

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

-----🙞🙜🕮🙞🙜-----



**BÁO CÁO BÀI TẬP LỚN**

**MÔN: LẬP TRÌNH VỚI PYTHON**

**Chủ đề : Trò chơi Rắn săn mồi**

**Giảng viên giảng dạy : Cô Võ Hoài Thư**

**Nhóm sinh viên thực hiện : Nhóm 24**

Nguyễn Hoàng Anh B19DCCN023

Nguyễn Quốc Bảo B19DCCN059

Hoàng Vân Anh B19DCCN016

Hà Nội – 2021

**MỤC LỤC**

[**I. GIỚI THIỆU** 3](#_heading=h.gjdgxs)

[**1. Ý tưởng về trò chơi** 3](#_heading=h.30j0zll)

[**2. Phân tích** 3](#_heading=h.1fob9te)

[**3. Luật chơi** 3](#_heading=h.3znysh7)

[3.1. Luật chơi 3](#_heading=h.2et92p0)

[3.2. Điểm: 3](#_heading=h.tyjcwt)

[**4. Phân tích chức năng của game** 3](#_heading=h.3dy6vkm)

[4.1 Chức năng “Play game”: 4](#_heading=h.1t3h5sf)

[4.2 Chức năng “Difficulty”: 4](#_heading=h.4d34og8)

[4.3 Chức năng “Rank”: 4](#_heading=h.2s8eyo1)

[4.4 Chức năng “Credit” 4](#_heading=h.17dp8vu)

[4.5 Chức năng “Exit” 4](#_heading=h.3rdcrjn)

[**II. Ý TƯỞNG CHƯƠNG TRÌNH** 5](#_heading=h.26in1rg)

[**1. Con rắn** 5](#_heading=h.lnxbz9)

[**2. Di chuyển rắn** 5](#_heading=h.35nkun2)

[**3. Thức ăn** 5](#_heading=h.1ksv4uv)

[**4. Va chạm ( Rắn tự va vào mình hoặc va chạm vật cản)** 5](#_heading=h.44sinio)

[**5. Điểm số** 5](#_heading=h.2jxsxqh)

[**6. Vector2 trong thư viện pygame.math** 5](#_heading=h.z337ya)

[**III. CÀI ĐẶT CHƯƠNG TRÌNH** 6](#_heading=h.3j2qqm3)

[**1. Tạo cửa sổ trò chơi** 6](#_heading=h.1y810tw)

[**2. Tạo map** 6](#_heading=h.4i7ojhp)

[**3. Tạo thức ăn** 6](#_heading=h.2xcytpi)

[**4. Tạo và vẽ con rắn** 7](#_heading=h.1ci93xb)

[4.1 Khởi tạo con rắn. 7](#_heading=h.3whwml4)

[4.2 Vẽ con rắn : 8](#_heading=h.2bn6wsx)

[**5. Di chuyển con rắn** 9](#_heading=h.qsh70q)

[**6. Rắn ăn thức ăn** 10](#_heading=h.3as4poj)

[**7. Va chạm** 11](#_heading=h.1pxezwc)

[**8. Điểm số** 11](#_heading=h.49x2ik5)

[**9. Button** 11](#_heading=h.2p2csry)

[**10. Thiết lập giao diện của trò chơi** 12](#_heading=h.147n2zr)

[10.1 Menu 12](#_heading=h.3o7alnk)

[10.2 Play 13](#_heading=h.23ckvvd)

[10.3 Difficulty 16](#_heading=h.ihv636)

[10.4 Rank 16](#_heading=h.32hioqz)

[10.5 Credit 16](#_heading=h.1hmsyys)

[**III. CÁC HÌNH ẢNH TRONG TRÒ CHƠI** 17](#_heading=h.41mghml)

[1. Giao diện menu chính 17](#_heading=h.2grqrue)

[2. Giao diện chơi game 18](#_heading=h.vx1227)

[3. Giao diện khi thua cuộc 19](#_heading=h.3fwokq0)

[4. Giao diện Difficulty 20](#_heading=h.1v1yuxt)

[5. Giao diện Rank 21](#_heading=h.4f1mdlm)

[6. Giao diện Credit 22](#_heading=h.2u6wntf)

# **I. GIỚI THIỆU**

## **1. Ý tưởng về trò chơi**

Game Rắn săn mồi – Snake là một trò chơi kinh điển, xuất hiện trên hầu hết các điện thoại “cục gạch” của Nokia. Game không kén chọn người dùng, được nhiều lứa tuổi yêu thích. Với 4 phím di chuyển, người chơi chỉ cần khéo léo điều khiển rắn ăn các con mồi xuất hiện trên màn hình để nó dài ra mà không đụng vào tường hay cơ thể của nó.

Ở trò chơi của nhóm, người chơi có thể di chuyển con rắn để ăn thức ăn bằng cách sử dụng các phím di chuyển, con rắn đồng thời sẽ dài ra. Vị trí của rắn và thức ăn sữ xuất hiện ngẫu nhiên. Sau 4 lần ăn được thức ăn, tốc độ của rắn sẽ nhanh dần. Trò chơi sẽ kết thúc khi con rắn đâm vào tường, vật cản hoặc tự đâm vào mình.

## **2. Phân tích**

2.1. Màn hình game:

- Kích thước : width = 800, height = 600 (px).

- Được bố trí như một ma trận 40\*30 ô có kích thước mỗi ô là 20\*20.

2.2. Cấu tạo rắn:

- Con rắn khi khởi tạo có độ dài = 3 ô

2.3. Vật cản:

- Vật cản là bờ tường và tảng đá và được load từ ảnh có sẵn.

2.4. Thức ăn:

- Thức ăn (Nấm) được hiển thị ngẫu nhiên và không nằm trên bất kì vật cản nào.

- Kích thước của thức ăn bằng 1 ô, được load từ ảnh có sẵn.

## **3. Luật chơi**

### 3.1. Luật chơi

- Người chơi chọn nút PLAY để vào màn chơi. Rắn sẽ bắt đầu di chuyển. Người chơi di chuyển rắn bằng bằng phím sang phải , sang trái , lên trên , xuống dưới sao cho rắn ăn được thức ăn và không bị đâm vào vật cản hoặc chính mình.

- Rắn có thể đi vào bụi cây.

- Cứ sau khi ăn đủ 4 thức ăn, tốc độ của rắn tăng lên 1.

- Người chơi có thể tùy chọn mức độ khó ban đầu để thay đổi tốc độ ban đầu của rắn.

### 3.2. Điểm:

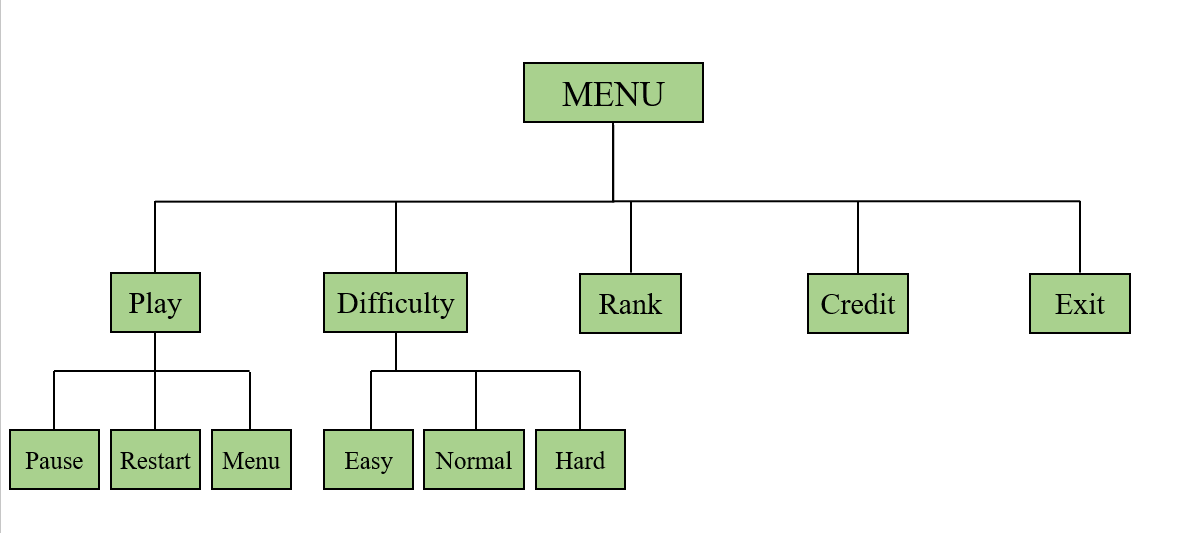
- Mỗi lần rắn ăn thức ăn được cộng 1 điểm

- Số điểm của người chơi được hiện ở góc trên bên trái màn hình game.

- Người chơi có thể xem top 5 điểm cao nhất.

## **4. Phân tích chức năng của game**

Sơ đồ :



### 4.1 Chức năng “Play game”:

Chức năng chính của chương trình, chương trình sẽ mở giao diện chơi game.

- Màn hình chính sẽ hiển thị con rắn, khoảng trắng rắn có thể di chuyển, vật cản, thức ăn, bụi cây.

- Khi đang chơi, người chơi bấm phím SPACE để Pause, game sẽ tạm dừng lại. Và bấm phím SPACE một lần nữa, game sẽ tiếp tục.

- Khi người chơi thua cuộc, game kết thúc. Màn hình hiển thị 1 bảng điều khiển bao gồm Restart và Menu:

+ Chọn Restart: game sẽ bắt đầu lại ở level mà người chơi đã chọn từ trước, số điểm trở về 0.

+ Chọn Menu : người chơi sẽ trở về giao diện Menu.

### 4.2 Chức năng “Difficulty”:

Chức năng để chọn độ khó khi trò chơi bắt đầu.

- Chương trình sẽ hiển thị 1 bảng lựa chọn bao gồm : Easy, Medium, Hard:

+ Easy : Mức độ dễ. Tốc độ ban đầu của rắn là 10.

+ Medium : Mức độ trung bình. Tốc độ ban đầu của rắn là 15.

+ Hard : Mức độ khó. Tốc độ ban đầu của rắn là 20.

- Khi người chơi lựa chọn mức độ, màn hình sẽ hiển thị lại mức độ mà người chơi đã chọn.

### 4.3 Chức năng “Rank”:

Chức năng xem top 5 điểm kỷ lục.

- Chương trình mở giao diện để xem điểm kỷ lục bao gồm số điểm của 5 người chơi cao nhất.

- Số điểm sẽ được sắp xếp từ cao đến thấp

- Mỗi khi kết thúc lượt game, hệ thống sẽ kiểm tra điểm của người chơi, nếu điểm kết thúc cao hơn ít nhất 1 số điểm trong top 5, hệ thống sẽ xóa số điểm đứng cuối top 10 và lưu lại số điểm người chơi vừa đạt được.

### 4.4 Chức năng “Credit”

Các thông tin về trò chơi

### 4.5 Chức năng “Exit”

Chức năng thoát ứng dụng game.

- Người chơi sẽ thoát khỏi ứng dụng game.

# **II. Ý TƯỞNG CHƯƠNG TRÌNH**

## **1. Con rắn**

- Sử dụng Vector2 trong thư viện pygame để lưu tọa độ x, tọa độ y của một phần của con rắn.

- Mỗi con rắn sẽ là một tập hợp các vector.

## **2. Di chuyển rắn**

Ví dụ : Để di chuyển con rắn sang phải, chúng ta di chuyển khối đầu sang phải. Sau đó chúng ta lấy khối trước đầu rắn và di chuyển đến vị trí trước đây của đầu và cứ thế đi qua toàn bộ con rắn.

Để di chuyển con rắn, khối đầu rắn sẽ di chuyển tới vị trí đầu mới. Khối sau đầu sẽ di chuyển đến đến vị trí trước của đầu. Các khối sau sẽ lần lượt di chuyển đến vị trí của khối trước đó.

## **3. Thức ăn**

Thức ăn được tạo ngẫu nhiên nhờ vào việc sử dụng thư viện random và có kiểm tra sao cho không trùng vào các vị trí của vật cản. Khi vị trí của thức ăn trùng với vị trí của đầu rắn, nghĩa là con rắn ăn thức ăn, và vị trí của thức ăn sẽ được thay đổi.

## **4. Va chạm ( Rắn tự va vào mình hoặc va chạm vật cản)**

Để kiểm tra xem rắn có tự va vào mình không, chúng ta kiểm tra xem khối đầu của rắn có trùng với bất kỳ khối nào khác con rắn hay không.

Tương tự, để kiểm tra xem rắn có va vào vật cản không, chúng ta kiểm tra xem khối đầu của rắn của trùng với vị trí của các vật cản hay không.

## **5. Điểm số**

Khi vị trí của khối đầu rắn trùng với vị trí của thức ăn, điểm số sẽ tăng lên 1. Khi người chơi thua, màn hình sẽ hiện số điểm mà người chơi đạt được và nhập số điểm vào file điểm số.

Để sắp xếp lại những điểm số cao nhất, dùng một hàm để mở file và sắp xếp lại.

Để hiển thị những điểm số cao nhất, mở file điểm đã lưu sẵn.

## **6. Vector2 trong thư viện pygame.math**

Chương trình sẽ dùng vector2 trong pygame.math , dưới đây sẽ giới thiệu qua về vector2.

+ Sử dụng Vector2 trong thư viện pygame thay vì sử dụng danh sách để thuận tiện trong việc xử lý dữ liệu về tọa độ trong trò chơi. Chẳng hạn như tại một điểm có tọa độ (10, 2).

| v = Vector2(10, 2)  v.x 10  v.y 2  Di chuyển sang phải:  r = Vector2(1, 0) , v +=r | v = [10,2]  v[x] 10  v.[y] 2  Di chuyển sang phải:  v[x]+=1 |
| --- | --- |

+ Dễ dàng trong việc xác định phương hướng giữa 2 khối:

Ví dụ, chúng ta có 2 khối có tọa độ lần lượt là a = (2, 10) và b = (3, 10). Lấy tọa độ của khối a trừ cho b, ta được 1 vector có tọa độ (-1, 0) , nghĩa là khối khối a ở bên trái của khối b. Tương tự, nếu hiệu 2 vector cho kết quả (1,0); (0, 1); (0, -1), chúng ta sẽ biết được các khối a lần lượt ở phải, dưới, trên so với khối b.

+ Dễ dàng trong việc xác định quan hệ phương hướng giữa 3 khối:

(4, 10); (5, 10); (6,10) : cùng y => Ba khối xếp thẳng đứng.

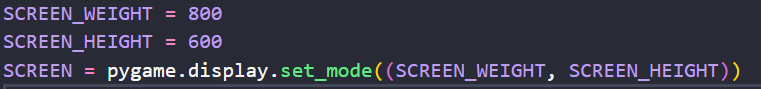
(4, 10); (4, 11); (4, 12): cùng x => Ba khối xếp nằm ngang.

# **III. CÀI ĐẶT CHƯƠNG TRÌNH**

Chương trình gồm 5 module : Bien.py , Event.py, Food.py, Object.py, Snake.py và 1 phần main : Snake.py.

* Bien.py : Lưu trữ các thông số, các biến được sử dụng ở chương trình để có thể tiếp cận được đến đối tượng được khai báo bất cứ lúc nào.
* Snake.py : Module Snake chứa class Snake. Class này chứa thông tin về con rắn bao gồm : khởi tạo con rắn lúc bắt đầu, di chuyển.
* Food.py : Module Food chứa class Food. Class này chứa thông tin về thức ăn bao gồm : tạo vị trí của thức ăn, hiển thị thức ăn, kiểm tra thức ăn không trùng với các vật cản.
* Object.py : Module Object chứa class Button, hình ảnh, âm thanh liên quan đến button được sử dụng trong trò chơi.
* Event.py : Module Event chứa class PlayGame là class chứa các hoạt động chính của trò chơi

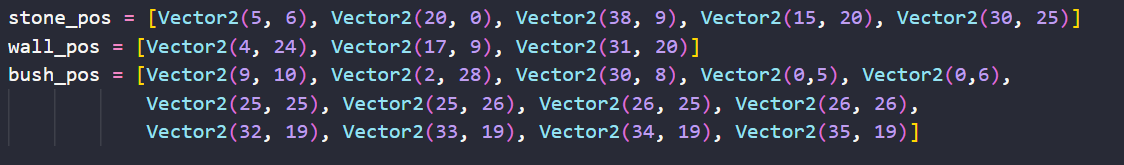
**1. Tạo cửa sổ trò chơi**



*Module Bien.py 1*

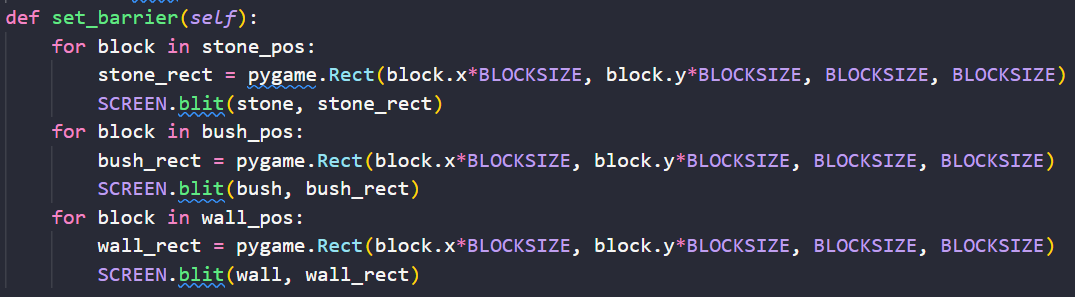
**2. Tạo map**

Trong map bao gồm đá, tường và bụi cây.



*Module Bien.py 2*

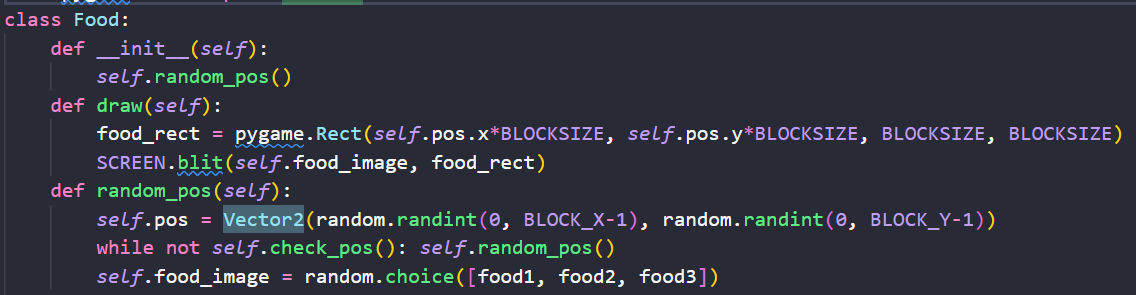
Tạo hàm set\_barrier().



*Module Event.py 1*

**3. Tạo thức ăn**

Tạo một class Food.

 Ở trong pygame, khi self.pos.x = 5, pygame sẽ đặt nó ở vị trí 5 pixel từ bên trái. Nhưng chúng ta lại muốn nở ở vị trí gấp 5 lần khối mà chúng ta quy định sẵn. Vì vậy phải thêm 1 đơn vị là BLOCKSIZE là kích thước ô, sau đó nhân với vị trí self.pos.x.

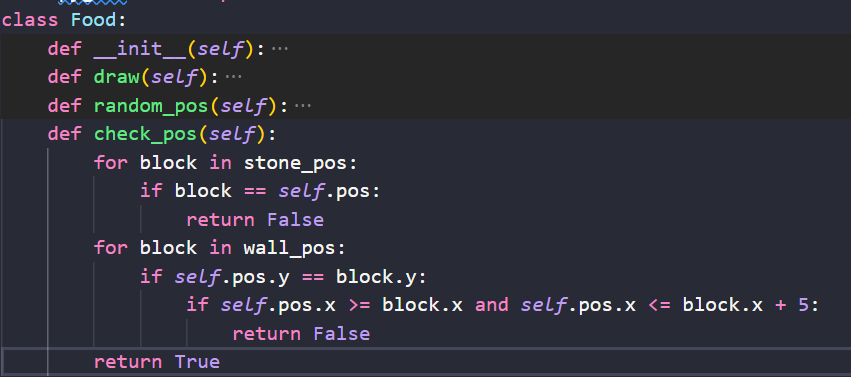


Sử dụng thư viện random để tạo ra các giá trị ngẫu nhiên. Dùng randint để tạo ra số nguyên ngẫu nhiên từ giá trị này sang giá trị khác. Ở tọa độ x và y, tọa độ sẽ bắt đầu từ 0 đến vị trí mép của khung (là BLOCK\_X và BLOCK\_Y đã quy định sẵn)



Hình ảnh thức ăn cũng sẽ được chọn ngẫu nhiên từ 3 hình ảnh đã được thiết lập sẵn bằng việc sử dụng hàm choice trong thư viện random.

Bên cạnh đó, thức ăn phải không trùng vào vị trí vật cản. Hàm check\_pos để kiểm tra điều này.



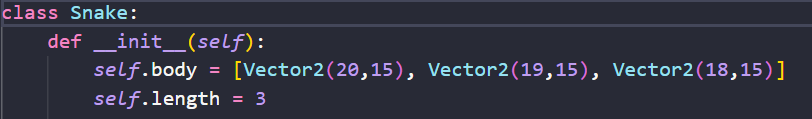
## **4. Tạo và vẽ con rắn**

Tạo một class Snake

Việc tạo con rắn cũng tương tự như tạo thức ăn, chỉ khác việc chúng ta phải tạo ra nhiều khối và liên kết các khối với nhau.

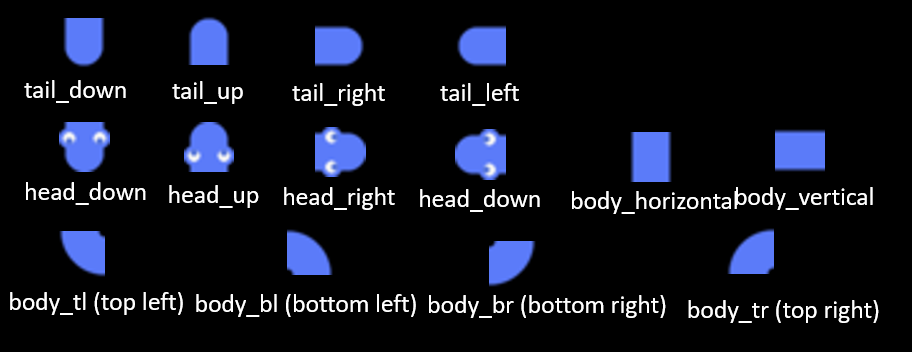
### 4.1 Khởi tạo con rắn.

Độ dài ban đầu là 3.

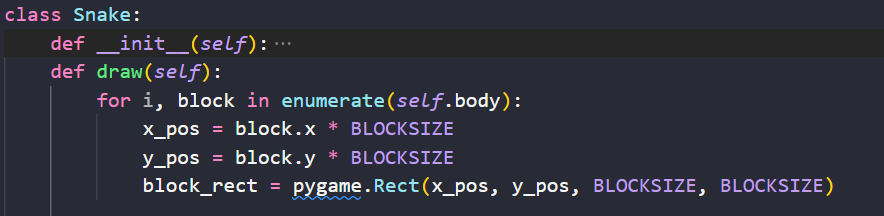


### 4.2 Vẽ con rắn :

Để vẽ được một con rắn, sử dụng các hình ảnh có sẵn và tìm đúng vị trí để ghép vào.



* Trước tiên, cần có 1 cơ sở là tọa độ của các khối của con rắn.



* Vẽ con rắn dựa vào từng vị trí.

*(Tiếp tục đoạn code trên)*



Vòng lặp for chạy hết toàn bộ con rắn, chỉ mục i cho biết đang chạy đến phần nào của con rắn:

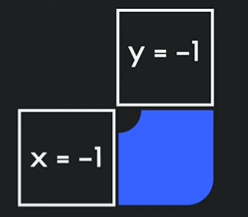
+ Nếu i = 0, tức là đang duyệt ở phần đầu của rắn, chương trình sẽ vẽ ra phần đầu của rắn.

+ Nếu i = chiều dài của rắn -1 , tức là đang duyệt ở phần đuôi của rắn, chương trình sẽ vẽ ra phần đuôi của rắn.

+ Ở các phần tiếp theo của con rắn. Cần tìm ra khối tiếp theo (after\_block) và khối trước (before\_block) của khối hiện tại.

Để xác định mối quan hệ giữa hai khối trước before\_block và khối sau after\_block đối với khối hiện tại, chúng ta lấy tọa độ của vector của khối trước/ khối sau trừ đi tọa độ của vector của khối hiện tại.

Nếu before\_block.x = after\_block.x , ba khối xếp nằm ngang với nhau, chương trình sẽ vẽ ra khối nằm ngang.

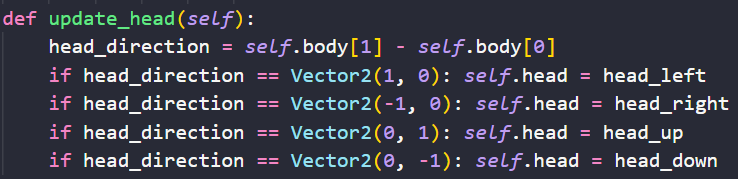
Nếu before\_block.y = after\_block.y , ba khối xếp thẳng đứng với nhau, chương trình sẽ vẽ ra khối thẳng đứng.

Nếu before\_block.x = -1 và after\_block.y = -1 hoặc before\_block.y = -1 và after\_block.x = -1, nghĩa là con rắn đang đi sang phải và đi lên hoặc đi xuống và sang trái.

Tương tự với 3 góc còn lại.

* Xác định hướng của đầu rắn

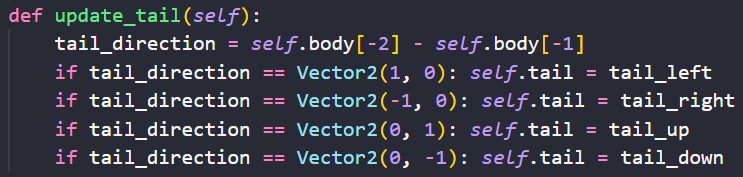
Để xác định hướng của đầu rắn, chúng ta lấy tọa độ của khối ngay sau khối đầu trừ đi tọa độ của khối đầu để xem mối quan hệ giữa hai khối. Phép trừ này sẽ trả về một vector phương hướng, dựa vào đó để xác định hướng của đầu rắn.



*Module Bien.py 3*

* Xác định hướng của đuôi rắn

Tương tự như đầu rắn, để xác định hướng của đuôi rắn, chúng ta lấy tọa độ của khối ngay trước đuôi trừ đi tọa độ của khối đuôi để xem mối quan hệ giữa hai khối.

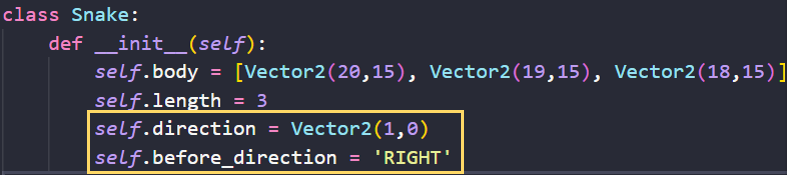


Sau đó thêm 2 hàm update\_head() và update\_tail() vào phần hàm draw()

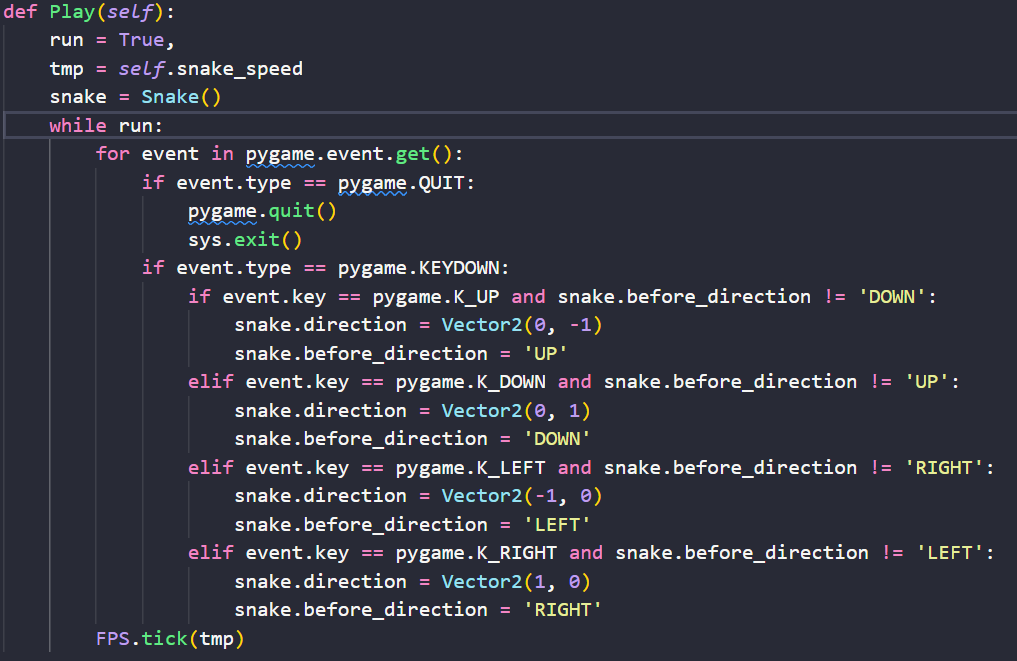
## **5. Di chuyển con rắn**



Để di chuyển con rắn, chèn khối vào vị trí ngay sau đầu của con rắn. Trong đó, vector direction được quy định khi người chơi tùy theo input của người chơi. Vector direction và hướng trước đó được thiết lập sẵn khi trò chơi bắt đầu.



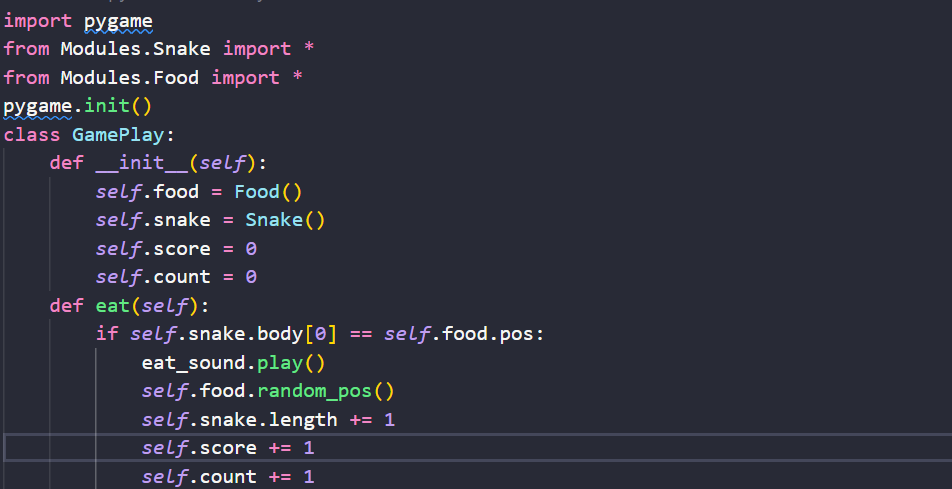
Khi người chơi thao tác di chuyển con rắn:



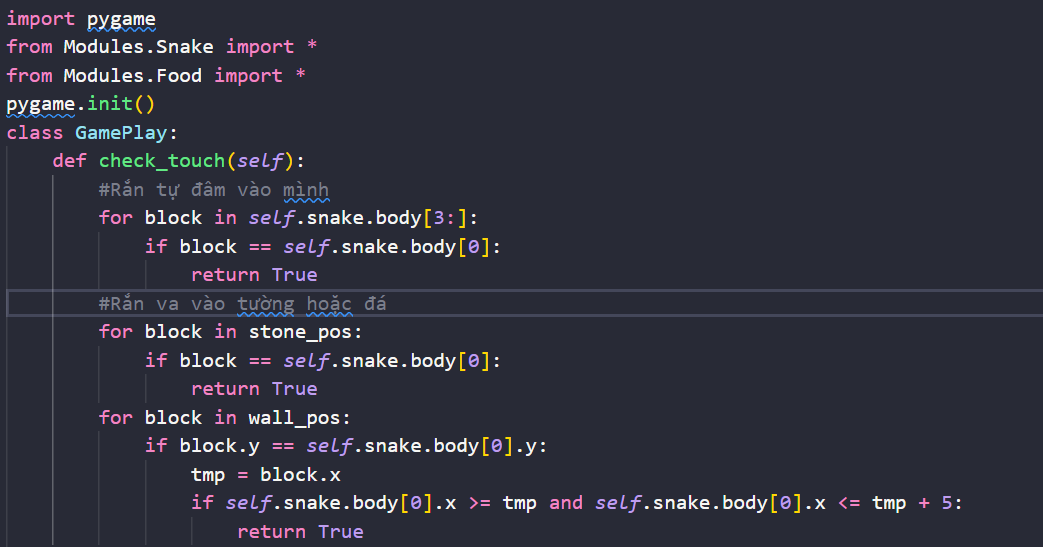
*File Game.py 1*

## **6. Rắn ăn thức ăn**

Tạo một class PlayGame trong Module Event.py

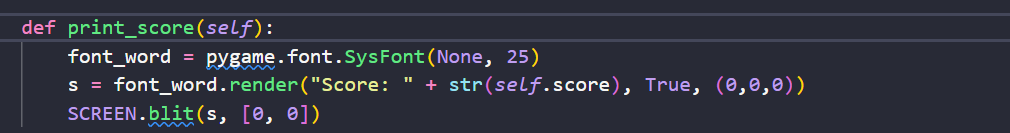


## **7. Va chạm**

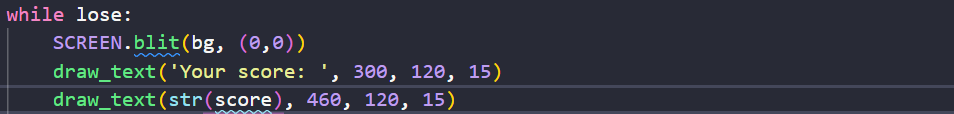


## **8. Điểm số**

Khi người chơi đang chơi, điểm số của người chơi sẽ được hiện ở góc trên trái của màn hình.

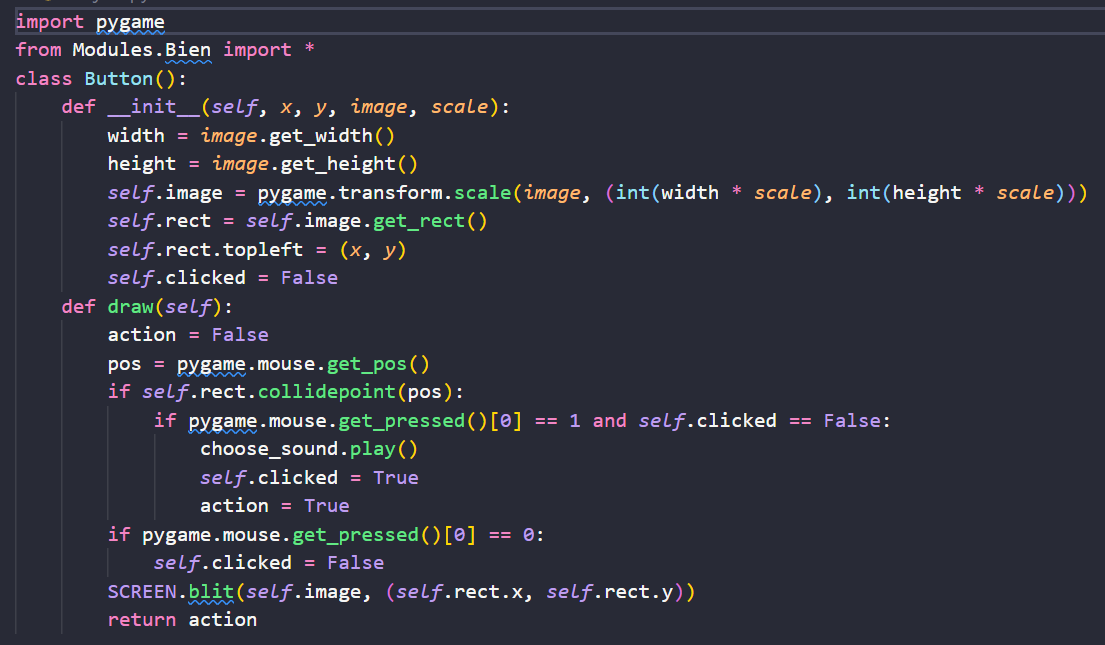


Khi trò chơi kết thúc, số điểm của người chơi cũng sẽ được hiện ra.



## **9. Button**

Tạo một module Object.py.



Hàm \_\_init\_\_() truyền vào các tham số x, y, image, scale lần lượt là tọa độ x , tọa độ y, hình ảnh và tỉ lệ ảnh của button. Tỉ lệ ảnh để có thể tủy chỉnh kích thước ảnh sao cho phù hợp với ý muốn. Ví dụ về một button :

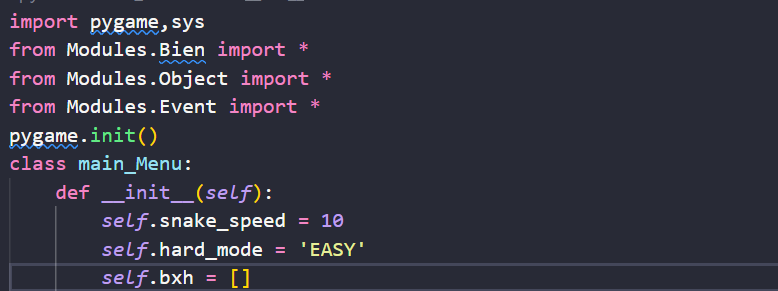


Ở hàm draw, kiểm tra nếu người chơi click vào button. Cờ action ban đầu là False. Nếu người chơi click vào vị trí của button, cờ action là True, và thực hiện chức năng của button. Ví dụ :

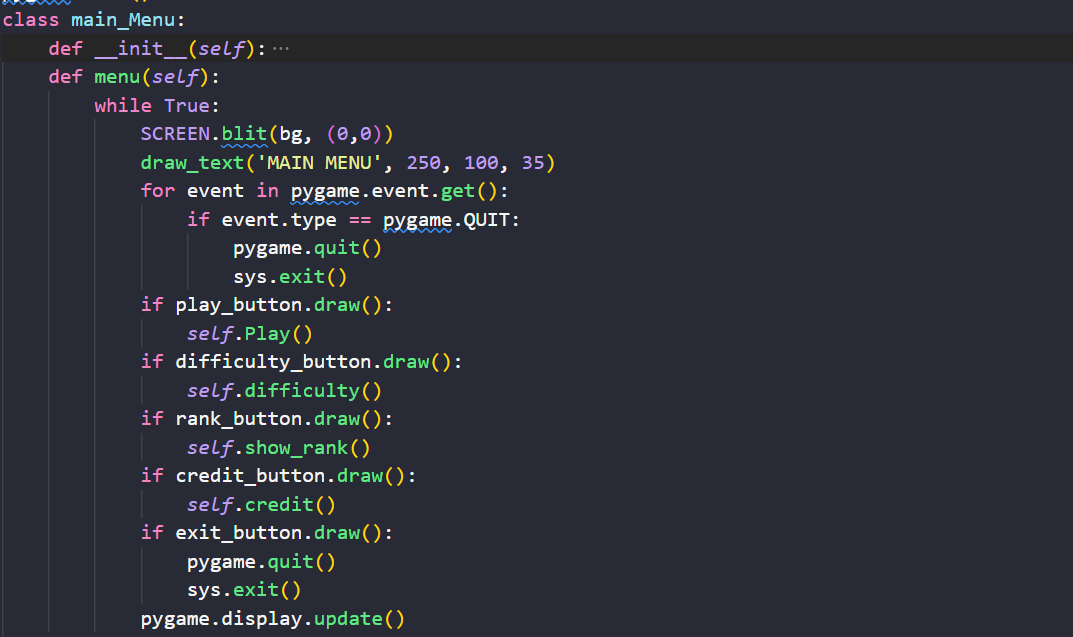


## **10. Thiết lập giao diện của trò chơi**

*(File Game.py)*

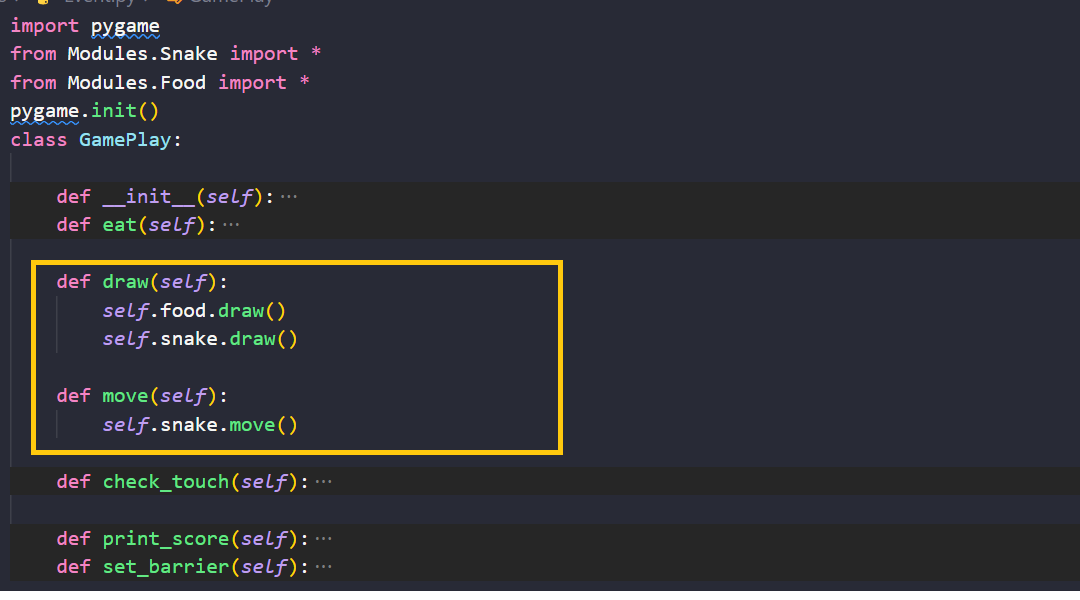
Ở class main\_Menu , giao diện chính của trò chơi, chúng ta khởi tạo các giá trị ban đầu 

### 10.1 Menu



### 10.2 Play

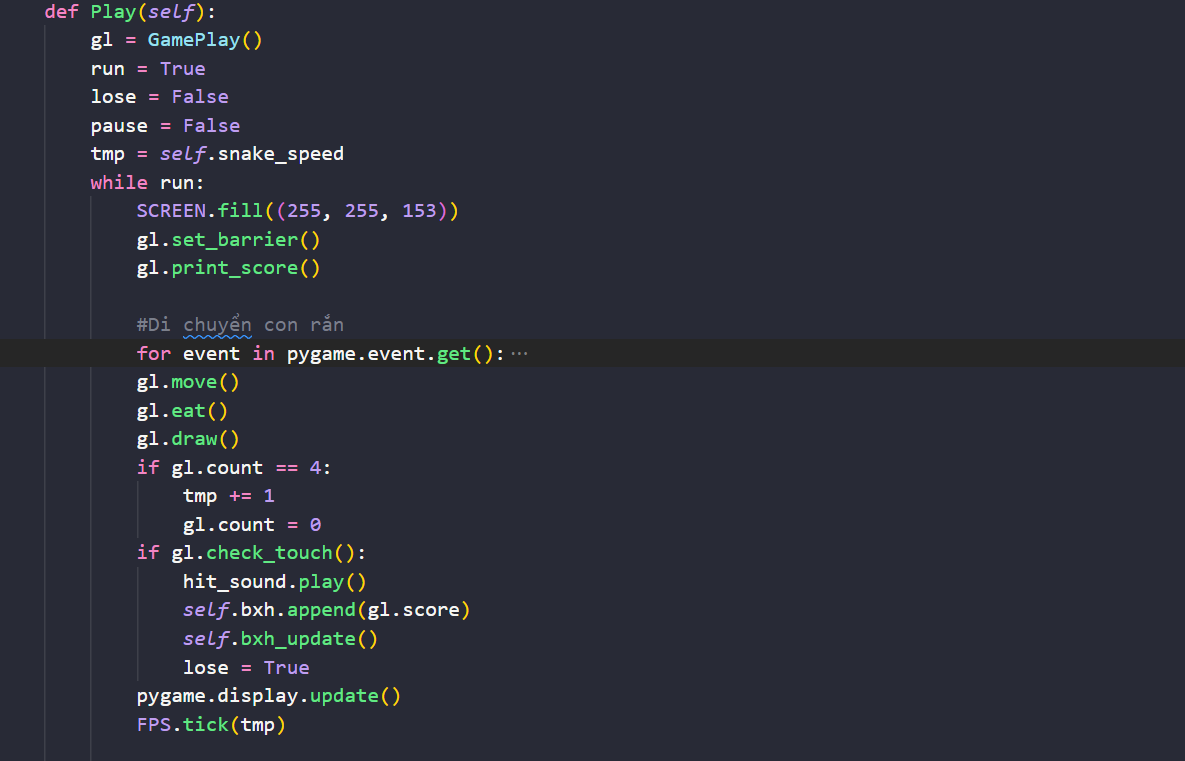
Ở class PlayGame trong module Event.py, trong phần \_\_init\_\_() đã bao gồm cả food và snake, vì vậy sẽ tiện lợi hơn nếu gọi đối tượng thuộc class GamePlay. Thiết lập thêm một số hàm cần thiết vào class PlassGame.



*Module Event.py 2*

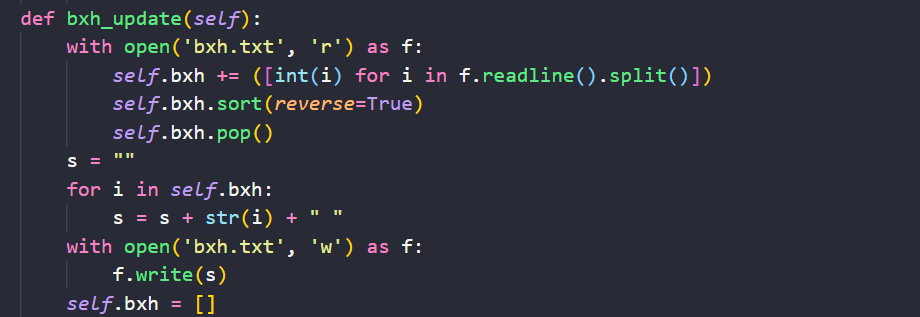
Các cờ run, lose, pause để làm cờ cho các thao tác trong game.

* run:

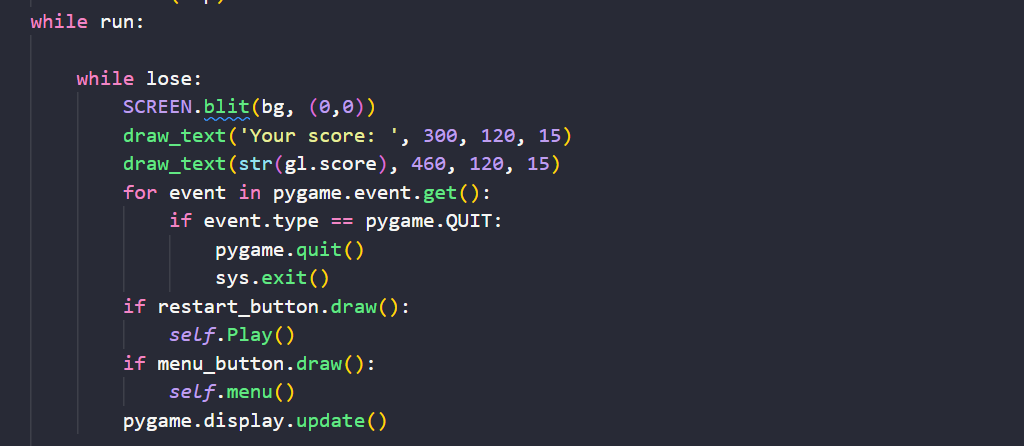


Ở câu lệnh *if gl.check\_touch()*, nếu trả về giá trị True, số điểm của người chơi sẽ được ghi vào file *bxh.txt.* Và đồng thời, cờ lose thay đổi thành True, người chơi thua cuộc.

Hàm *bxh\_update()* thực viện nhiêm vụ sắp xếp lại và lưu 5 điểm cao nhất.

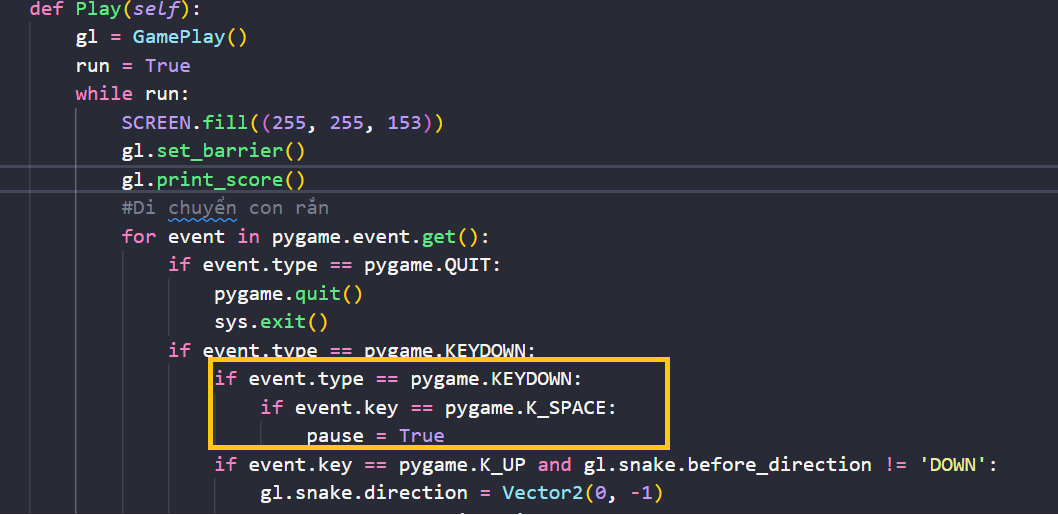


* lose:

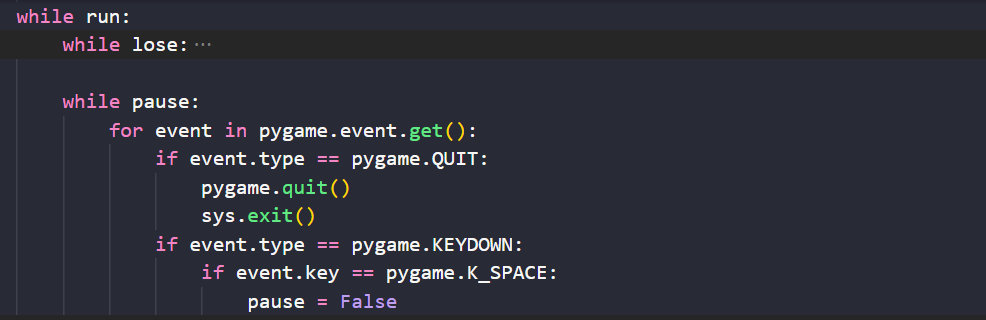


Khi người chơi thua cuộc, màn hình hiện ra giao diện bao gồm dòng chữ thông báo số điểm người chơi đã đạt được, button Restart và button Menu. Khi người chơi chọn button Restart, chương trình chạy lại hàm Play, người chơi tiếp tục chơi ở các thiết lập đã thiết lập từ màn trước, màn hình sẽ hiện màn chơi. Khi người chơi chọn button Menu, chương trình chạy lại từ đầu, màn hình sẽ hiện lại giao diện menu.

* pause:

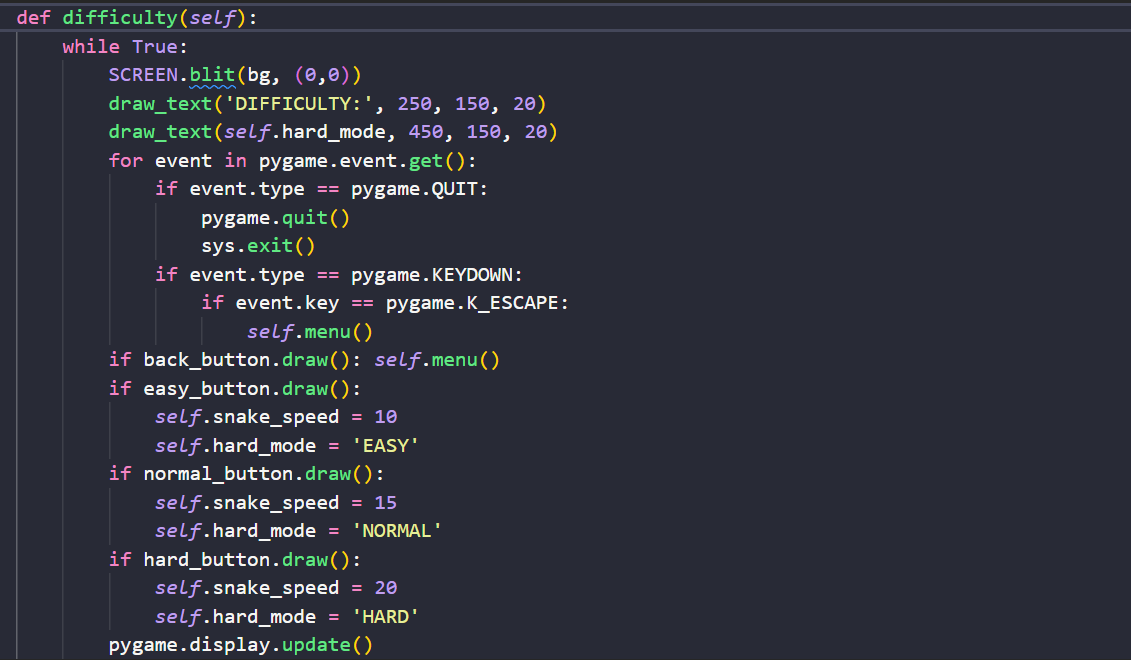


Khi đang chơi, người chơi nhấn phím SPACE để tạm dừng trò chơi. Cờ pause khi đó sẽ thành True. Người chơi nhấm phím SPACE lần nữa để tiếp tục trò chơi.



### 10.3 Difficulty

Khi người chơi chọn Difficulty, màn hình sẽ hiện ra giao diện Difficulty bao gồm 3 button : Easy, Normal, Hard và 1 button Back. Khi người chơi chọn các mức độ, tốc độ của con rắn cũng sẽ thay đổi tùy theo từng mức độ. Khi người chơi chọn button Back, màn hình sẽ trở lại giao diện menu chính.

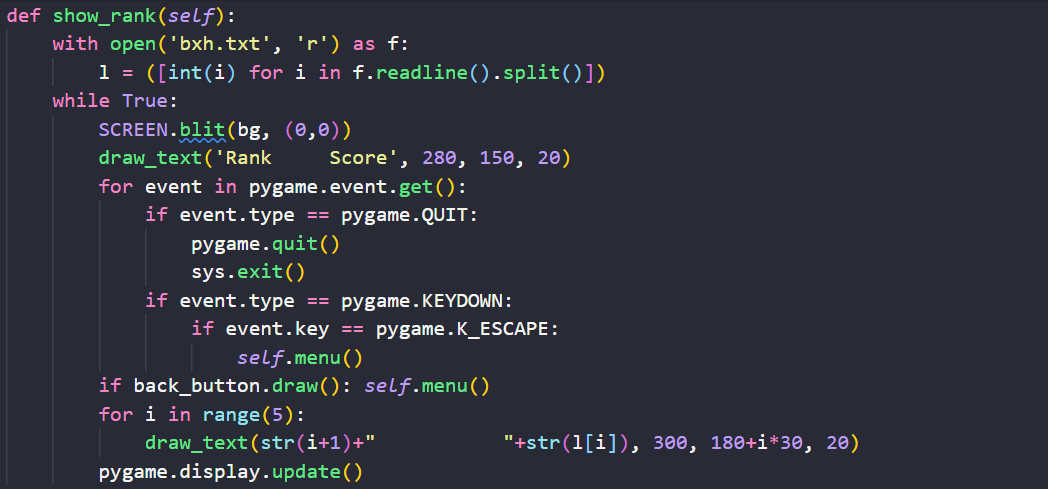


### 10.4 Rank

Khi người chơi chọn Rank, màn hình sẽ hiện ra giao diện Rank gồm 5 điểm cao nhất được sắp xếp theo thứ tự giảm dần và 1 button Back.

Để hiện ra 5 điểm số cao nhất, thực hiện thao tác mở file và dùng hàm draw\_text để lần lượt in ra các điểm số.

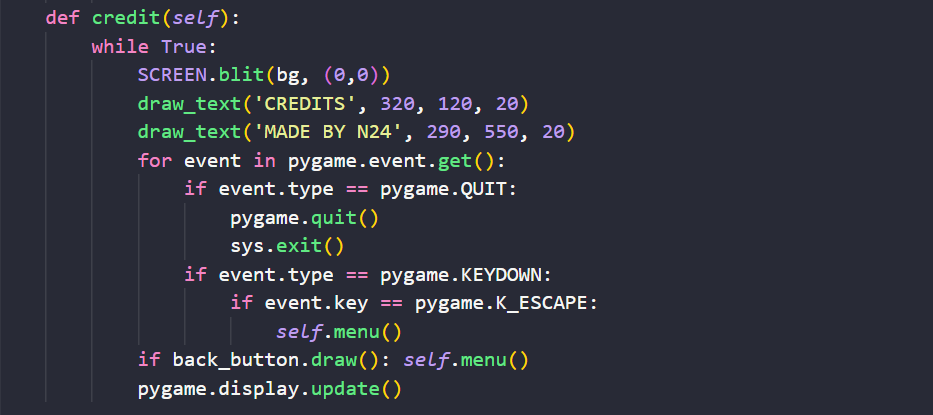
Khi người chơi chọn button Back, màn hình sẽ trở về giao diện menu chính.



### 10.5 Credit

Khi người chơi chọn Credit trong giao diện menu chính, màn hình sẽ hiện ra giao diện Credit gồm các thông tin của trò chơi và 1 button Back.

Khi người chơi chọn button Back, màn hình sẽ trở về giao diện menu chính.

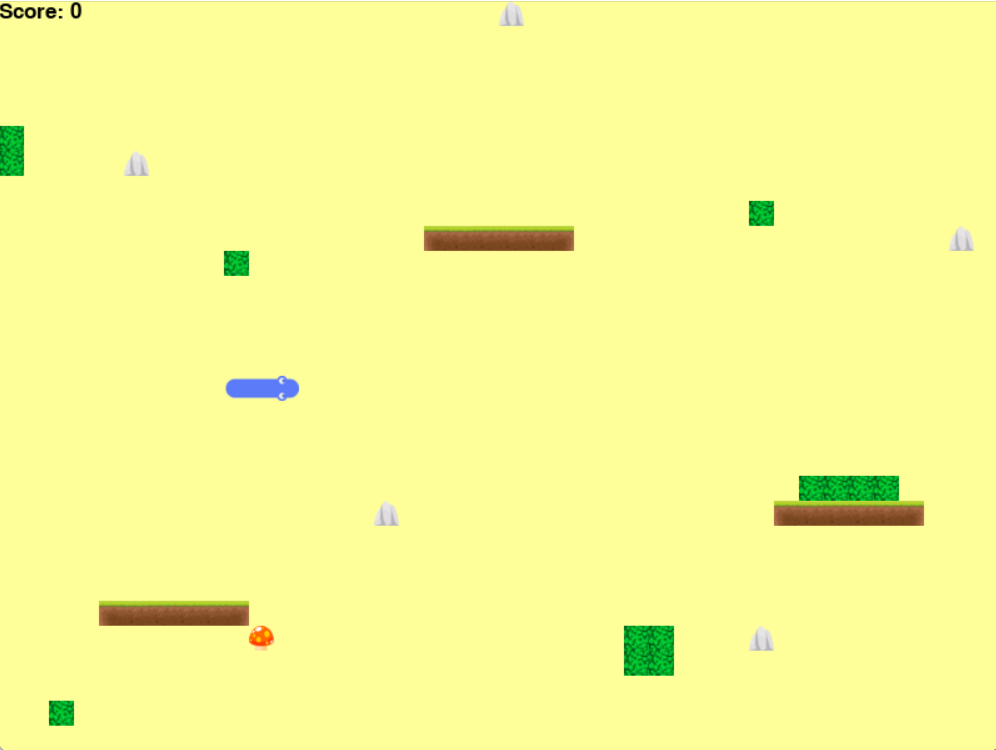


# **III. CÁC HÌNH ẢNH TRONG TRÒ CHƠI**

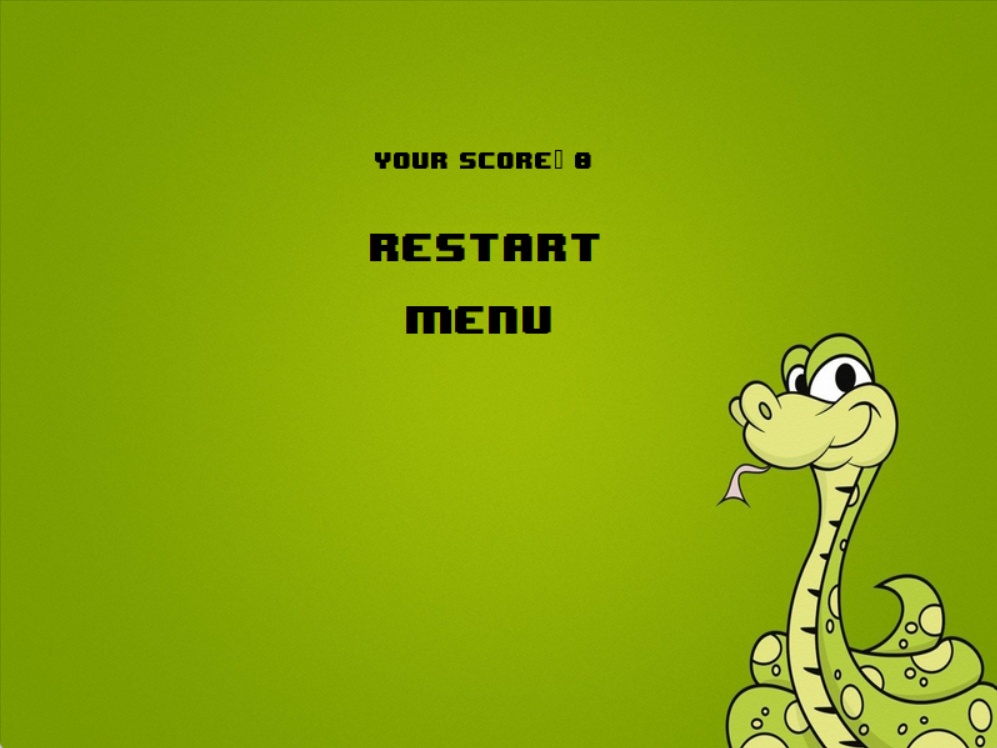
## 1. Giao diện menu chính



## 2. Giao diện chơi game



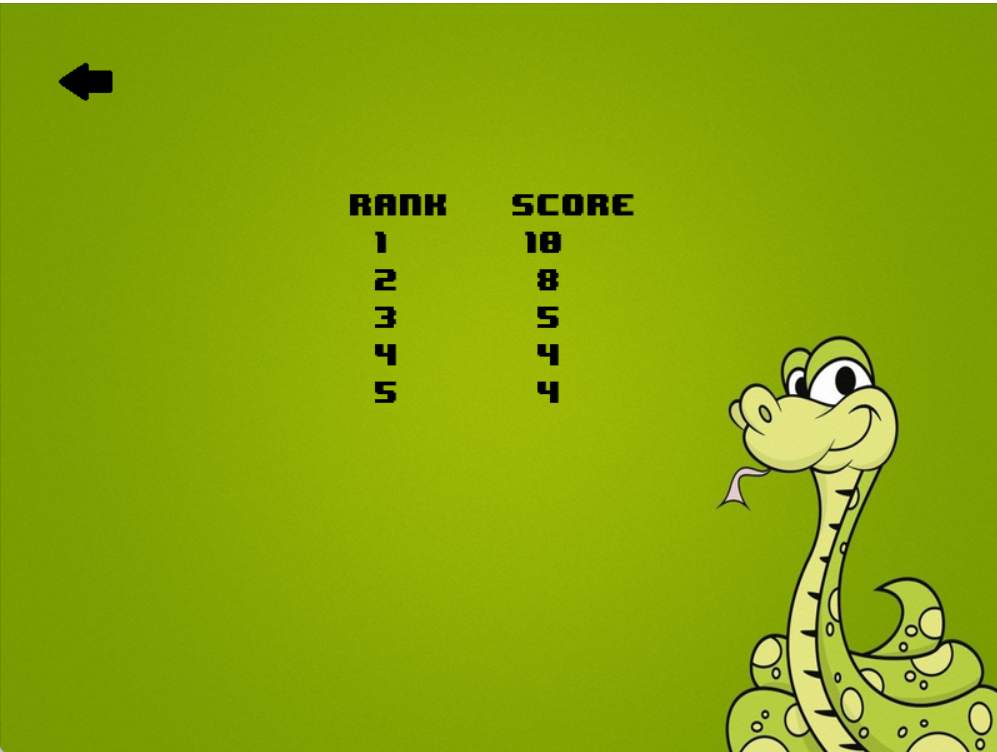
## 3. Giao diện khi thua cuộc



## 4. Giao diện Difficulty



## 5. Giao diện Rank



## 6. Giao diện Credit

