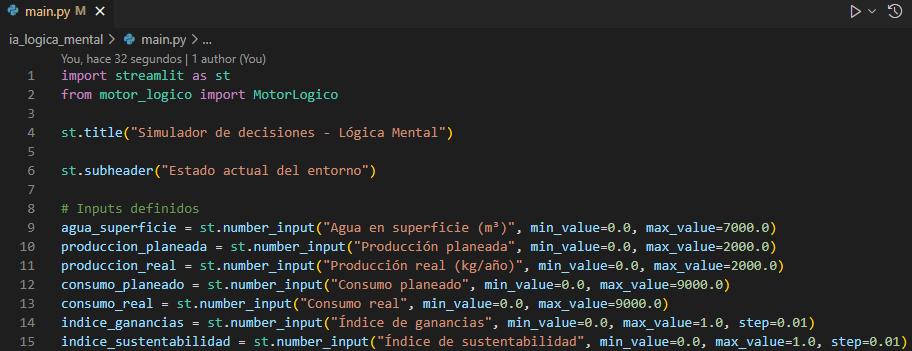
**1. Interfaz (main.py)**

El punto de entrada es una interfaz construida con **Streamlit**, que permite al usuario ingresar los valores actuales del entorno:

* Agua en superficie, producción planeada y real, consumo planeado y real
* Índice de ganancias e índice de sustentabilidad

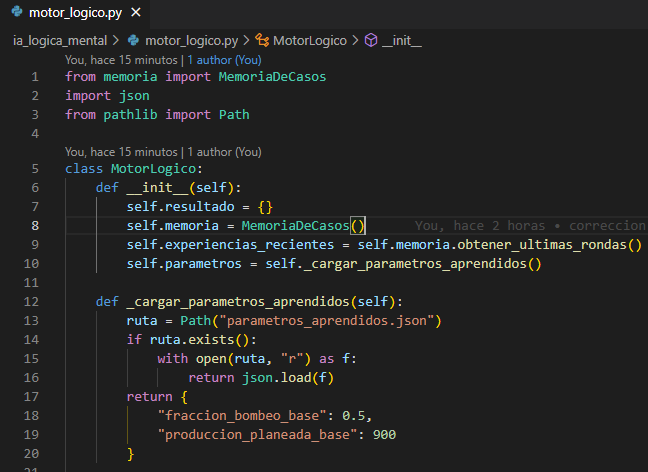


Una vez enviados, estos datos se agrupan como un diccionario estado y se pasan al componente central del sistema: el **motor lógico**.

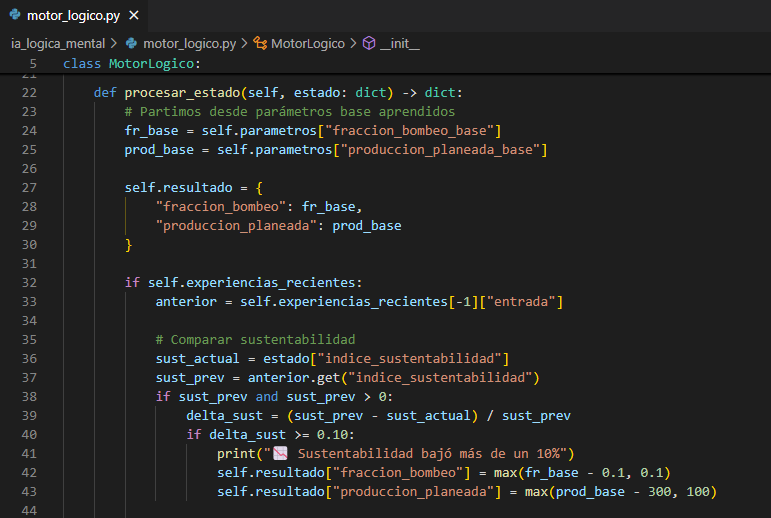
**2. Motor lógico (motor\_logico.py)**

Este módulo realiza tres funciones principales:

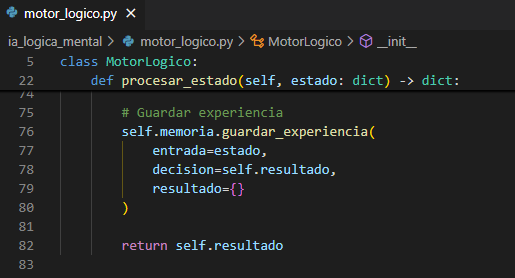
* **Carga los parámetros aprendidos** desde un archivo (parametros\_aprendidos.json), que contiene valores base para fracción de bombeo y producción planeada.



* **Compara la ronda actual con la ronda anterior** (obtenida desde la memoria de corto plazo), calculando variaciones porcentuales de cuatro variables clave:
  + Sustentabilidad
  + Ganancias
  + Agua superficial
  + Consumo real
* Si se detectan cambios significativos (por ejemplo, una caída >10% en ganancias), el sistema ajusta los valores recomendados con base en reglas predefinidas.
* En todos los casos, el sistema parte desde los **valores aprendidos** como base de la decisión y modifica desde allí si las condiciones lo justifican.



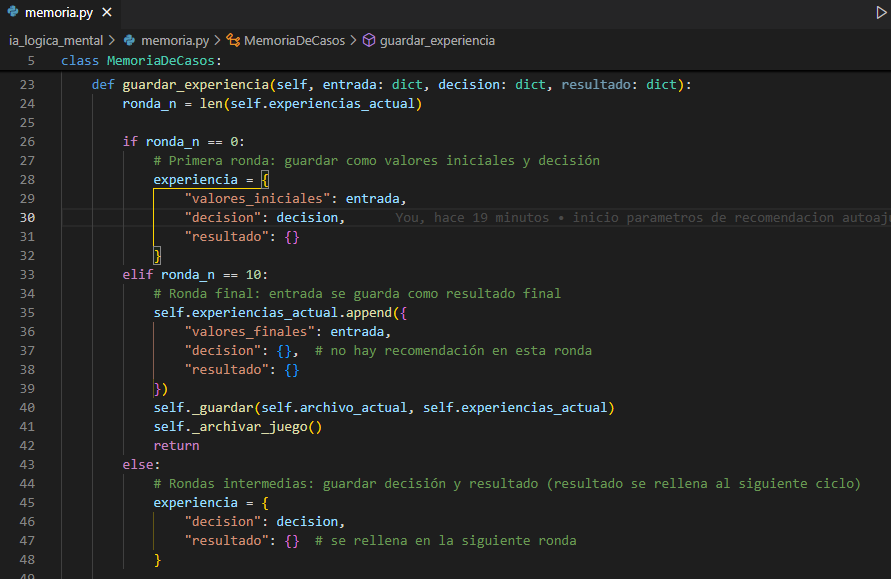
Finalmente, la decisión generada y los datos actuales se envían a memoria.py para su registro.



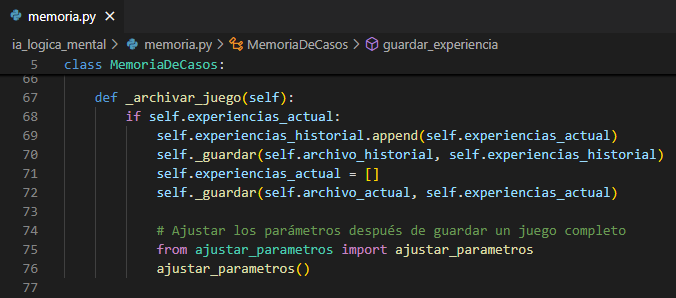
**3. Memoria (memoria.py)**

Este módulo gestiona la **memoria de corto y largo plazo** del sistema. Su lógica interna sigue estas reglas:

* **Primera ronda**: guarda el estado como valores\_iniciales, junto con la decisión tomada.
* **Rondas 2 a 10**: guarda la decisión y en la siguiente ronda actualiza la experiencia anterior con el resultado real de esa decisión.
* **Ronda 11**: guarda los datos como valores\_finales y archiva todo el bloque de 11 rondas (un “juego” completo) en experiencias\_historial.json.



Después de archivar, llama automáticamente al módulo de ajuste de parámetros.

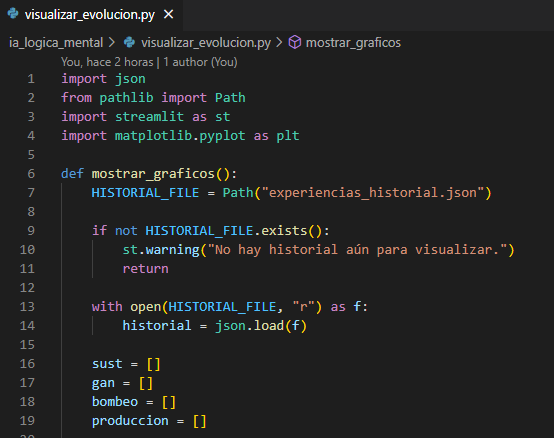


**4. Ajuste de parámetros (ajustar\_parametros.py)**

Este módulo analiza todos los juegos archivados y calcula un **promedio ponderado** de las decisiones pasadas, considerando su eficacia:

* Evalúa si el índice de ganancias y/o sustentabilidad **mejoraron entre el inicio y el final** del juego.
* Asigna un peso a cada juego en función de esa mejora.
* Promedia todas las decisiones intermedias (rondas 2 a 10) usando ese peso y actualiza los valores base en parametros\_aprendidos.json.

Esto permite que el sistema **aprenda de su historial** y afine los valores de sus recomendaciones en rondas futuras.

**5. Visualización (visualizar\_evolucion.py)**

La interfaz también permite graficar la evolución del sistema:

* Lee el archivo experiencias\_historial.json.
* Extrae de cada ronda los valores de:
  + Índice de sustentabilidad
  + Índice de ganancias
  + Fracción de bombeo
  + Producción planeada
* Genera un gráfico combinado que muestra cómo evolucionan estas variables a lo largo del tiempo.

Este módulo permite **observar el aprendizaje y desempeño del sistema**, tanto en decisiones como en resultados.

El archivo ajustar\_parametros.py cumple una función clave dentro del sistema de lógica mental: **ajustar automáticamente los parámetros base de recomendación** (fracción de bombeo y producción planeada) en función del desempeño obtenido en juegos anteriores registrados en la memoria de largo plazo. Es decir, este módulo representa el mecanismo de aprendizaje y retroalimentación del sistema.

El proceso se activa automáticamente cuando se completa un juego (es decir, cuando se llega a la ronda 10). En ese momento, la memoria transfiere los datos al historial de experiencias (experiencias\_historial.json) y llama a ajustar\_parametros().

Este ajuste se realiza mediante un análisis ponderado de las decisiones tomadas y sus resultados en términos de sustentabilidad y ganancias.

**⚙️ Funcionamiento general**

1. **Carga de historial y parámetros**:
   * El sistema comienza revisando si existe el archivo experiencias\_historial.json, el cual contiene los registros completos de todos los juegos anteriores.
   * También se define la ruta a parametros\_aprendidos.json, donde se guardarán los valores base ajustados.
2. **Evaluación de eficacia**:
   * Cada juego se evalúa en función del cambio entre sus valores iniciales y finales en dos variables clave: **índice de sustentabilidad** e **índice de ganancias**.
   * Esta comparación entrega un “score de eficacia”, que se usa como peso para valorar las decisiones tomadas durante ese juego. Mejores resultados generan mayor influencia en el ajuste de parámetros.
3. **Promediado ponderado**:
   * Para cada ronda intermedia (de la 1 a la 9) de cada juego, se extraen las decisiones tomadas (fracción de bombeo y producción planeada) y se acumulan multiplicadas por el peso calculado.
   * Se lleva un registro del peso total acumulado para luego calcular un promedio ponderado.
4. **Cálculo y actualización**:
   * Si el peso total es mayor que cero (es decir, si hay información útil), se calculan los nuevos parámetros base como el promedio ponderado de las decisiones.
   * Finalmente, se escriben estos valores actualizados en parametros\_aprendidos.json.

**📈 Importancia en el sistema**

Este módulo permite que el sistema mejore de forma adaptativa en función de su experiencia. Si un conjunto de decisiones ha producido resultados positivos en términos de sustentabilidad y ganancias, el sistema tenderá a replicar esas decisiones en juegos futuros. En cambio, si las decisiones llevaron a un deterioro, su influencia se verá reducida.

En la práctica, este mecanismo emula una forma simplificada de aprendizaje automático basada en lógica difusa, donde no se entrena un modelo matemático complejo, pero sí se ajustan gradualmente los parámetros de comportamiento con base en evidencia histórica.