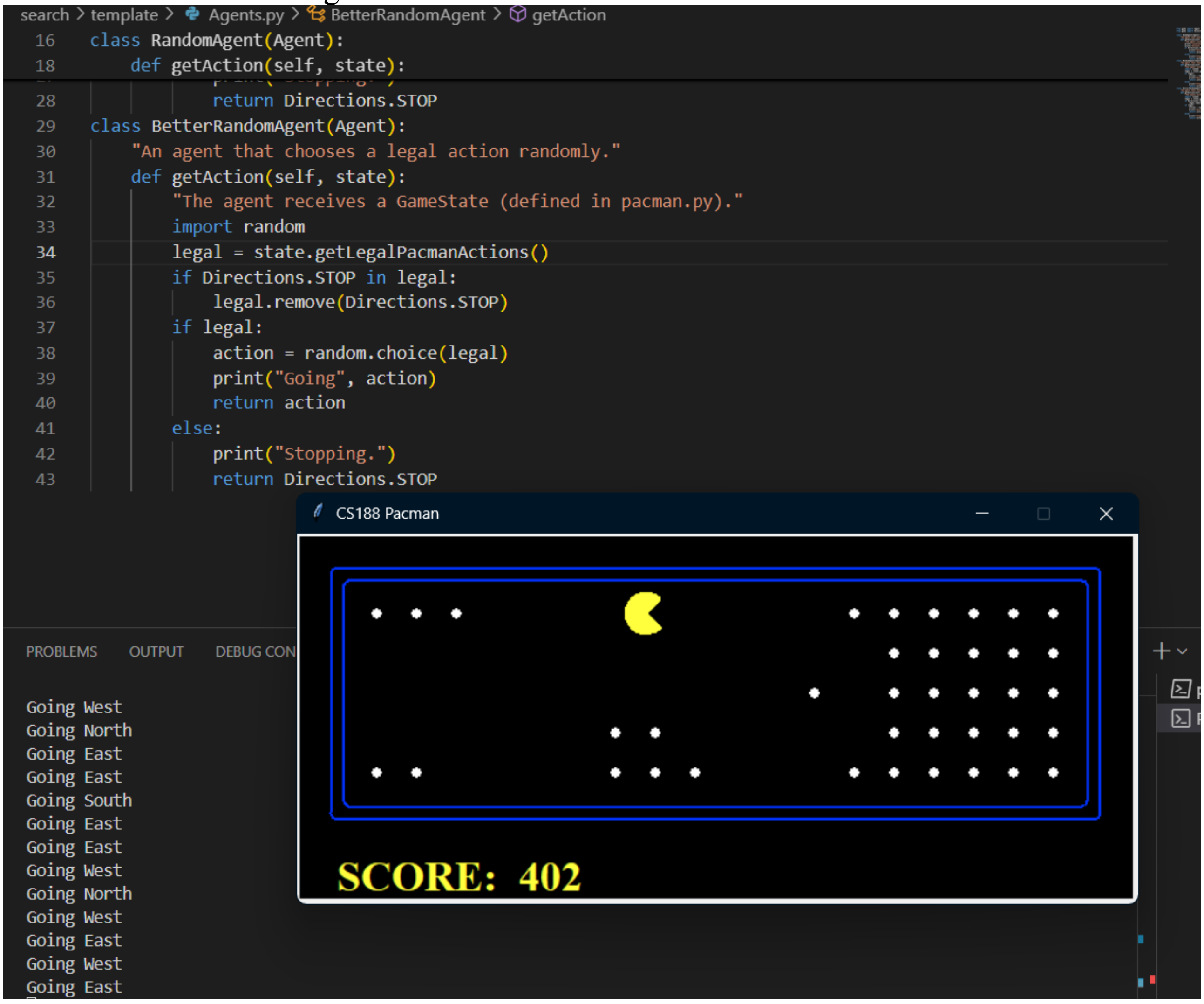


```
Record:        Win
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py --layout o
penSearch --pacman RandomAgent
Going Stop
Going South
Going Stop
Going Stop
Going South
Going Stop
Going West
Going West
Going West
Going East
```

While running RandomAgent in openSearch.lay:
Attempt 1 - 200 in 11.07s
Attempt 2 - 115 in 11.00s
Attempt 3 - 160 in 11.01s
Attempt 4 - 210 in 10.97s
Attempt 5 - 99 in 11.10s
Attempt 6 - 190 in 11.03s
=> Average 162.3 pts in 11s with RandomAgent in openSearch environment.
It typically consumed everything in the center and was less likely to consume food on the edges.

## 3. Exercise 3:

Created BetterRandomAgent


search > template > Agents.py > BetterRandomAgent > getAction
PROBLEMS
OUTPUT
DEBUG CON
CS188 Pacman

```python
class RandomAgent(Agent):
    def getAction(self, state):
            return Directions.STOP
class BetterRandomAgent(Agent):
    "An agent that chooses a legal action randomly."
    def getAction(self, state):
        "The agent receives a GameState (defined in pacman.py)."
        import random
        legal = state.getLegalPacmanActions()
        if Directions.STOP in legal:
            legal.remove(Directions.STOP)
        if legal:
            action = random.choice(legal)
            print("Going", action)
            return action
        else:
            print("Stopping.")
            return Directions.STOP
```

```
Going West
Going North
Going East
Going East
Going South
Going East
Going East
Going West
Going North
Going West
Going East
Going West
Going East
```

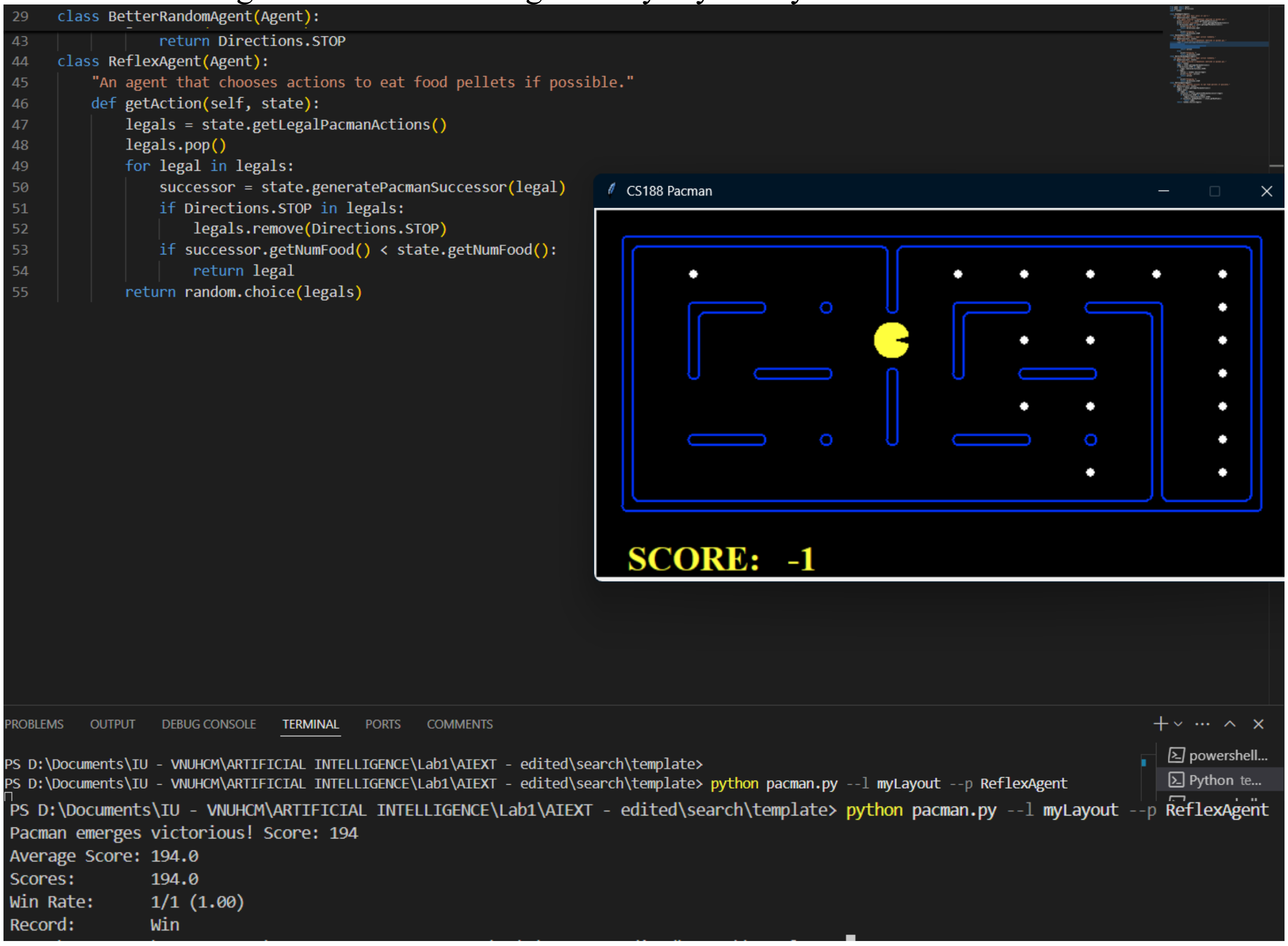
SCORE: 402

Running BetterRandomAgent in openSearch.lay:
Attempt 1 - 225 in 11.07s
Attempt 2 - 212 in 11.15s
Attempt 3 - 160 in 11.00s
Attempt 4 - 256 in 11.01s
Attempt 5 - 145 in 11.12s
Attempt 6 - 155 in 11.10s
=> Average 192.17 points in 11 seconds with BetterRandomAgent in the openSearch environment.

BetterRandomAgent traveled without halting in a random path. RandomAgent moved more slowly than BetterRandomAgent. As a result, it was penalized with fewer points. The agent moved in the direction of the food path, and when there were spaces, it went randomly until it found the food in the openSearch layout.

## 4. Exercise 4:

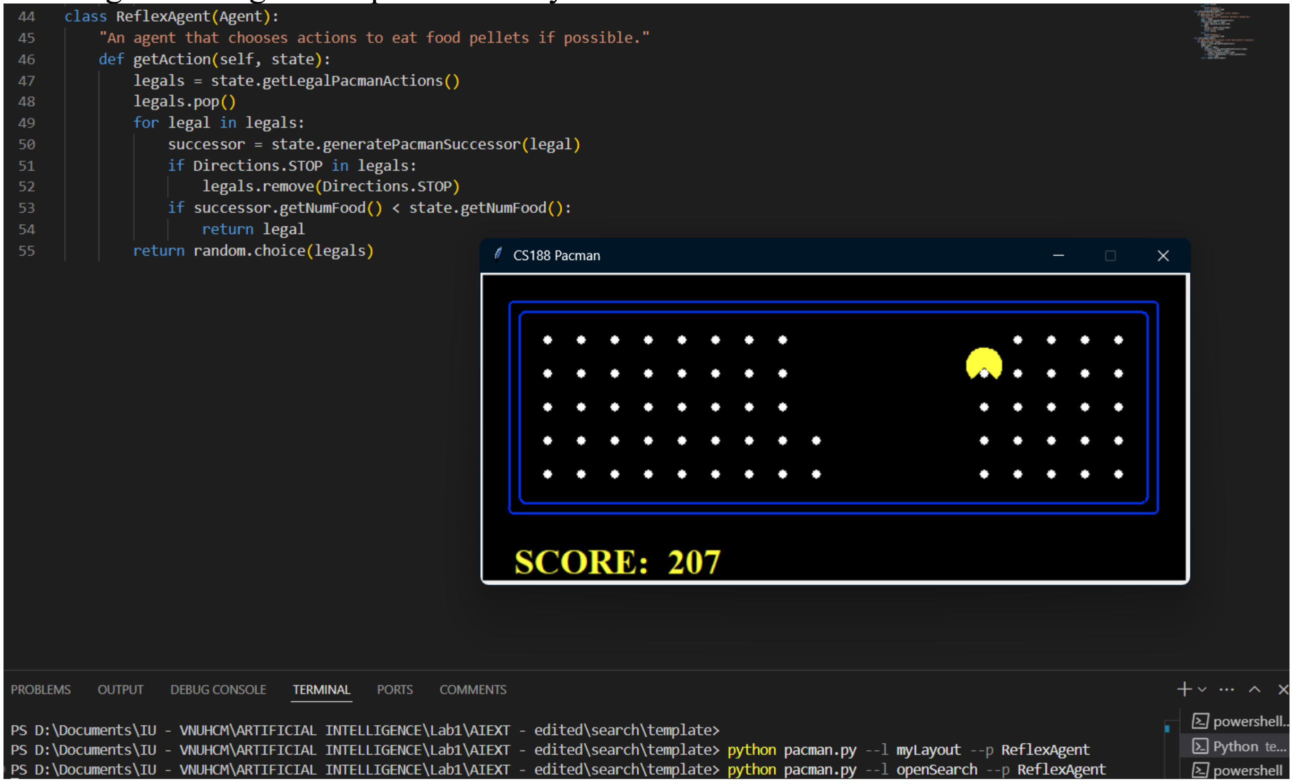
Created ReflexAgent and tried running it in myLayout.lay



```
class BetterRandomAgent(Agent):
            return Directions.STOP
class ReflexAgent(Agent):
    "An agent that chooses actions to eat food pellets if possible."
    def getAction(self, state):
        legals = state.getLegalPacmanActions()
        legals.pop()
        for legal in legals:
            successor = state.generatePacmanSuccessor(legal)
            if Directions.STOP in legals:
                legals.remove(Directions.STOP)
            if successor.getNumFood() < state.getNumFood():
                return legal
        return random.choice(legals)
```

```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template>
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py --l myLayout --p ReflexAgent
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py --l myLayout --p ReflexAgent
Pacman emerges victorious! Score: 194
Average Score: 194.0
Scores:        194.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Running ReflexAgent in openSearch.lay



```
class ReflexAgent(Agent):
    "An agent that chooses actions to eat food pellets if possible."
    def getAction(self, state):
        legals = state.getLegalPacmanActions()
        legals.pop()
        for legal in legals:
            successor = state.generatePacmanSuccessor(legal)
            if Directions.STOP in legals:
                legals.remove(Directions.STOP)
            if successor.getNumFood() < state.getNumFood():
                return legal
        return random.choice(legals)
```

```
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template>
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py --l myLayout --p ReflexAgent
PS D:\Documents\IU - VNUHCM\ARTIFICIAL INTELLIGENCE\Lab1\AIEXT - edited\search\template> python pacman.py --l openSearch --p ReflexAgent
```

The agent always used the zig-zag technique to consume the entire right side of the map in an openSearch environment. Then, at random, it entered the other side of the map to locate food pellets, which it then consumed entirely. Of course, for the remaining part, it continued to use the zig-zag technique to consume all the food. The time for the game to finish in the openSearch layout using the ReflexAgent class was quite higher than in the BetterRandomAgent one. However, overall, they were similar to each other.

Pac-man's position: **gameState.getPacmanPosition()**
All gosts' position: **gameState.getGhostPositions()**
Walls' location: **gameState.getWalls()**
Capsules' position: **gameState.getCapsules()**
Each food pellet's position: **gameState.getFood()**
The total number of food pellets still available: **gameState.getNumFood()**
Whether it has won or lost the game: **gameState.isLose()**, **gameState.isWin()**
Pac-man's current score in the game: **gameState.getScore()**