Logotipo

Descripción generada automáticamente

UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

FACULTAD DE INGENIERIA

SI404 - Inteligencia Artificial

Informe de TB 02

Rat in a Maze with Neural Network

Profesor:

Wester Edison Zela Moraya

Integrantes:

Alessandra Abad Mendizábal

Renato Tomasto Salvador

Ricardo Valencia Lafosse

2020-02

Contenido

[1. Descripcion y Fundamentación del Problema 3](#_Toc55700003)

[2. Descripción de la Base de Conocimiento 3](#_Toc55700004)

[3. Descripción de la arquitectura de Red Neuronal y describir la salida 4](#_Toc55700005)

[4. Código fuente de la aplicación 5](#_Toc55700006)

[Imports y Variables Globales 5](#_Toc55700007)

[Clases (Qmaze y Experience) 6](#_Toc55700008)

[Metodos 7](#_Toc55700009)

[Comandos de Ejecucion 9](#_Toc55700010)

[5. Pruebas de uso y ejecución de la aplicación 10](#_Toc55700011)

[6. Bibliografía 12](#_Toc55700012)

# Descripción y Fundamentación del Problema

Los rompecabezas de laberintos tradicionales se han utilizado mucho en la investigación y educación de estructuras de datos y algoritmos. El conocido algoritmo de ruta más corta de Dijkstra sigue siendo el método más práctico para resolver este tipo de acertijos, pero debido a su familiaridad y naturaleza intuitiva, estos acertijos son bastante buenos para demostrar y probar técnicas de Reiforcement Learning.

Un laberinto simple consta de una cuadrícula rectangular de celdas (generalmente cuadradas), una rata y una celda de "queso".

Para que sea lo más simple posible:

* Usaremos un laberinto pequeño de 7x7 como ejemplos.
* El queso siempre está en la celda inferior derecha del laberinto.
* Tenemos dos tipos de celdas: celdas libres (blancas) y celdas ocupadas (negras).
* La rata puede partir de cualquier celda libre (para este ejemplo, de la celda superior izquierda) y solo se le permite viajar por las celdas libres.

# Descripción de la Base de Conocimiento

El Reiforcement Learning es una técnica de aprendizaje automático para resolver problemas mediante un sistema de retroalimentación (recompensas y penalizaciones) aplicado a un agente que opera en un entorno y necesita pasar por una serie de estados para alcanzar un estado final predefinido.

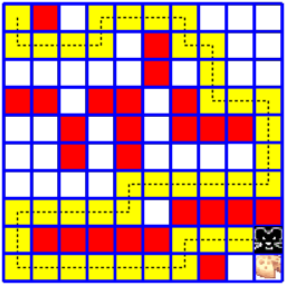
Un ejemplo clásico es una rata (agente) que está tratando de encontrar la ruta más corta desde una celda inicial hasta una celda de queso objetivo en un laberinto (entorno). El agente está experimentando y explotando experiencias pasadas (episode o una sesión de entrenamiento) para lograr su objetivo. Puede fallar una y otra vez, pero con suerte, después de muchas pruebas y errores (recompensas y sanciones) llegará a la solución del problema.

La solución se alcanzará si el agente encuentra la secuencia óptima de estados en los que la suma acumulada de recompensas es máxima (en resumen, atraemos al agente para que acumule una recompensa máxima y, al hacerlo, resuelve nuestro problema).

Hay que tener en cuenta que puede suceder que, para alcanzar la meta, el agente tendrá que soportar muchas penalizaciones (recompensas negativas) en su camino. Por ejemplo, la rata en el laberinto anterior recibe una pequeña penalización por cada movimiento legal. La razón de esto es que queremos que llegue a la celda objetivo en la ruta más corta posible. Sin embargo, el camino más corto hacia la celda de queso es a veces largo y sinuoso, y nuestro agente (la rata) puede tener que soportar muchas penalizaciones hasta llegar al "queso" (a veces llamado "recompensa retrasada").

Para el desarrollo de este proyecto se usará el lenguaje de programación Python 3.8, debido a que tiene compatibilidad con TensorFlow y las librerías Keras, que se utilizaran para manejar los modelos de redes neuronales, y además se utilizara pygame para poder graficar los modelos.

# Descripción de la arquitectura de Red Neuronal y describir la salida



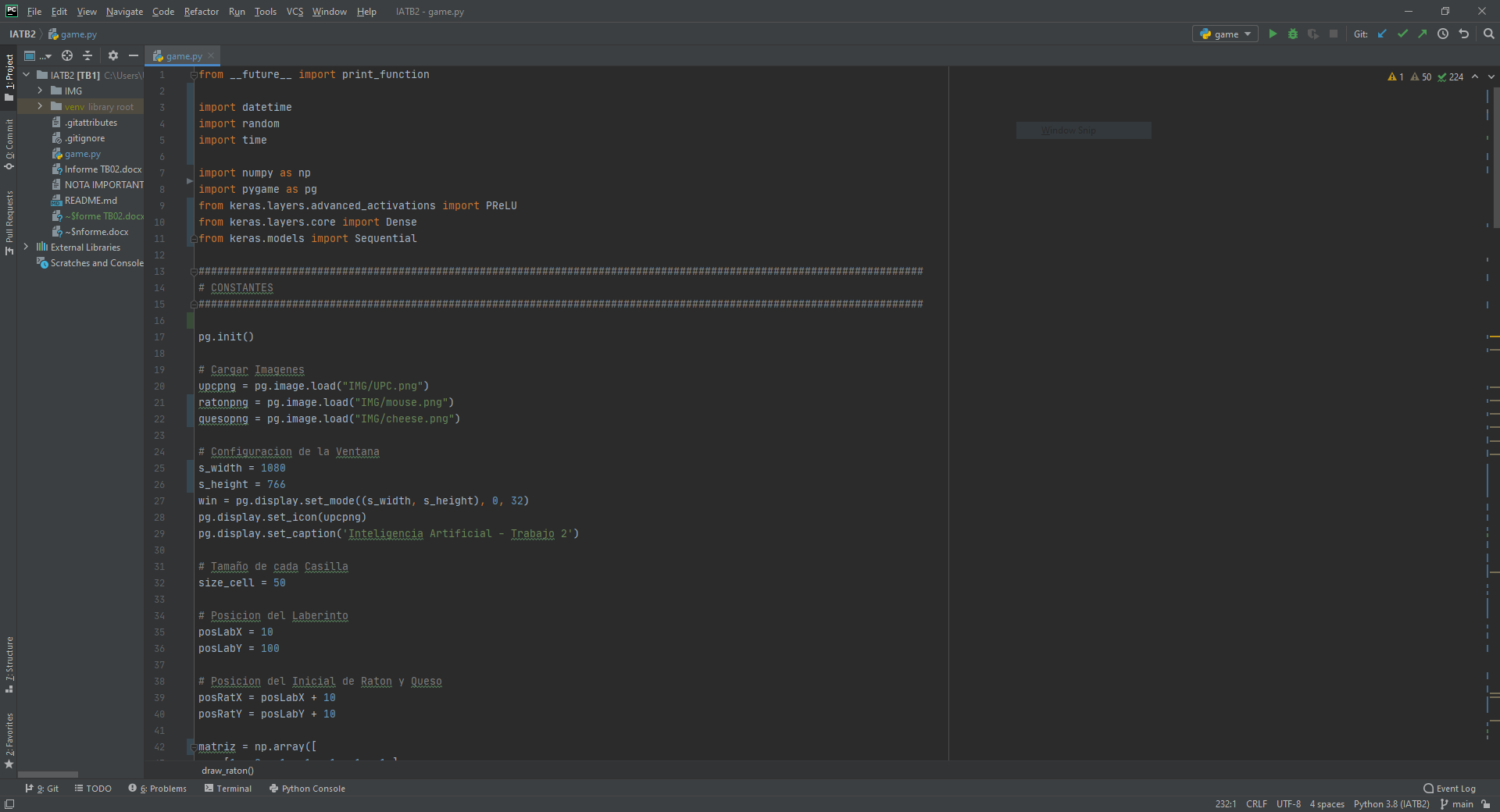
Se define una cantidad de Epochs, o sesiones de entrenamiento, para cada cual se propondrá un entorno nuevo libre de ningún cambio, y luego se realizarán una serie de intentos donde la rata tendrá que llegar a un queso.

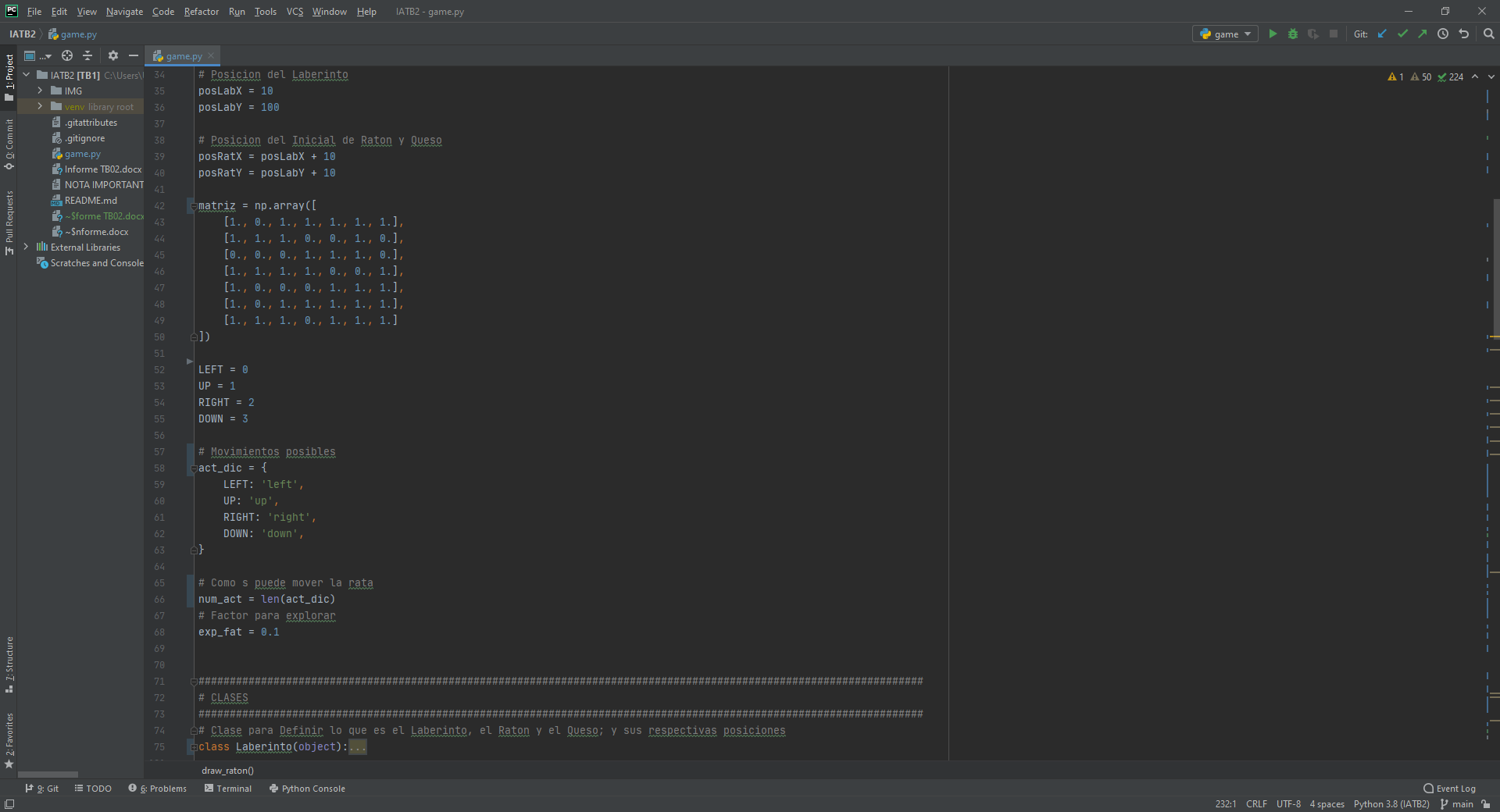
Al principio con movimientos aleatorios, que se irán registrando en un dataset, y que poco a poco podrán servir de guía para las futuras generaciones de ratas.

Debido a esto, el margen de Victorias por Intento será diminuto, por lo que la salida de cada Epoch inicial será posiblemente de 0.00, hasta que se llegue al máximo de 1.00 cuando la aplicación termine de correr.

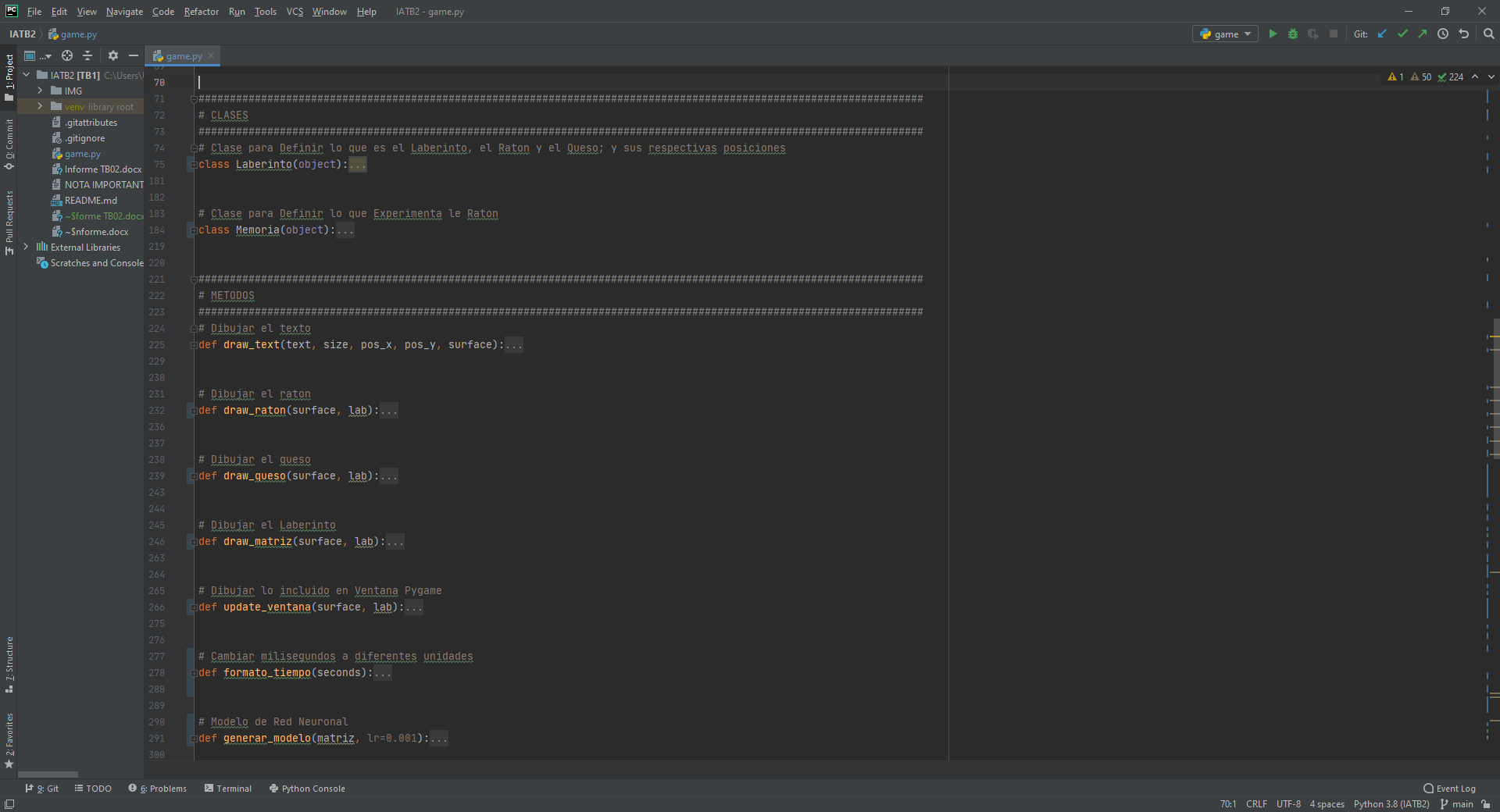
# Código fuente de la aplicación

## Imports y Variables Globales

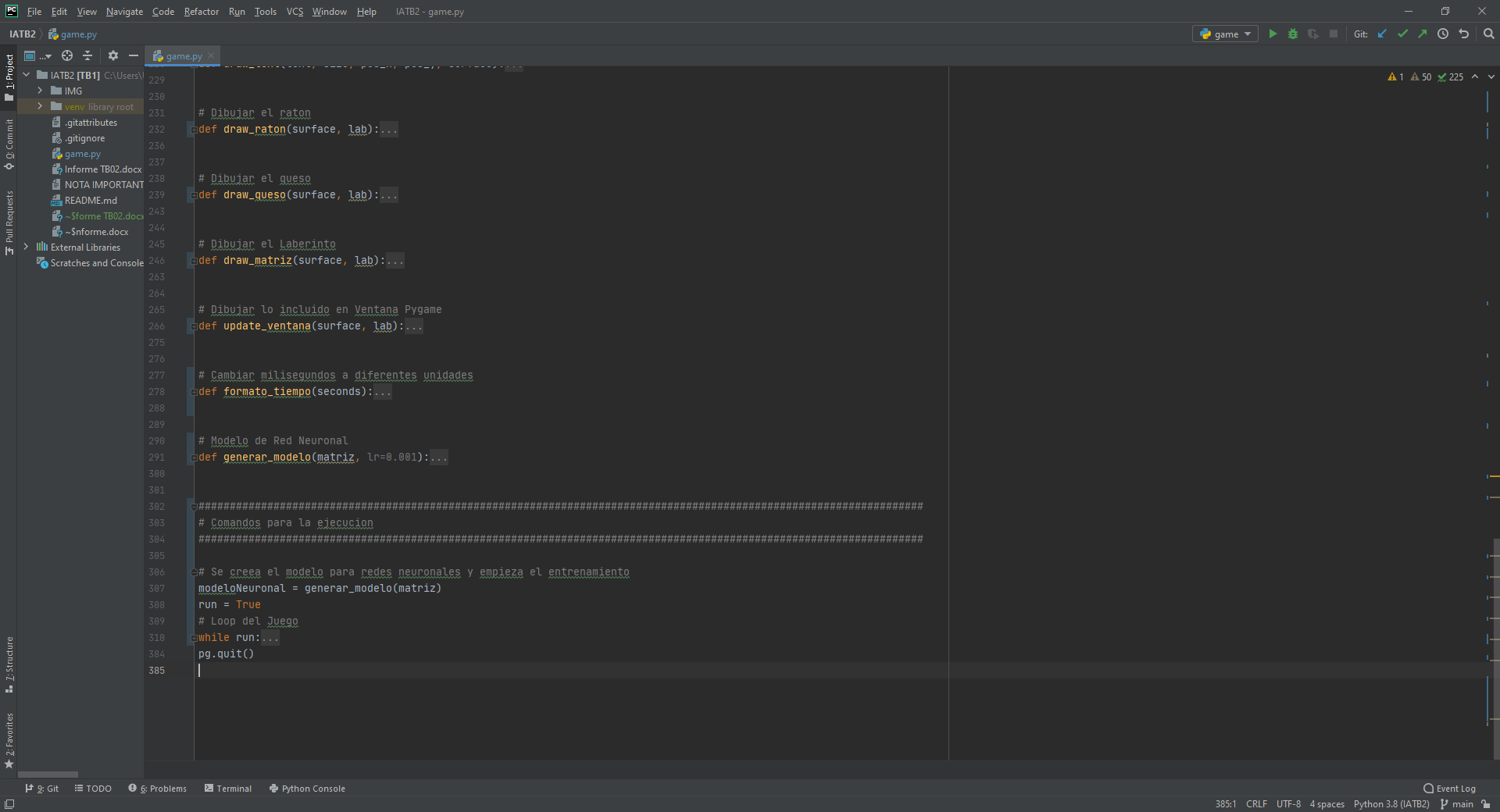




## Clases y Metodos



## Comandos de Ejecución



# Pruebas de uso y ejecución de la aplicación

Video de Youtube con DEMO de la aplicación: https://youtu.be/FNxyPtVK2s4



La aplicación acaba de empezar y muestra el primer entorno del laberinto. En consola existe un error debido a que el GPU donde se esta corriendo es un AMD, que es incompatible con el Cuda, y por ende advierte que no esta optimizado el uso de TensorFlow.



Los primeros entrenamientos (EPOCHS) tienen bastantes intentos fallidos de llegar al queso, debido a que no tienen aun datos para respaldar los posibles caminos ideales.

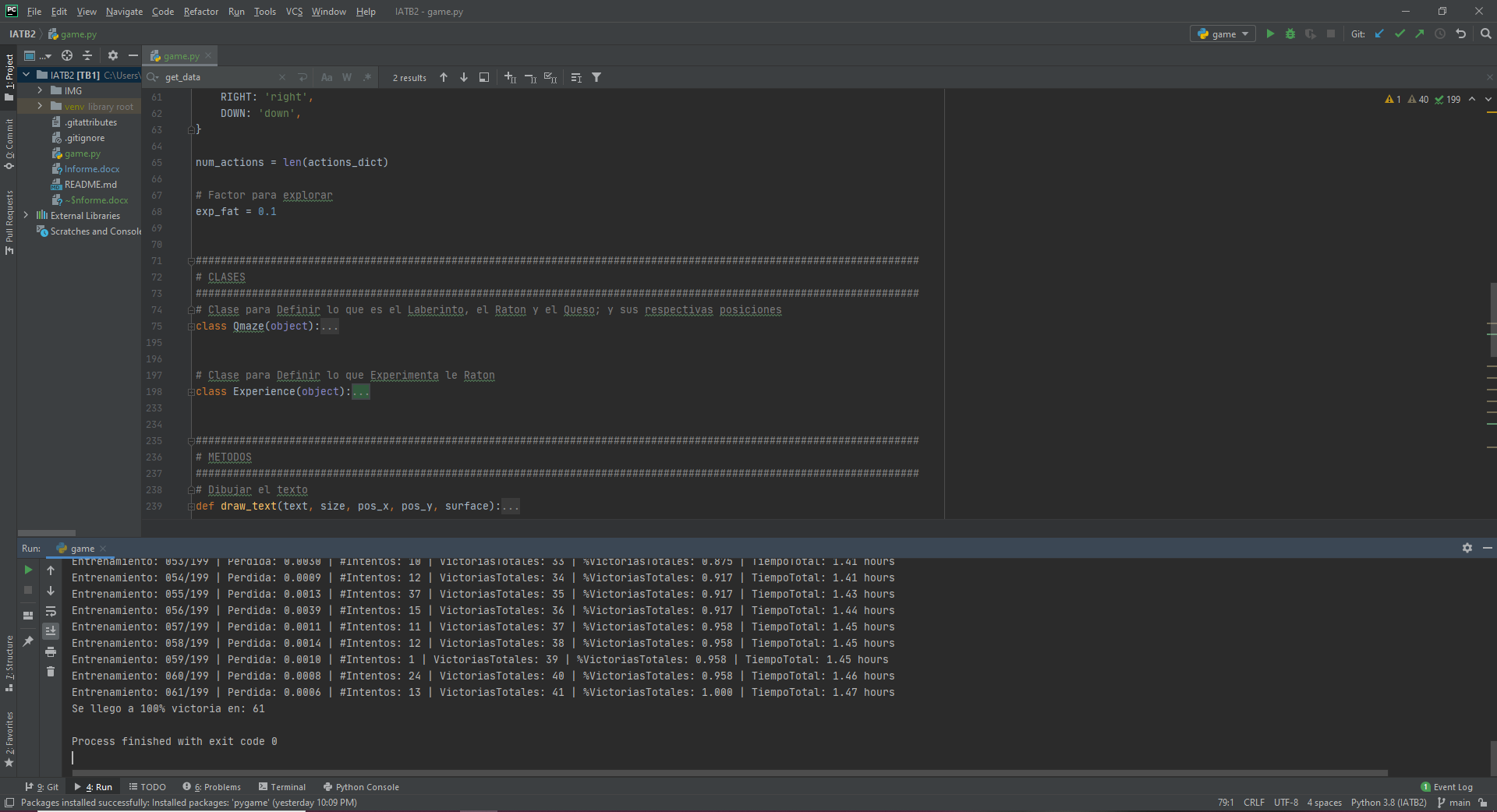
Se puede apreciar que usualmente cuando los números de intentos pasan los 100, se considera el entrenamiento fallido y se pasa a buscar otras alternativas de ruta.

Cuando Victorias totales aumenta, significa que se ha encontrado un camino hacia el queso, en el caso del entrenamiento 003, en el intento 59.



La aplicación ha estado corriendo durante 1 hora y 37 minutos y ya ha alcanzado 50% del aprendizaje necesario para poder llegar al queso directamente.

Se puede apreciar que él %VictoriasTotales está incrementando mucho más rápido que antes, debido a que las pruebas ya no tienen tantos intentos fallidos porque se van recordando los caminos correctos.



Finalmente, luego de 1 hora y 47 minutos, en el Entrenamiento 61 se llega al 100% de Fidelidad con respecto a las victorias totales, y se cierra la ventana.

# Bibliografía

* DEEP LEARNING INTRODUCCIÓN PRÁCTICA CON KERAS. https://torres.ai/deep-learning-inteligencia-artificial-keras/ (10 de octubre del 2020)
* Pygame Front Page. <https://www.pygame.org/docs/> (10 de octubre del 2020)