| **HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  **KHOA CÔNG NGHỆ THÔNG TIN 1**  -------------------------------------------------    **Báo cáo bài tập lớn**  **Môn học: Lập trình với Python**  Chủ đề: **Python + PyTorch + Pygame Reinforcement Learning – Train an AI to Play Snake**  Giáo viên giảng day : Phạm Văn Sự  Nhóm: PYPROGF22PRJ.28  Thông tin thành viên:  Lê Duy Mạnh - B20DCCN423  Nguyễn Hoàng Tùng - B20DCCN627  Lường Thanh Huy - B20CCN312 |
| --- |
|  |

**Phần 1: Cơ sở lý thuyết**

**Phần 2: Tạo trò chơi (môi trường cho agent)**

**Phần 3: Tạo agent**

**Phần 4: Triển khai model**

Phần 1: Cơ sở lý thuyết

1.1 Lý thuyết về học tăng cường

Học tăng cường (RL) là một lĩnh vực học máy liên quan đến cách các tác nhân thông minh nên thực hiện các hành động trong một môi trường để tối đa hóa khái niệm phần thưởng tích lũy.

Rl đang dạy một Software agent cách ứng xử trong một môi trường bằng cách nói cho nó biết nó hoạt động tốt như thế nào.

1.2 Deep Q Learning

phương pháp này mở rộng việc học tăng cường bằng cách sử dụng mạng lưới thần kinh sâu để dự đoán các hành động

1.3 Tổng quan :

Agent

* game
* model

Training:

* state = get\_state(game)
* action = get\_move(state)
  + model.predict()

- reward, game\_over, score = game.play\_step(action)

- new\_state = get\_state(game)

- remember

- model.train()

Game(Pygame)

* play\_step(action)

→ reward, game\_over, score

Model (Pytorch)

Linear\_QNet(DQN)

* model.predict(state)

→ action

Reward

* eat food : +10 p
* game over: -10 p
* else: 0 p

Action

[1, 0, 0] -> đi thẳng

[0, 1, 0] -> rẽ phải

[0, 0, 1] -> rẽ trái

?Không có hành động đi ngược do đi ngược con rắn sẽ chết ngay lập tức

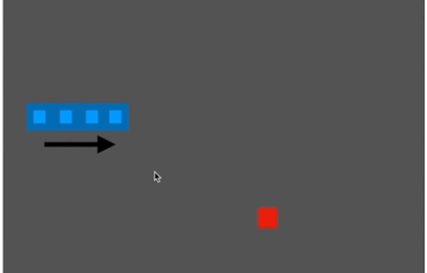
State (11 values) :boolean

gồm: [danger straight , danger right, danger left,

direction left, direction right, direction up, direction down

food left, food right, food up, food down]

Ví dụ:



state tương ứng:

[0, 0, 0,

0, 1, 0 , 0,

0, 1, 0, 1]

→ direction right, food right, food down

1.4 (Deep) Q Learning

Các bước:

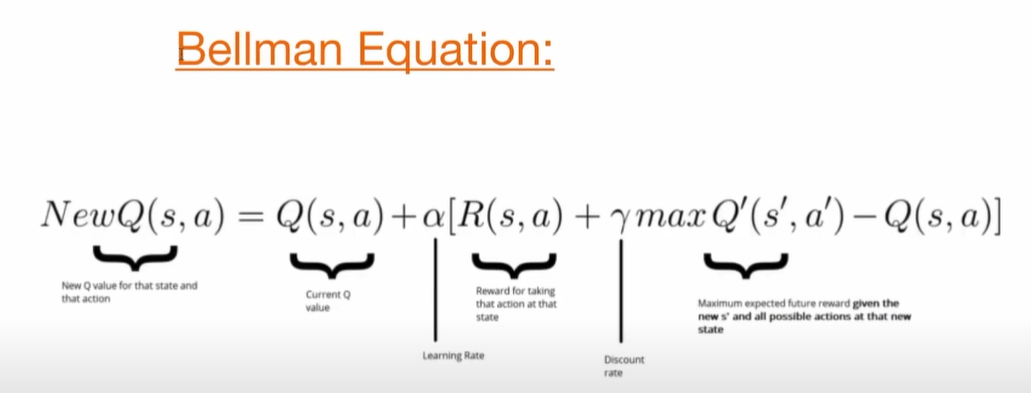
0. Init Q Value ⇔ Init model

1. Choose action (model.predict(state) / random move)

2. Perform action

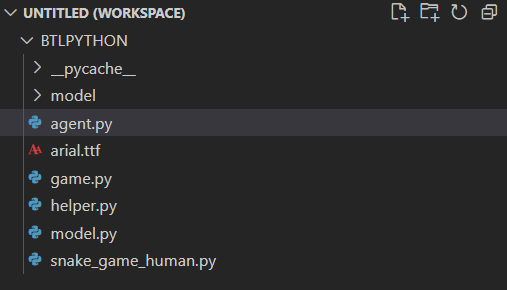
3. Measure reward

4. Update Q value (+train model)



Phần 2: Tạo trò chơi

B1: tạo môi trường



Sử dụng anaconda để quản lý dự án:

install một số thư viện cần thiết để thực hiện trò chơi:

* pygame
* pytorch
* matplotlib (để vẽ đồ thị việc học của agent)
* numpy

1.Tạo game snake cho người chơi

import pygame

import random

from enum import Enum

from collections import namedtuple

pygame.init()

font = pygame.font.Font('arial.ttf', 25)

#font = pygame.font.SysFont('arial', 25)

class Direction(Enum):

RIGHT = 1

LEFT = 2

UP = 3

DOWN = 4

Point = namedtuple('Point', 'x, y')

# rgb colors

WHITE = (255, 255, 255)

RED = (200,0,0)

BLUE1 = (0, 0, 255)

BLUE2 = (0, 100, 255)

BLACK = (0,0,0)

BLOCK\_SIZE = 20

SPEED = 20

class SnakeGame:

def \_\_init\_\_(self, w=640, h=480):

self.w = w

self.h = h

# init display

self.display = pygame.display.set\_mode((self.w, self.h))

pygame.display.set\_caption('Snake')

self.clock = pygame.time.Clock()

# init game state

self.direction = Direction.RIGHT

self.head = Point(self.w/2, self.h/2)

self.snake = [self.head,

Point(self.head.x-BLOCK\_SIZE, self.head.y),

Point(self.head.x-(2\*BLOCK\_SIZE), self.head.y)]

self.score = 0

self.food = None

self.\_place\_food()

def \_place\_food(self):

x = random.randint(0, (self.w-BLOCK\_SIZE )//BLOCK\_SIZE )\*BLOCK\_SIZE

y = random.randint(0, (self.h-BLOCK\_SIZE )//BLOCK\_SIZE )\*BLOCK\_SIZE

self.food = Point(x, y)

if self.food in self.snake:

self.\_place\_food()

def play\_step(self):

# 1. nhan dau vao tu ban phim nguoi dung

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

quit()

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

self.direction = Direction.LEFT

elif event.key == pygame.K\_RIGHT:

self.direction = Direction.RIGHT

elif event.key == pygame.K\_UP:

self.direction = Direction.UP

elif event.key == pygame.K\_DOWN:

self.direction = Direction.DOWN

# 2. move

self.\_move(self.direction) # update the head

self.snake.insert(0, self.head)

# 3. check if game over

game\_over = False

if self.\_is\_collision():

game\_over = True

return game\_over, self.score

# 4. place new food or just move

if self.head == self.food:

self.score += 1

self.\_place\_food()

else:

self.snake.pop()

# 5. update ui and clock

self.\_update\_ui()

self.clock.tick(SPEED)

# 6. return game over and score

return game\_over, self.score

def \_is\_collision(self):

# hits boundary

if self.head.x > self.w - BLOCK\_SIZE or self.head.x < 0 or self.head.y > self.h - BLOCK\_SIZE or self.head.y < 0:

return True

# hits itself

if self.head in self.snake[1:]:

return True

return False

def \_update\_ui(self):

self.display.fill(BLACK)

for pt in self.snake:

pygame.draw.rect(self.display, BLUE1, pygame.Rect(pt.x, pt.y, BLOCK\_SIZE, BLOCK\_SIZE))

pygame.draw.rect(self.display, BLUE2, pygame.Rect(pt.x+4, pt.y+4, 12, 12))

pygame.draw.rect(self.display, RED, pygame.Rect(self.food.x, self.food.y, BLOCK\_SIZE, BLOCK\_SIZE))

text = font.render("Score: " + str(self.score), True, WHITE)

self.display.blit(text, [0, 0])

pygame.display.flip()

def \_move(self, direction):

x = self.head.x

y = self.head.y

if direction == Direction.RIGHT:

x += BLOCK\_SIZE

elif direction == Direction.LEFT:

x -= BLOCK\_SIZE

elif direction == Direction.DOWN:

y += BLOCK\_SIZE

elif direction == Direction.UP:

y -= BLOCK\_SIZE

self.head = Point(x, y)

if \_\_name\_\_ == '\_\_main\_\_':

game = SnakeGame()

# game loop

while True:

game\_over, score = game.play\_step()

if game\_over == True:

break

print('Final Score', score)

pygame.quit()

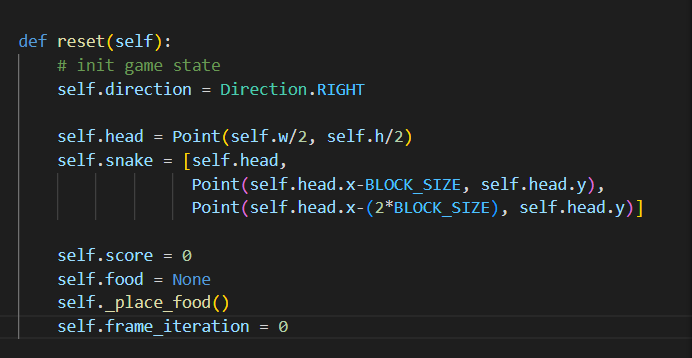
Phần 3.Tạo Agent

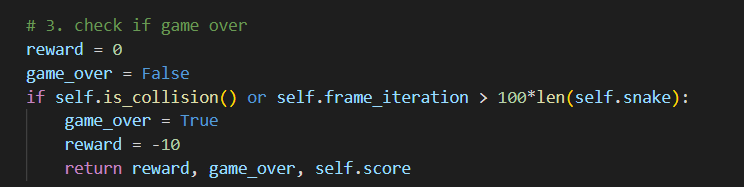
3.1 sửa môi trường

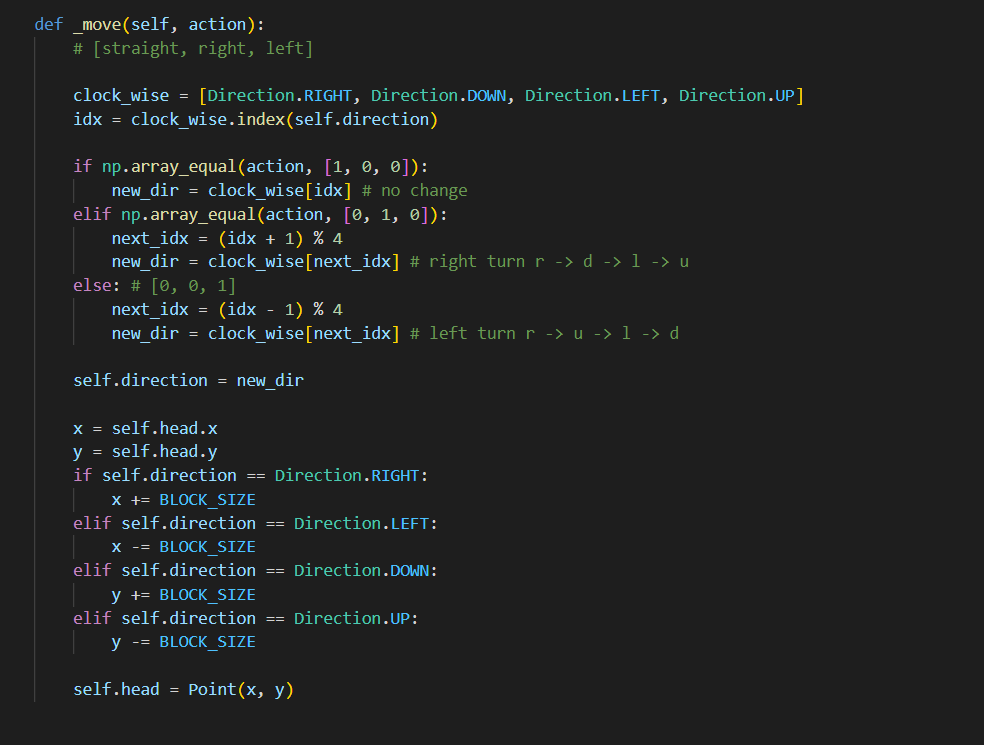
để agent có thể tự động chạy ta cần cài đặt thêm một số chức năng:

* reset game
* reward
* play(action) -> direction
* game\_iteration
* is\_collision

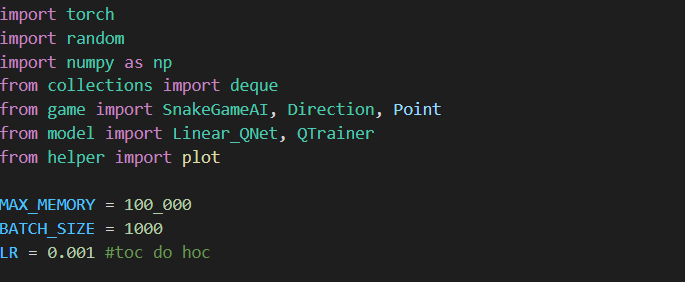
Thay vì dùng direction thì ta thay đổi nó thành action do agent tự tạo







33.2 Tạo Agent

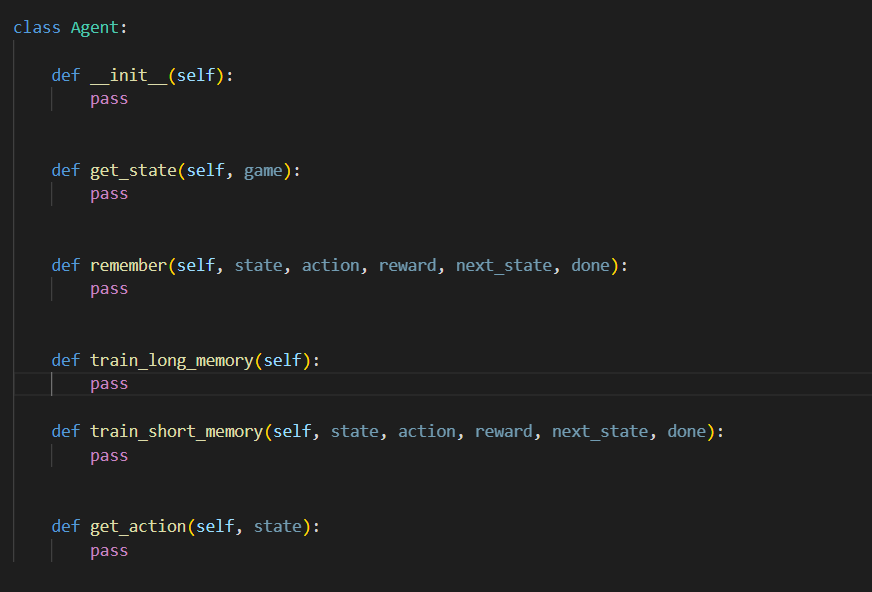


* import các thư viện cần thiết
* khai báo 3 biến toàn cục:

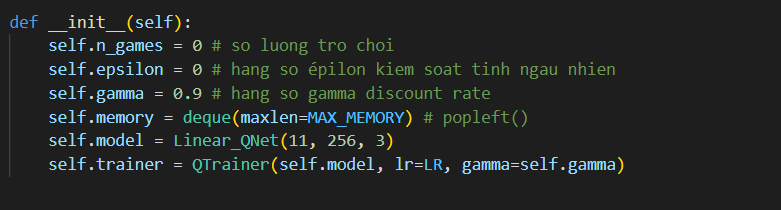
+MAX\_MEMORY: bộ nhớ tối đa của deque để lưu các dữ liêụ mỗi lần học

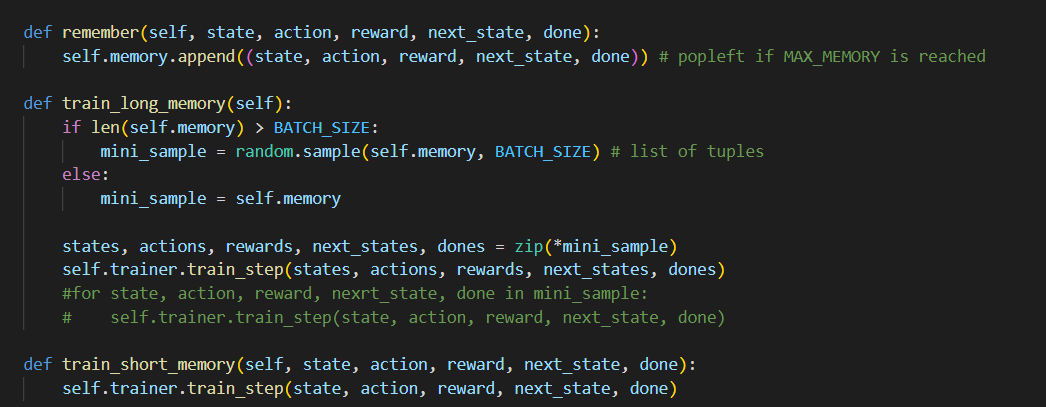
+ BATCH\_SIZE: size của trò chơi

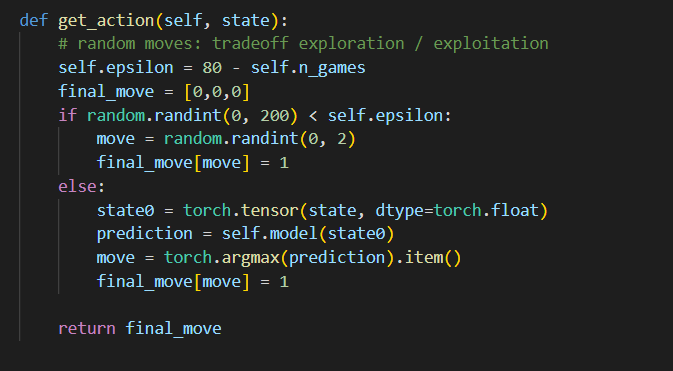
+ LR: tốc độ học của agent

* những phương thức của lớp Agent
* 

cài đặt cụ thể các phương thức:



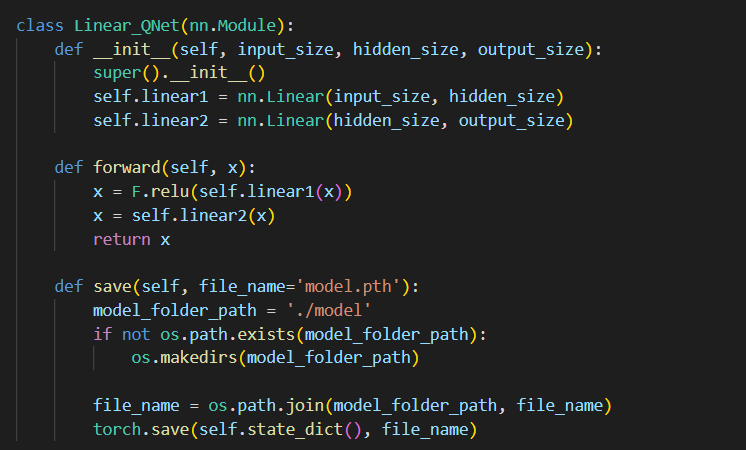


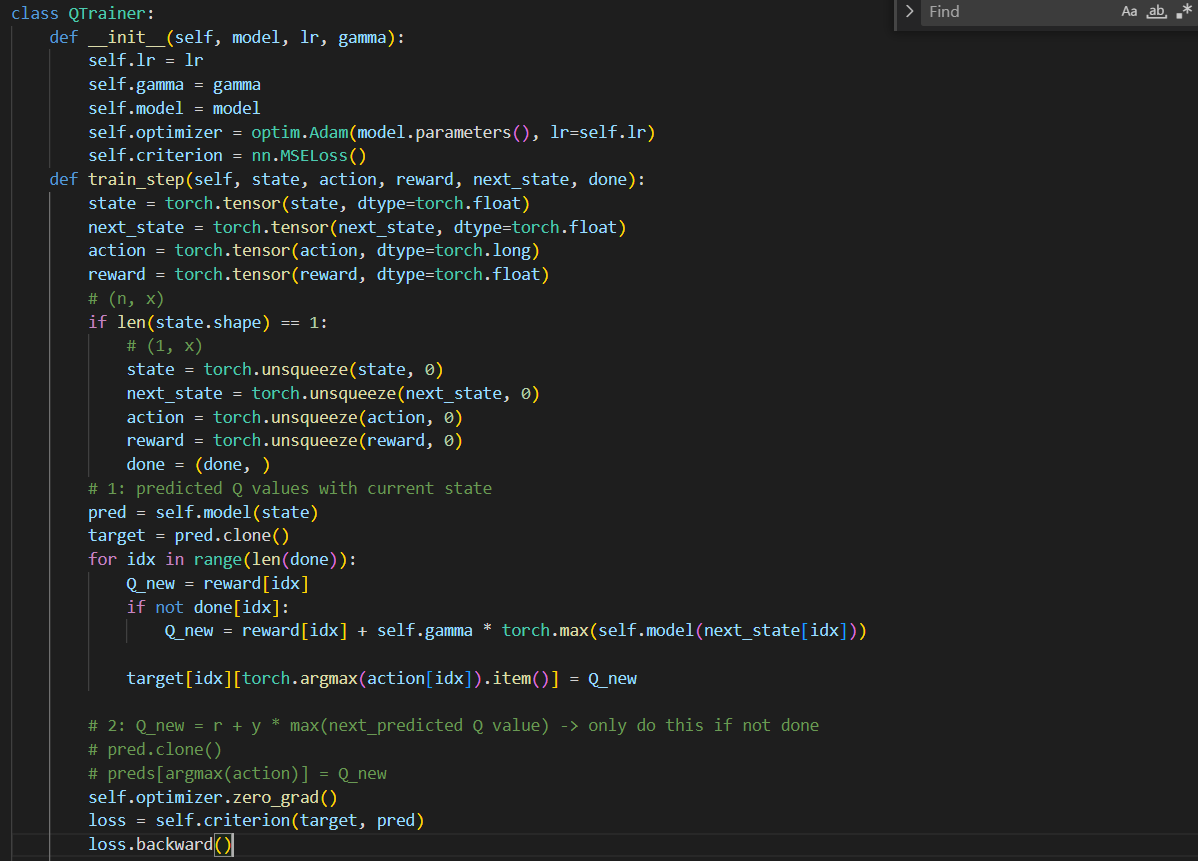


Phần 4: Triển khai model

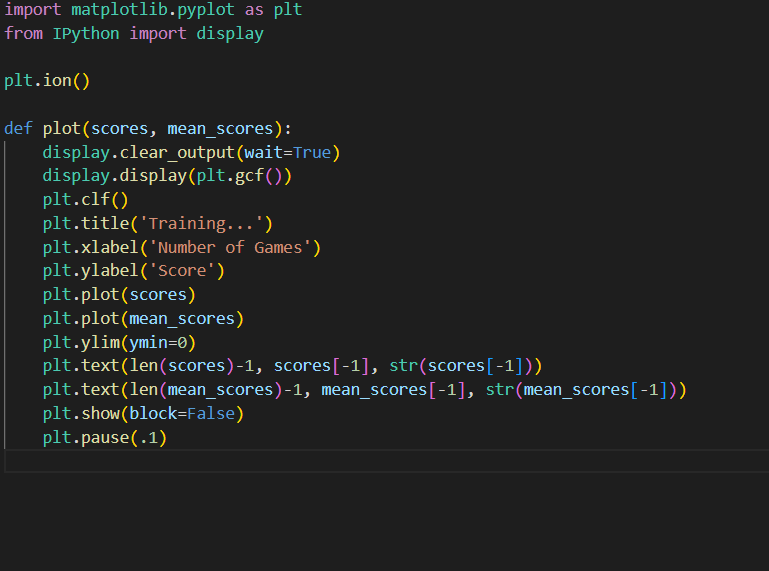
* import các thư viện cần thiết



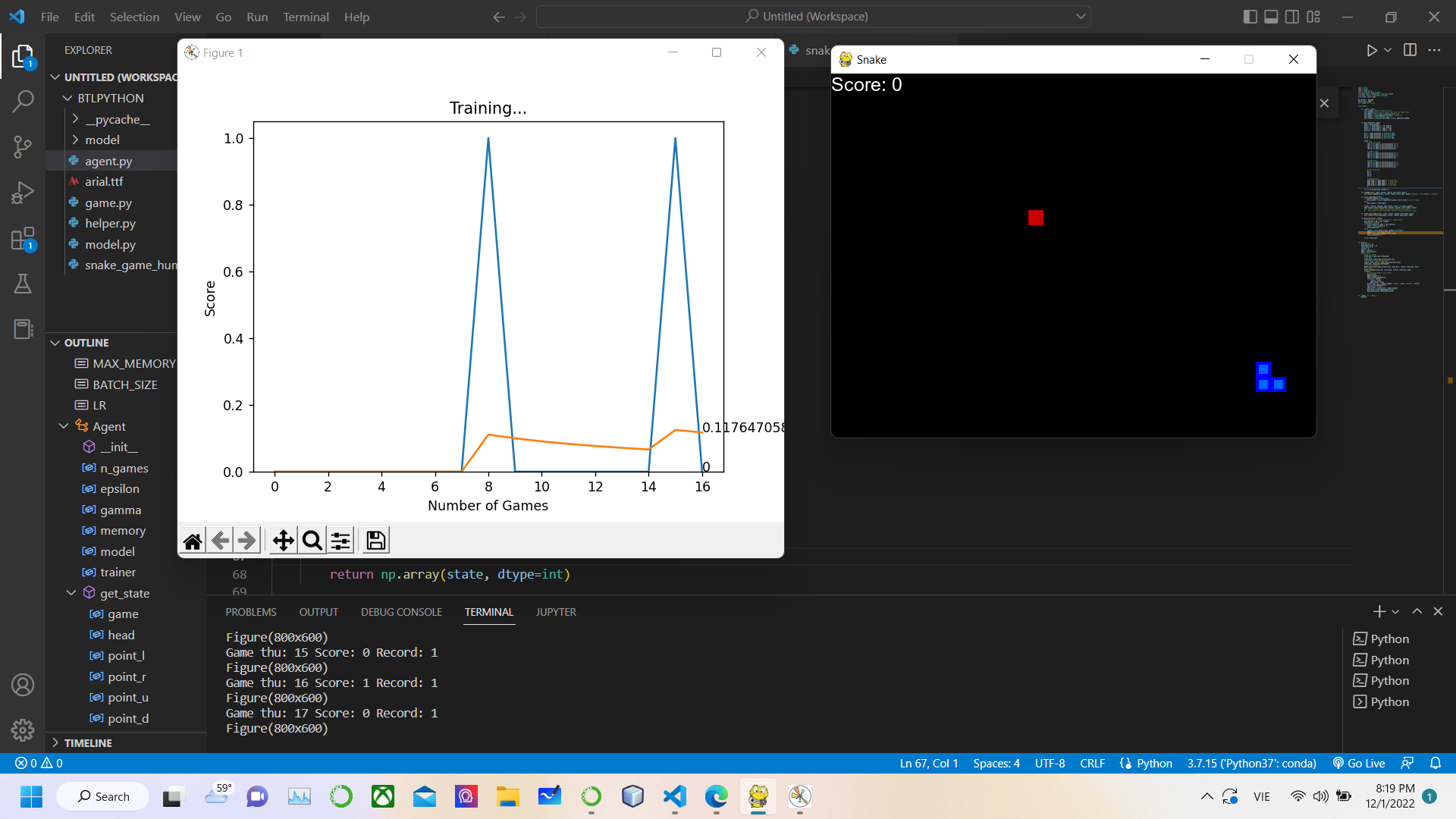




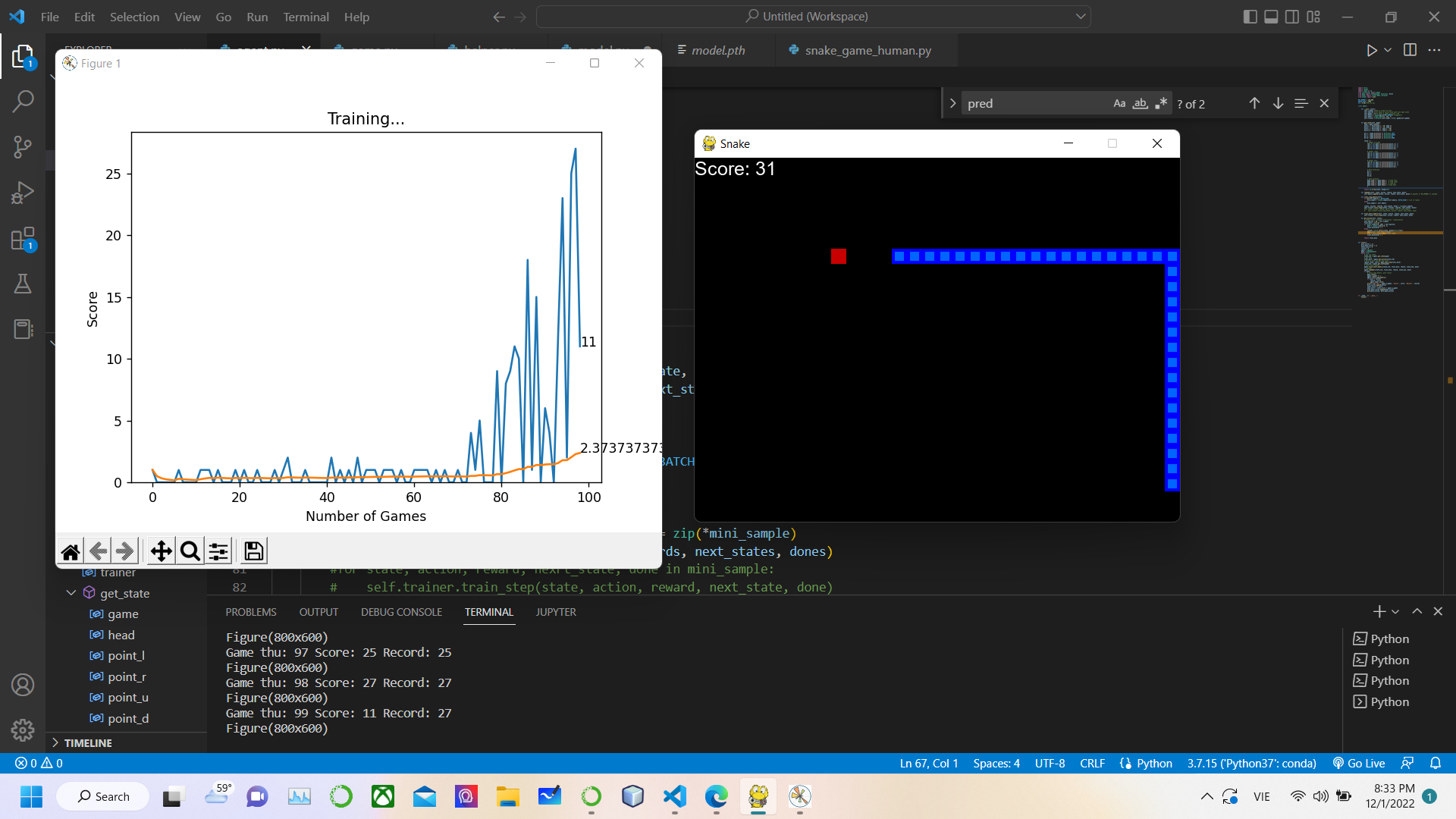
* biểu diễn quá trình học của Agent qua matplotlib



Kết quả:



Những game đầu tiên điểm số của con rắn chỉ là 1,2



Từ game 80 trở đi điểm của con rắn đã cao lên