Ejercicio 6

Crea un programa en Python que simule un pequeño sistema de gestión de un gimnasio, utilizando un diccionario principal llamado gimnasio.
En este diccionario, cada clave representará el identificador de un socio (por ejemplo, su número o nombre), y cada valor será otro diccionario con los datos del socio, incluyendo al menos:
"nombre" → nombre del socio
"cuota" → importe mensual de su cuota (valor numérico)
El programa debe permitir realizar las siguientes operaciones (de manera sencilla, sin usar funciones):
Insertar un nuevo socio con sus datos.
Modificar los datos de un socio existente (por ejemplo, actualizar su cuota).
Eliminar un socio por su identificador.
Calcular una métrica básica sobre las cuotas, como la media, la suma total o la cuota máxima.
Recorrer el diccionario con un bucle for para mostrar todos los socios y sus datos de forma legible.
El código debe escribirse de la forma más sencilla posible, como si lo hiciera una persona que está comenzando a programar.
No se deben usar funciones ni estructuras avanzadas, solo operaciones básicas con diccionarios, bucles y condicionales.
Incluye algunos comentarios y explicaciones para indicar lo que hace cada parte del programa.

Ejercicio 7

Crea un programa en Python que gestione la información de un gimnasio utilizando un único diccionario principal llamado gimnasio.
Cada elemento del diccionario representará un socio, donde:
La clave será el identificador del socio (por ejemplo, un número o string).
El valor será otro diccionario con al menos dos campos: "nombre" y "cuota".
El programa debe permitir, sin utilizar funciones, las siguientes operaciones:
Insertar un nuevo socio en el diccionario.
Modificar los datos (nombre o cuota) de un socio existente.
Eliminar un socio por su identificador.
Calcular una métrica sobre las cuotas (por ejemplo, suma, media o valor máximo).
recorrer el diccionario con un bucle for, mostrando las claves y los valores de forma legible.
Implementa un control de errores básico (por ejemplo, intentar modificar o eliminar un socio inexistente).
El código debe ser lo más simple y legible posible, como si fuera escrito por una persona que está comenzando a programar, pero con una estructura ordenada y explicaciones paso a paso.

Ejercicio 8

Diseña un programa en Python que gestione una colección de registros prácticos de la vida cotidiana (por ejemplo, productos de una tienda, alumnos de una escuela, o pacientes de una clínica).
Cada registro será un diccionario con al menos tres campos.
Todos los registros se almacenarán dentro de una lista principal.
Implementa funciones que permitan:
Añadir un nuevo registro, validando que no existan duplicados y que los campos no estén vacíos.
Buscar un registro por uno de sus campos (por ejemplo, nombre o ID).
Calcular una media sobre un campo numérico (por ejemplo, precios, notas o edades).
Controla los errores más comunes con estructuras try/except y validaciones simples (duplicados, tipos incorrectos o campos vacíos).
Usa un lenguaje claro y una estructura muy básica, como si lo hubiera hecho un programador principiante, pero bien explicado.
Además, crea un ejemplo práctico de una estructura inversa (diccionario de listas) que represente la misma información de otro modo, y explica:
Cómo funciona este tipo de estructura.
Qué ventajas y desventajas tiene frente a la lista de diccionarios.

Ejercicio 9

Crea un programa sencillo en Python que simule un sistema de registro de fichajes (entrada y salida de empleados).
El programa debe usar el módulo datetime para registrar la fecha y hora de cada acción.
Implementa tres funciones principales:
registrar_entrada(): almacena la hora actual como hora de entrada.
registrar_salida(): almacena la hora actual como hora de salida.
mostrar_registros(): muestra todos los fichajes guardados en pantalla de forma legible.

Usa un diccionario o lista de diccionarios para guardar los registros y controla errores comunes con try/except, tales como:
Entradas incorrectas por teclado.
Índices o claves inexistentes.
Operaciones inválidas.
Explica brevemente qué hace cada parte del código y por qué se utiliza datetime (por ejemplo, para capturar la hora del sistema en tiempo real).
El código debe ser funcional, sencillo y fácilmente entendible para principiantes.