Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Tema: Sistema de administración de Gimnasio.

Descripción:

Se requiere el desarrollo de un sistema de gestión de reservas para un gimnasio. Este sistema debe permitir la administración de las clases, entrenadores, miembros y reservas de los miembros para las diferentes clases. Además, el sistema debe ofrecer herramientas para la gestión eficiente del gimnasio, como la adición de nuevas clases, la asignación de entrenadores a las clases y la administración de las membresías de los clientes. Finalmente, el sistema debe contar con funciones de informes y análisis que permitan visualizar la ocupación de las clases, identificar las clases más populares, y analizar la frecuencia de asistencia de los miembros. Este sistema implementará un CRUD (Create, Read, Update, Delete), lo que significa que permitirá crear, leer, actualizar y eliminar datos de clases, entrenadores, miembros y reservas. Estas operaciones son fundamentales para gestionar eficientemente los datos en la aplicación.

Consignas:

- Definir una función para la creación y gestión del archivo JSON.

```python
import json


# Funcion para leer el archivo JSON, si no existe retorna un mensaje, también establece
# si hay errores de lectura en el archivo.
1 usage
def leer_archivo_json(nombre_archivo):
    try:
        with open(nombre_archivo, 'r') as archivo:
            datos = json.load(archivo)
        return datos
    except FileNotFoundError:
        print(f"El archivo {nombre_archivo} no existe. Se creará uno nuevo.")
        return {"entrenadores": [], "clases": [], "miembros": [], "reservas": []}
    except json.JSONDecodeError:
        print(f"El archivo {nombre_archivo} no contiene un JSON válido.")
        return {"entrenadores": [], "clases": [], "miembros": [], "reservas": []}


# Funcion para escribir el JSON
1 usage
def escribir_archivo_json(nombre_archivo, datos):
    with open(nombre_archivo, 'w') as archivo:
        json.dump(datos, archivo, indent=4)
```

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Universidad Nacional ARTURO JAURETCHE

Definir una función que liste todos los entrenadores registrados. La función deberá imprimir cada entrenador en la lista.

```
def listar_entrenadores():
    for entrenador in entrenadores:
        print(entrenador)
```

Definir una función que agregue un entrenador con nombre, especialidad y horario disponible. La función deberá agregar estos datos a una lista.

```
def agregar_entrenador(nombre, especialidad, horario_disponible):
    entrenadores.append({"nombre": nombre,
                         "especialidad": especialidad,
                         "horario_disponible": horario_disponible})
```

Definir una función que permita modificar cualquier dato de un entrenador existente. La función deberá permitir al usuario modificar el nombre, especialidad o el horario disponible de un entrenador. Si el entrenador no existe, la función deberá imprimir un mensaje indicando que el entrenador no fue encontrado.

```
def actualizar_entrenador(nombre):
    global entrenadores  # Necesario para modificar la lista global
    # Buscar el entrenador por nombre
    for entrenador in entrenadores:
        if entrenador["nombre"] == nombre:
            print(f"Datos actuales del entrenador '{nombre}':")
            print(entrenador)
            print("¿Qué datos desea actualizar?")
            opcion = input("1. Especialidad\n2. Horario disponible\nSeleccione una opción: ")
            if opcion == "1":
                nueva_especialidad = input("Ingrese la nueva especialidad: ")
                entrenador["especialidad"] = nueva_especialidad
                print("Especialidad actualizada correctamente.")
            elif opcion == "2":
                nuevo_horario = input("Ingrese el nuevo horario disponible: ")
                entrenador["horario_disponible"] = nuevo_horario
                print("Horario disponible actualizado correctamente.")
            else:
                print("Opción no válida.")
            return
    # Si no se encuentra al entrenador
    print(f"No se encontró ningún entrenador con el nombre '{nombre}'.")
```

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Universidad Nacional ARTURO JAURETCHE

- Definir una función que elimine un entrenador existente. La función deberá eliminar un entrenador de la lista según el nombre proporcionado. Si el entrenador no existe, la función deberá imprimir un mensaje indicando que el entrenador no fue encontrado.

```
def eliminar_entrenador(nombre):
    global entrenadores  # Necesario para modificar la lista global
    # Buscar el entrenador por nombre
    for entrenador in entrenadores:
        if entrenador["nombre"] == nombre:
            entrenadores.remove(entrenador)
            print(f"Entrenador '{nombre}' eliminado correctamente.")
            return
    # Si no se encuentra al entrenador
    print(f"No se encontró ningún entrenador con el nombre '{nombre}'.")
```

- Definir una función que liste todas las clases disponibles. La función deberá imprimir cada clase en la lista.

```
def listar_clases():
    for clase in clases:
        print(clase)
```

- Definir una función que agregue una clase con nombre, horario, duración, capacidad máxima, nivel de dificultad y nombre del entrenador. La función deberá agregar estos datos a una lista.

```
def agregar_clase(nombre, horario, duracion, capacidad_maxima, nivel_dificultad, nombre_entrenador):
    clases.append({"nombre": nombre,
                   "horario": horario,
                   "duracion": duracion,
                   "capacidad_maxima": capacidad_maxima,
                   "nivel_dificultad": nivel_dificultad,
                   "nombre_entrenador": nombre_entrenador})
```

- Definir una función que permita modificar cualquier dato de una clase existente. La función deberá permitir al usuario modificar el nombre, horario, duración, capacidad máxima, nivel de dificultad o el nombre del entrenador de una clase. Si la clase no existe, la función deberá imprimir un mensaje indicando que la clase no fue encontrada.

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

```python
def actualizar_clase(nombre):
    global clases  # Necesario para modificar la lista global
    # Buscar la clase por nombre
    for clase in clases:
        if clase["nombre"] == nombre:
            print(f"Datos actuales de la clase '{nombre}':")
            print(clase)
            print("¿Qué datos desea actualizar?")
            opcion = input("1. Nombre\n2. Horario\n3. Duracion\n4. Capacidad\n5. Dificultad\n6. Nombre "
                           "entrenador\nSeleccione una opción: ")
            if opcion == "1":
                nuevo_nombre = input("Ingrese el nuevo nombre de la clase: ")
                clase["nombre"] = nuevo_nombre
                print("Nombre actualizado correctamente.")
            elif opcion == "2":
                nuevo_horario = input("Ingrese el nuevo horario disponible: ")
                clase["horario"] = nuevo_horario
                print("Horario disponible actualizado correctamente.")
            elif opcion == "3":
                nueva_duracion = input("Ingrese la nueva duracion: ")
                clase["duracion"] = nueva_duracion
                print("Duracion actualizada correctamente")
            elif opcion == "4":
                nueva_capacidad = input("Ingrese la nueva capacidad: ")
                clase["capacidad_maxima"] = nueva_capacidad
                print("Capacidad maxima actualizada")
            elif opcion == "5":
                nueva_dificultad = input("Ingrese la nueva dificultad: ")
                clase["nivel_dificultad"] = nueva_dificultad
            elif opcion == "6":
                nuevo_entrenador = input("Ingrese el nuevo entrenador: ")
                clase["nombre_entrenador"] = nuevo_entrenador
            else:
                print("Opción no válida.")
            return
    # Si no se encuentra al entrenador
    print(f"No se encontró ningún entrenador con el nombre '{nombre}'.")
```

Definir una función que elimine una clase existente. La función deberá eliminar una clase de la lista según el nombre proporcionado. Si la clase no existe, la función deberá imprimir un mensaje indicando que la clase no fue encontrada.

```python
def eliminar_clase(nombre):
    global clases
    for clase in clases:
        if clase["nombre"] == nombre:
            clases.remove(clase)
            print(f"Clase '{nombre}' eliminada correctamente.")
            return
            # Si no se encuentra la clase
        print(f"No se encontró ninguna clase con el nombre '{nombre}'.")
```

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Universidad Nacional ARTURO JAURETCHE

🔱 Definir una función que liste todos los miembros registrados. La función deberá imprimir cada miembro en la lista.

```
def listar_miembros():
    for miembro in miembros:
        print(miembro)
```

🔱 Definir una función que agregue un miembro con nombre, número de membresía y tipo de membresía. La función deberá agregar estos datos a una lista.

```
def agregar_miembro(nombre, numero_membresia, tipo_membresia):
    miembros.append({"nombre": nombre,
                     "numero_membresia": numero_membresia,
                     "tipo_membresia": tipo_membresia,
                     "reservas": []})
```

🔱 Definir una función que permita modificar cualquier dato de un miembro existente. La función deberá permitir al usuario modificar el nombre, número de membresía o el tipo de membresía de un miembro. Si el miembro no existe, la función deberá imprimir un mensaje indicando que el miembro no fue encontrado.

```
def actualizar_miembro(nombre):
    global miembros  # Necesario para modificar la lista global
    # Buscar la clase por nombre
    for miembro in miembros:
        if miembro["nombre"] == nombre:
            print(f"Datos actuales del miembro '{nombre}':")
            print(miembro)
            print("¿Qué datos desea actualizar?")
            opcion = input("1. Nombre\n2. Numero de membresia\n3. Tipo de membresia\nSeleccione una opción: ")
            if opcion == "1":
                nuevo_nombre = input("Ingrese el nuevo nombre del miembro: ")
                miembro["nombre"] = nuevo_nombre
                print("Nombre actualizado correctamente.")
            elif opcion == "2":
                nuevo_num_membresia = input("Ingrese el nuevo numero de membresia: ")
                miembro["numero_membresia"] = nuevo_num_membresia
                print("Numero de membresia actualizado correctamente.")
            elif opcion == "3":
                nuevo_tipo_membresia = input("Ingrese el nuevo tipo de membresia: ")
                miembro["tipo_membresia"] = nuevo_tipo_membresia
                print("Tipo de membresia actualizada correctamente")
            else:
                print("Opción no válida.")
            return
    # Si no se encuentra al entrenador
    print(f"No se encontró ningún entrenador con el nombre '{nombre}'.")
```

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Universidad Nacional ARTURO JAURETCHE

- Definir una función que elimine un miembro existente. La función deberá eliminar un miembro de la lista según el nombre proporcionado. Si el miembro no existe, la función deberá imprimir un mensaje indicando que el miembro no fue encontrado.

```
def eliminar_miembro(nombre):
    global miembros
    for miembro in miembros:
        if miembro["nombre"] == nombre:
            miembros.remove(miembro)
            print(f"El miembro '{nombre}' fue eliminado correctamente.")
            return
            # Si no se encuentra la clase
        print(f"No se encontró ningun miembro con el nombre '{nombre}'.")
```

- Definir una función que liste todas las reservas realizadas. La función deberá imprimir cada reserva en la lista.

```
def listar_reservas():
    for reserva in reservas:
        print(reserva)
```

- Definir una función que agregue una reserva con el nombre de la clase, nombre del miembro, fecha de reserva y método de pago. La función deberá agregar estos datos a una lista.

```
def agregar_reserva(nombre_clase, nombre_miembro, fecha_reserva, metodo_pago):
    reservas.append({"nombre_clase": nombre_clase,
                     "nombre_miembro": nombre_miembro,
                     "fecha_reserva": fecha_reserva,
                     "metodo_pago": metodo_pago})
```

- Definir una función que permita modificar cualquier dato de una reserva existente. La función deberá permitir al usuario modificar el nombre de la clase, nombre del miembro, fecha de reserva o método de pago de una reserva. Si la reserva no existe, la función deberá imprimir un mensaje indicando que la reserva no fue encontrada.

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

```python
def actualizar_reserva(nombre):
    global reservas  # Necesario para modificar la lista global
    # Buscar la reserva por nombre
    for reserva in reservas:
        if reserva["nombre_clase"] == nombre:
            print(f"Datos actuales de la reserva:")
            print(reserva)
            print("¿Qué datos desea actualizar?")
            opcion = input("1. Nombre de la clase\n"
                           "2. Nombre del miembro\n3. Fecha de la reserva\n"
                           "4. Metodo de pago\nSeleccione una opción: ")
            if opcion == "1":
                nuevo_nombre = input("Ingrese el nuevo nombre de la clase: ")
                reserva["nombre_clase"] = nuevo_nombre
                print("Nombre actualizado correctamente.")
            elif opcion == "2":
                nuevo_nombre_miembro = input("Ingrese el nuevo nombre del miembro: ")
                reserva["nombre_miembro"] = nuevo_nombre_miembro
                print("Nombre del miembro actualizado correctamente.")
            elif opcion == "3":
                nueva_fecha_reserva = input("Ingrese la nueva fecha de reserva: ")
                reserva["tipo_membresia"] = nueva_fecha_reserva
                print("Fecha de reserva actualizada correctamente")
            elif opcion == "4":
                nuevo_metodo_pago = input("Ingrese el nuevo metodo de pago: ")
                reserva["metodo_pago"] = nuevo_metodo_pago
                print("Metodo de pago actualizado correctamente")
            else:
                print("Opción no válida.")
            return
        # Si no se encuentra al entrenador
        print(f"No se encontró ningún entrenador con el nombre '{nombre}'.")
```

Definir una función que elimine una reserva existente. La función deberá eliminar una reserva de la lista según los datos proporcionados. Si la reserva no existe, la función deberá imprimir un mensaje indicando que la reserva no fue encontrada.

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

```python
def eliminar_reserva(nombre_clase, nombre_miembro, fecha_reserva):
    global reservas  # Necesario para modificar la lista global

    # Buscar y eliminar la reserva en la lista general de reservas
    reserva_encontrada = None
    for reserva in reservas:
        if (reserva["nombre_clase"] == nombre_clase and
                reserva["nombre_miembro"] == nombre_miembro and
                reserva["fecha_reserva"] == fecha_reserva):
            reserva_encontrada = reserva
            break
    if reserva_encontrada:
        reservas.remove(reserva_encontrada)
        print(f"Reserva para la clase '{nombre_clase}' en la fecha '{fecha_reserva}' eliminada correctamente.")
    else:
        print(f"No se encontró ninguna reserva para la clase '{nombre_clase}' en la fecha '{fecha_reserva}'.")
    # Buscar el miembro correspondiente
    miembro_encontrado = None
    for miembro in miembros:
        if miembro["nombre"] == nombre_miembro:
            miembro_encontrado = miembro
            break
    # Si se encuentra el miembro, eliminar la reserva de su lista de reservas
    if miembro_encontrado:
        reserva_encontrada_en_miembro = None
        for reserva in miembro_encontrado["reservas"]:
            if (reserva["nombre_clase"] == nombre_clase and
                    reserva["fecha_reserva"] == fecha_reserva):
                reserva_encontrada_en_miembro = reserva
                break
        if reserva_encontrada_en_miembro:
            miembro_encontrado["reservas"].remove(reserva_encontrada_en_miembro)
            print(f"Reserva eliminada de la lista del miembro '{nombre_miembro}'.")
        else:
            print(
                f"No se encontró ninguna reserva en la lista del miembro '{nombre_miembro}"
                f"' para la clase '{nombre_clase}' en la fecha '{fecha_reserva}'.")
    else:
        print(f"No se encontró ningún miembro con el nombre '{nombre_miembro}'.")
```

Definir una función que identifique las clases más populares. La función deberá analizar la lista de reservas y devolver las clases ordenadas por cantidad de reservas.

```python
def clases_mas_populares():
    popularidad = []
    for reserva in reservas:
        clase = reserva["nombre_clase"]
        encontrada = False
        for i in range(len(popularidad)):
            if popularidad[i]["nombre_clase"] == clase:
                popularidad[i]["cantidad"] += 1
                encontrada = True
                break
        if not encontrada:
            popularidad.append({"nombre_clase": clase, "cantidad": 1})
    popularidad_ordenada = sorted(popularidad, key=lambda x: x["cantidad"], reverse=True)
    return popularidad_ordenada
```

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Universidad Nacional ARTURO JAURETCHE

🔱 Definir una función que analice la frecuencia de asistencia de los miembros. La función deberá contar cuántas reservas tiene cada miembro y devolver esta información.

```
def frecuencia_asistencia_miembros():
    asistencia = []
    for reserva in reservas:
        miembro = reserva["nombre_miembro"]
        encontrado = False
        for i in range(len(asistencia)):
            if asistencia[i]["nombre_miembro"] == miembro:
                asistencia[i]["cantidad"] += 1
                encontrado = True
                break
        if not encontrado:
            asistencia.append({"nombre_miembro": miembro, "cantidad": 1})
    return asistencia
```

🔱 Definir una función para el menú principal que permita al usuario seleccionar opciones para agregar, listar y analizar datos de entrenadores, clases, miembros y reservas. La función deberá proporcionar una interfaz interactiva y llamar a las funciones correspondientes según la elección del usuario.

```
def menu_principal():
    condicion = 1
    while condicion == 1:
        print("")
        print("********* Bienvenido a Iron Gim *********".center(150))
        print("")
        print("****** MENU PRINCIPAL ******".center(150))
        print("")
        print("*** ENTRENADORES ***".center(150))
        print("1. Listar entrenadores".center(150))
        print("2. Agregar entrenador".center(150))
        print("3. Actualizar entrenador".center(150))
        print("4. Eliminar entrenador".center(150))
        print("")
        print("*** CLASES ***".center(150))
        print("5. Listar clases".center(150))
        print("6. Agregar clase".center(150))
        print("7. Actualizar clase".center(150))
        print("8. Eliminar clase".center(150))
        print("")
        print("*** MIEMBROS ***".center(150))
        print("9. Listar miembros".center(150))
        print("10. Agregar miembro".center(150))
        print("11. Actualizar miembro".center(150))
        print("12. Eliminar miembro".center(150))
        print("")
```

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Universidad Nacional ARTURO JAURETCHE

```
    print("*** ESTADISTICA ***".center(150))
    print("17. Clases más populares".center(150))
    print("18. Frecuencia de asistencia de miembros".center(150))
    print("")
    print("0. Salir".center(150))
    print("")
    opcion = input("Seleccione una opción: ")
    print("")
    # ENTRENADORES
    if opcion == "1":
        listar_entrenadores()
    elif opcion == "2":
        nombre = input("Ingrese el nombre del entrenador: ")
        especialidad = input("Especialidad: ")
        horario_disponible = input("Horario disponible: ")
        agregar_entrenador(nombre, especialidad, horario_disponible)
    elif opcion == "3":
        nombre = input("Ingrese el nombre del entrenador cuyos datos quiere actualizar: ")
        actualizar_entrenador(nombre)
    elif opcion == "4":
        nombre = input("Ingrese el nombre del entrenador que desea eliminar: ")
        eliminar_entrenador(nombre)
    # CLASES
    elif opcion == "5":
        listar_clases()
    elif opcion == "6":
        nombre = input("Escriba el nombre de la clase: ")
        horario = input("Ingrese el horario: ")
        duracion = input("Duración de la clase en horas: ")
        capacidad_maxima = input("Ingrese la cantidad de vacantes: ")
        nivel_dificultad = input("Ingrese el nivel de dificultad: ")
        nombre_entrenador = input("Ingrese el nombre del entrenador: ")
        agregar_clase(nombre, horario, duracion, capacidad_maxima, nivel_dificultad, nombre_entrenador)
    elif opcion == "7":
        nombre = input("Ingrese el nombre de la clase cuyos datos quiere actualizar: ")
        actualizar_clase(nombre)
    elif opcion == "8":
        nombre = input("Ingrese el nombre de la clase que desea eliminar: ")
        eliminar_clase(nombre)
    # MIEMBROS
    elif opcion == "9":
        listar_miembros()
    elif opcion == "10":
        nombre = input("Ingrese el nombre del miembro: ")
        numero_membresia = input("Digite el número de membresía: ")
        tipo_membresia = input("Especifique el tipo de membresía que posee: ")
        agregar_miembro(nombre, numero_membresia, tipo_membresia)
    elif opcion == "11":
        nombre = input("Ingrese el nombre del miembro cuyos datos quiere actualizar: ")
        actualizar_miembro(nombre)
    elif opcion == "12":
        nombre = input("Ingrese el nombre del miembro que desea eliminar: ")
        eliminar_miembro(nombre)
```

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Universidad Nacional ARTURO JAURETCHE

```
    # RESERVAS
    elif opcion == "13":
        listar_reservas()
    elif opcion == "14":
        nombre_clase = input("Ingrese el nombre de la clase: ")
        nombre_miembro = input("Ingrese el nombre del miembro: ")
        fecha_reserva = input("Ingrese la fecha de reserva: ")
        metodo_pago = input("Ingrese el método de pago: ")
        # Agregar la reserva
        agregar_reserva(nombre_clase, nombre_miembro, fecha_reserva, metodo_pago)
        # Obtener el miembro correspondiente
        miembro_encontrado = None
        for miembro in miembros:
            if miembro["nombre"] == nombre_miembro:
                miembro_encontrado = miembro
                break
        # Si se encuentra el miembro, agregar la reserva a ese miembro
        if miembro_encontrado:
            reserva = {
                "nombre_clase": nombre_clase,
                "fecha_reserva": fecha_reserva,
                "metodo_pago": metodo_pago
            }
            agregar_reserva_a_miembro(miembro_encontrado, reserva)
        else:
            print(f"No se encontró ningún miembro con el nombre '{nombre_miembro}'.")
    elif opcion == "15":
        nombre = input("Ingrese el nombre de la reserva cuyos datos quiere actualizar: ")
        actualizar_reserva(nombre)
    elif opcion == "16":
        nombre_clase = input("Ingrese el nombre de la clase: ")
        nombre_miembro = input("Ingrese el nombre del miembro: ")
        fecha_reserva = input("Ingrese la fecha de reserva: ")
        eliminar_reserva(nombre_clase, nombre_miembro, fecha_reserva)
    # ESTADISTICAS
    elif opcion == "17":
        print(clases_mas_populares())
    elif opcion == "18":
        print(frecuencia_asistencia_miembros())
    # SALIR
    elif opcion == "0":
        print("********* ¡¡¡Gracias, vuelva pronto!!! *********".center(150))
    # Guardar los datos antes de salir
        datos_gimnasio["entrenadores"] = entrenadores
        datos_gimnasio["clases"] = clases
        datos_gimnasio["miembros"] = miembros
        datos_gimnasio["reservas"] = reservas
        escribir_archivo_json( nombre_archivo: 'gym_data.json', datos_gimnasio)
        condicion = 0
    else:
        print("Opción no válida. Intente nuevamente.")
```

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Universidad Nacional ARTURO JAURETCHE

Código:

```
import json


# Funcion para leer el archivo JSON, si no existe retorna un mensaje, también
establece
# si hay errores de lectura en el archivo.
def leer_archivo_json(nombre_archivo):
    try:
        with open(nombre_archivo, 'r') as archivo:
            datos = json.load(archivo)
        return datos
    except FileNotFoundError:
        print(f"El archivo {nombre_archivo} no existe. Se creará uno nuevo.")
        return {"entrenadores": [], "clases": [], "miembros": [], "reservas": []}
    except json.JSONDecodeError:
        print(f"El archivo {nombre_archivo} no contiene un JSON válido.")
        return {"entrenadores": [], "clases": [], "miembros": [], "reservas": []}


# Funcion para escribir el JSON
def escribir_archivo_json(nombre_archivo, datos):
    with open(nombre_archivo, 'w') as archivo:
        json.dump(datos, archivo, indent=4)


# Lectura de las entidades creadas.
datos_gimnasio = leer_archivo_json('gym_data.json')
entrenadores = datos_gimnasio["entrenadores"]
clases = datos_gimnasio["clases"]
miembros = datos_gimnasio["miembros"]
reservas = datos_gimnasio["reservas"]


# Funciones para gestionar los entrenadores
def listar_entrenadores():
    for entrenador in entrenadores:
        print(entrenador)


def agregar_entrenador(nombre, especialidad, horario_disponible):
    entrenadores.append({"nombre": nombre,
                         "especialidad": especialidad,
                         "horario_disponible": horario_disponible})


def actualizar_entrenador(nombre):
    global entrenadores  # Necesario para modificar la lista global
    # Buscar el entrenador por nombre
    for entrenador in entrenadores:
        if entrenador["nombre"] == nombre:
            print(f"Datos actuales del entrenador '{nombre}':")
            print(entrenador)
            print("¿Qué datos desea actualizar?")
            opcion = input("1. Especialidad\n2. Horario disponible\nSeleccione una
opción: ")
            if opcion == "1":
                nueva_especialidad = input("Ingrese la nueva especialidad: ")
                entrenador["especialidad"] = nueva_especialidad
                print("Especialidad actualizada correctamente.")
```

```
            elif opcion == "2":
                nuevo_horario = input("Ingrese el nuevo horario disponible: ")
                entrenador["horario_disponible"] = nuevo_horario
                print("Horario disponible actualizado correctamente.")
            else:
                print("Opción no válida.")
            return
    # Si no se encuentra al entrenador
    print(f"No se encontró ningún entrenador con el nombre '{nombre}'.")


def eliminar_entrenador(nombre):
    global entrenadores  # Necesario para modificar la lista global
    # Buscar el entrenador por nombre
    for entrenador in entrenadores:
        if entrenador["nombre"] == nombre:
            entrenadores.remove(entrenador)
            print(f"Entrenador '{nombre}' eliminado correctamente.")
            return
    # Si no se encuentra al entrenador
    print(f"No se encontró ningún entrenador con el nombre '{nombre}'.")


# Funciones para gestionar las clases que se dan en el gimnasio
def listar_clases():
    for clase in clases:
        print(clase)


def agregar_clase(nombre, horario, duracion, capacidad_maxima, nivel_dificultad,
nombre_entrenador):
    clases.append({"nombre": nombre,
                   "horario": horario,
                   "duracion": duracion,
                   "capacidad_maxima": capacidad_maxima,
                   "nivel_dificultad": nivel_dificultad,
                   "nombre_entrenador": nombre_entrenador})


def actualizar_clase(nombre):
    global clases  # Necesario para modificar la lista global
    # Buscar la clase por nombre
    for clase in clases:
        if clase["nombre"] == nombre:
            print(f"Datos actuales de la clase '{nombre}':")
            print(clase)
            print("¿Qué datos desea actualizar?")
            opcion = input("1. Nombre\n2. Horario\n3. Duracion\n4. Capacidad\n5.
Dificultad\n6. Nombre "
                           "entrenador\nSeleccione una opción: ")
            if opcion == "1":
                nuevo_nombre = input("Ingrese el nuevo nombre de la clase: ")
                clase["nombre"] = nuevo_nombre
                print("Nombre actualizado correctamente.")
            elif opcion == "2":
                nuevo_horario = input("Ingrese el nuevo horario disponible: ")
                clase["horario"] = nuevo_horario
                print("Horario disponible actualizado correctamente.")
            elif opcion == "3":
                nueva_duracion = input("Ingrese la nueva duracion: ")
                clase["duracion"] = nueva_duracion
                print("Duracion actualizada correctamente")
```

```
            elif opcion == "4":
                nueva_capacidad = input("Ingrese la nueva capacidad: ")
                clase["capacidad_maxima"] = nueva_capacidad
                print("Capacidad maxima actualizada")
            elif opcion == "5":
                nueva_dificultad = input("Ingrese la nueva dificultad: ")
                clase["nivel_dificultad"] = nueva_dificultad
            elif opcion == "6":
                nuevo_entrenador = input("Ingrese el nuevo entrenador: ")
                clase["nombre_entrenador"] = nuevo_entrenador
            else:
                print("Opción no válida.")
            return
    # Si no se encuentra al entrenador
    print(f"No se encontró ningún entrenador con el nombre '{nombre}'.")


def eliminar_clase(nombre):
    global clases
    for clase in clases:
        if clase["nombre"] == nombre:
            clases.remove(clase)
            print(f"Clase '{nombre}' eliminada correctamente.")
            return
            # Si no se encuentra la clase
        print(f"No se encontró ninguna clase con el nombre '{nombre}'.")


# Funciones para gestionar miembros del gimnasio
def listar_miembros():
    for miembro in miembros:
        print(miembro)


def agregar_miembro(nombre, numero_membresia, tipo_membresia):
    miembros.append({"nombre": nombre,
                     "numero_membresia": numero_membresia,
                     "tipo_membresia": tipo_membresia,
                     "reservas": []})


def agregar_reserva_a_miembro(miembro, reserva):
    miembro["reservas"].append(reserva)


def actualizar_miembro(nombre):
    global miembros  # Necesario para modificar la lista global
    # Buscar la clase por nombre
    for miembro in miembros:
        if miembro["nombre"] == nombre:
            print(f"Datos actuales del miembro '{nombre}':")
            print(miembro)
            print("¿Qué datos desea actualizar?")
            opcion = input("1. Nombre\n2. Numero de membresia\n3. Tipo de
membresia\nSeleccione una opción: ")
            if opcion == "1":
                nuevo_nombre = input("Ingrese el nuevo nombre del miembro: ")
                miembro["nombre"] = nuevo_nombre
                print("Nombre actualizado correctamente.")
            elif opcion == "2":
                nuevo_num_membresia = input("Ingrese el nuevo numero de membresia: ")
                miembro["numero_membresia"] = nuevo_num_membresia
```

```
                print("Numero de membresia actualizado correctamente.")
            elif opcion == "3":
                nuevo_tipo_membresia = input("Ingrese el nuevo tipo de membresia: ")
                miembro["tipo_membresia"] = nuevo_tipo_membresia
                print("Tipo de membresia actualizada correctamente")
            else:
                print("Opción no válida.")
            return
    # Si no se encuentra al entrenador
    print(f"No se encontró ningún entrenador con el nombre '{nombre}'.")


def eliminar_miembro(nombre):
    global miembros
    for miembro in miembros:
        if miembro["nombre"] == nombre:
            miembros.remove(miembro)
            print(f"El miembro '{nombre}' fue eliminado correctamente.")
            return
            # Si no se encuentra la clase
        print(f"No se encontró ningun miembro con el nombre '{nombre}'.")


# Funciones para gestionar reservas
def listar_reservas():
    for reserva in reservas:
        print(reserva)


def agregar_reserva(nombre_clase, nombre_miembro, fecha_reserva, metodo_pago):
    reservas.append({"nombre_clase": nombre_clase,
                     "nombre_miembro": nombre_miembro,
                     "fecha_reserva": fecha_reserva,
                     "metodo_pago": metodo_pago})


def actualizar_reserva(nombre):
    global reservas  # Necesario para modificar la lista global
    # Buscar la reserva por nombre
    for reserva in reservas:
        if reserva["nombre_clase"] == nombre:
            print(f"Datos actuales de la reserva:")
            print(reserva)
            print("¿Qué datos desea actualizar?")
            opcion = input("1. Nombre de la clase\n"
                           "2. Nombre del miembro\n3. Fecha de la reserva\n"
                           "4. Metodo de pago\nSeleccione una opción: ")
            if opcion == "1":
                nuevo_nombre = input("Ingrese el nuevo nombre de la clase: ")
                reserva["nombre_clase"] = nuevo_nombre
                print("Nombre actualizado correctamente.")
            elif opcion == "2":
                nuevo_nombre_miembro = input("Ingrese el nuevo nombre del miembro: ")
                reserva["nombre_miembro"] = nuevo_nombre_miembro
                print("Nombre del miembro actualizado correctamente.")
            elif opcion == "3":
                nueva_fecha_reserva = input("Ingrese la nueva fecha de reserva: ")
                reserva["tipo_membresia"] = nueva_fecha_reserva
                print("Fecha de reserva actualizada correctamente")
            elif opcion == "4":
                nuevo_metodo_pago = input("Ingrese el nuevo metodo de pago: ")
                reserva["metodo_pago"] = nuevo_metodo_pago
```

```
                print("Metodo de pago actualizado correctamente")
            else:
                print("Opción no válida.")
            return
        # Si no se encuentra al entrenador
        print(f"No se encontró ningún entrenador con el nombre '{nombre}'.")


def eliminar_reserva(nombre_clase, nombre_miembro, fecha_reserva):
    global reservas  # Necesario para modificar la lista global

    # Buscar y eliminar la reserva en la lista general de reservas
    reserva_encontrada = None
    for reserva in reservas:
        if (reserva["nombre_clase"] == nombre_clase and
                reserva["nombre_miembro"] == nombre_miembro and
                reserva["fecha_reserva"] == fecha_reserva):
            reserva_encontrada = reserva
            break
    if reserva_encontrada:
        reservas.remove(reserva_encontrada)
        print(f"Reserva para la clase '{nombre_clase}' en la fecha '{fecha_reserva}'
eliminada correctamente.")
    else:
        print(f"No se encontró ninguna reserva para la clase '{nombre_clase}' en la
fecha '{fecha_reserva}'.")
    # Buscar el miembro correspondiente
    miembro_encontrado = None
    for miembro in miembros:
        if miembro["nombre"] == nombre_miembro:
            miembro_encontrado = miembro
            break
    # Si se encuentra el miembro, eliminar la reserva de su lista de reservas
    if miembro_encontrado:
        reserva_encontrada_en_miembro = None
        for reserva in miembro_encontrado["reservas"]:
            if (reserva["nombre_clase"] == nombre_clase and
                    reserva["fecha_reserva"] == fecha_reserva):
                reserva_encontrada_en_miembro = reserva
                break
        if reserva_encontrada_en_miembro:
            miembro_encontrado["reservas"].remove(reserva_encontrada_en_miembro)
            print(f"Reserva eliminada de la lista del miembro '{nombre_miembro}'.")
        else:
            print(
                f"No se encontró ninguna reserva en la lista del miembro
'{nombre_miembro}"
                f"' para la clase '{nombre_clase}' en la fecha '{fecha_reserva}'.")
    else:
        print(f"No se encontró ningún miembro con el nombre '{nombre_miembro}'.")


# Funciones para informes y análisis
def clases_mas_populares():
    popularidad = []
    for reserva in reservas:
        clase = reserva["nombre_clase"]
        encontrada = False
        for i in range(len(popularidad)):
            if popularidad[i]["nombre_clase"] == clase:
                popularidad[i]["cantidad"] += 1
                encontrada = True
```

```
                break
        if not encontrada:
            popularidad.append({"nombre_clase": clase, "cantidad": 1})
    popularidad_ordenada = sorted(popularidad, key=lambda x: x["cantidad"],
reverse=True)
    return popularidad_ordenada


def frecuencia_asistencia_miembros():
    asistencia = []
    for reserva in reservas:
        miembro = reserva["nombre_miembro"]
        encontrado = False
        for i in range(len(asistencia)):
            if asistencia[i]["nombre_miembro"] == miembro:
                asistencia[i]["cantidad"] += 1
                encontrado = True
                break
        if not encontrado:
            asistencia.append({"nombre_miembro": miembro, "cantidad": 1})
    return asistencia


# Menú principal
def menu_principal():
    condicion = 1
    while condicion == 1:
        print("")
        print("********* Bienvenido a Iron Gim *********".center(150))
        print("")
        print("****** MENU PRINCIPAL ******".center(150))
        print("")
        print("*** ENTRENADORES ***".center(150))
        print("1. Listar entrenadores".center(150))
        print("2. Agregar entrenador".center(150))
        print("3. Actualizar entrenador".center(150))
        print("4. Eliminar entrenador".center(150))
        print("")
        print("*** CLASES ***".center(150))
        print("5. Listar clases".center(150))
        print("6. Agregar clase".center(150))
        print("7. Actualizar clase".center(150))
        print("8. Eliminar clase".center(150))
        print("")
        print("*** MIEMBROS ***".center(150))
        print("9. Listar miembros".center(150))
        print("10. Agregar miembro".center(150))
        print("11. Actualizar miembro".center(150))
        print("12. Eliminar miembro".center(150))
        print("")
        print("*** RESERVAS ***".center(150))
        print("13. Listar reservas".center(150))
        print("14. Agregar reserva".center(150))
        print("15. Actualizar reserva".center(150))
        print("16. Eliminar reserva".center(150))
        print("")
        print("*** ESTADISTICA ***".center(150))
        print("17. Clases más populares".center(150))
        print("18. Frecuencia de asistencia de miembros".center(150))
        print("")
        print("0. Salir".center(150))
        print("")
```

```
        opcion = input("Seleccione una opción: ")
        print("")
        # ENTRENADORES
        if opcion == "1":
            listar_entrenadores()
        elif opcion == "2":
            nombre = input("Ingrese el nombre del entrenador: ")
            especialidad = input("Especialidad: ")
            horario_disponible = input("Horario disponible: ")
            agregar_entrenador(nombre, especialidad, horario_disponible)
        elif opcion == "3":
            nombre = input("Ingrese el nombre del entrenador cuyos datos quiere
actualizar: ")
            actualizar_entrenador(nombre)
        elif opcion == "4":
            nombre = input("Ingrese el nombre del entrenador que desea eliminar: ")
            eliminar_entrenador(nombre)
        # CLASES
        elif opcion == "5":
            listar_clases()
        elif opcion == "6":
            nombre = input("Escriba el nombre de la clase: ")
            horario = input("Ingrese el horario: ")
            duracion = input("Duración de la clase en horas: ")
            capacidad_maxima = input("Ingrese la cantidad de vacantes: ")
            nivel_dificultad = input("Ingrese el nivel de dificultad: ")
            nombre_entrenador = input("Ingrese el nombre del entrenador: ")
            agregar_clase(nombre, horario, duracion, capacidad_maxima,
nivel_dificultad, nombre_entrenador)
        elif opcion == "7":
            nombre = input("Ingrese el nombre de la clase cuyos datos quiere
actualizar: ")
            actualizar_clase(nombre)
        elif opcion == "8":
            nombre = input("Ingrese el nombre de la clase que desea eliminar: ")
            eliminar_clase(nombre)
        # MIEMBROS
        elif opcion == "9":
            listar_miembros()
        elif opcion == "10":
            nombre = input("Ingrese el nombre del miembro: ")
            numero_membresia = input("Digite el número de membresía: ")
            tipo_membresia = input("Especifique el tipo de membresía que posee: ")
            agregar_miembro(nombre, numero_membresia, tipo_membresia)
        elif opcion == "11":
            nombre = input("Ingrese el nombre del miembro cuyos datos quiere
actualizar: ")
            actualizar_miembro(nombre)
        elif opcion == "12":
            nombre = input("Ingrese el nombre del miembro que desea eliminar: ")
            eliminar_miembro(nombre)
        # RESERVAS
        elif opcion == "13":
            listar_reservas()
        elif opcion == "14":
            nombre_clase = input("Ingrese el nombre de la clase: ")
            nombre_miembro = input("Ingrese el nombre del miembro: ")
            fecha_reserva = input("Ingrese la fecha de reserva: ")
            metodo_pago = input("Ingrese el método de pago: ")
            # Agregar la reserva
            agregar_reserva(nombre_clase, nombre_miembro, fecha_reserva, metodo_pago)
            # Obtener el miembro correspondiente
```

```
            miembro_encontrado = None
            for miembro in miembros:
                if miembro["nombre"] == nombre_miembro:
                    miembro_encontrado = miembro
                    break
            # Si se encuentra el miembro, agregar la reserva a ese miembro
            if miembro_encontrado:
                reserva = {
                    "nombre_clase": nombre_clase,
                    "fecha_reserva": fecha_reserva,
                    "metodo_pago": metodo_pago
                }
                agregar_reserva_a_miembro(miembro_encontrado, reserva)
            else:
                print(f"No se encontró ningún miembro con el nombre
'{nombre_miembro}'.")
        elif opcion == "15":
            nombre = input("Ingrese el nombre de la reserva cuyos datos quiere
actualizar: ")
            actualizar_reserva(nombre)
        elif opcion == "16":
            nombre_clase = input("Ingrese el nombre de la clase: ")
            nombre_miembro = input("Ingrese el nombre del miembro: ")
            fecha_reserva = input("Ingrese la fecha de reserva: ")
            eliminar_reserva(nombre_clase, nombre_miembro, fecha_reserva)
        # ESTADISTICAS
        elif opcion == "17":
            print(clases_mas_populares())
        elif opcion == "18":
            print(frecuencia_asistencia_miembros())
        # SALIR
        elif opcion == "0":
            print("********* ¡¡¡Gracias, vuelva pronto!!! *********".center(150))
        # Guardar los datos antes de salir
            datos_gimnasio["entrenadores"] = entrenadores
            datos_gimnasio["clases"] = clases
            datos_gimnasio["miembros"] = miembros
            datos_gimnasio["reservas"] = reservas
            escribir_archivo_json('gym_data.json', datos_gimnasio)
            condicion = 0
        else:
            print("Opción no válida. Intente nuevamente.")


# Iniciar el menú principal
menu_principal()
```

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Explicación del código:

Funciones Principales del Sistema

Lectura y Escritura de Datos en JSON

La función leer_archivo_json tiene como objetivo leer datos desde un archivo JSON y manejar diferentes escenarios de error para garantizar que el programa funcione correctamente incluso si el archivo no existe o contiene datos inválidos. La función toma un argumento nombre_archivo, que es una cadena que representa el nombre del archivo JSON a leer.

Se inicia un bloque try para intentar ejecutar el código que podría generar excepciones.

Se utiliza una declaración with para abrir el archivo en modo de lectura ('r'). La declaración with asegura que el archivo se cierre correctamente después de que se haya terminado de trabajar con él, incluso si ocurre una excepción. El archivo abierto se asigna al objeto archivo.

La función json.load se utiliza para leer el contenido del archivo y convertirlo de formato JSON a un diccionario de Python. Los datos leídos se almacenan en la variable datos. Si la lectura del archivo y la carga del JSON son exitosas, la función retorna el diccionario datos que contiene la información del archivo JSON.

FileNotFoundError: Este bloque except captura la excepción FileNotFoundError, que ocurre si el archivo especificado no existe y se imprime un mensaje indicando que el archivo no existe y que se creará uno nuevo. Se retorna un diccionario con listas vacías para las claves entrenadores, clases, miembros y reservas. Esto asegura que el programa tenga estructuras de datos válidas para trabajar, aunque el archivo original no exista.

JSONDecodeError: Este bloque except captura la excepción JSONDecodeError, que ocurre si el contenido del archivo no es un JSON válido. Se imprime un mensaje indicando que el archivo no contiene un JSON válido. Similar al manejo de FileNotFoundError, se retorna un diccionario con listas vacías para las claves entrenadores, clases, miembros y reservas. Esto permite que el programa continúe funcionando incluso si el archivo tiene datos no válidos.

Escritura en Archivo JSON: La función escribir_archivo_json tiene como objetivo escribir datos en un archivo en formato JSON. Se define una función que toma dos argumentos:

a) nombre_archivo: una cadena que representa el nombre del archivo en el que se escribirán los datos.
b) datos: los datos que se escribirán en el archivo, generalmente un diccionario o una lista.

Se utiliza una declaración with para abrir el archivo en modo de escritura ('w'). La declaración with asegura que el archivo se cierre correctamente después de que se haya terminado de trabajar con él, incluso si ocurre una excepción. El archivo abierto se asigna al objeto archivo.

La función json.dump se utiliza para convertir los datos de Python (almacenados en la variable datos) a formato JSON y escribirlos en el archivo.

a) datos: los datos que se van a escribir en el archivo.
b) archivo: el archivo en el que se escribirá el JSON.
c) indent=4: opcional, añade sangría de 4 espacios para mejorar la legibilidad del archivo JSON generado.

Gestión de Entrenadores

- Listar Entrenadores: Muestra una lista con los detalles de todos los entrenadores registrados.
- Agregar Entrenador: Permite añadir un nuevo entrenador especificando su nombre, especialidad y horario disponible.

- Actualizar Entrenador: Facilita la actualización de los datos de un entrenador, permitiendo modificar su especialidad o horario disponible.
- Eliminar Entrenador: Elimina un entrenador del sistema según su nombre.

Gestión de Clases

- Listar Clases: Muestra una lista con los detalles de todas las clases disponibles.
- Agregar Clase: Permite añadir una nueva clase especificando su nombre, horario, duración, capacidad máxima, nivel de dificultad y nombre del entrenador.
- Actualizar Clase: Facilita la actualización de los datos de una clase, permitiendo modificar su nombre, horario, duración, capacidad, nivel de dificultad o nombre del entrenador.
- Eliminar Clase: Elimina una clase del sistema según su nombre.

Gestión de Miembros

- Listar Miembros: Muestra una lista con los detalles de todos los miembros del gimnasio.
- Agregar Miembro: Permite añadir un nuevo miembro especificando su nombre, número de membresía y tipo de membresía.
- Actualizar Miembro: Facilita la actualización de los datos de un miembro, permitiendo modificar su nombre, número de membresía o tipo de membresía.
- Eliminar Miembro: Elimina un miembro del sistema según su nombre.

Gestión de Reservas

- Listar Reservas: Muestra una lista con los detalles de todas las reservas realizadas.
- Agregar Reserva: Permite añadir una nueva reserva especificando el nombre de la clase, nombre del miembro, fecha de la reserva y método de pago. Además, se agrega la reserva a la lista de reservas del miembro correspondiente.
- Actualizar Reserva: Facilita la actualización de los datos de una reserva, permitiendo modificar el nombre de la clase, nombre del miembro, fecha de la reserva o método de pago.
- Eliminar Reserva: Elimina una reserva del sistema según el nombre de la clase, nombre del miembro y fecha de la reserva.

Informes y Análisis

- Clases Más Populares: Genera un informe de las clases más populares basado en la cantidad de reservas.
- Frecuencia de Asistencia de Miembros: Genera un informe de la frecuencia de asistencia de los miembros basado en las reservas realizadas.

Menú Principal del Sistema

El sistema presenta un menú principal que permite al usuario interactuar con todas las funciones mencionadas. El menú está organizado en diferentes secciones:

- Entrenadores: Incluye opciones para listar, agregar, actualizar y eliminar entrenadores.
- Clases: Incluye opciones para listar, agregar, actualizar y eliminar clases.

- Miembros: Incluye opciones para listar, agregar, actualizar y eliminar miembros.
- Reservas: Incluye opciones para listar, agregar, actualizar y eliminar reservas.
- Estadísticas: Incluye opciones para generar informes de las clases más populares y la frecuencia de asistencia de los miembros.
- Salir: Permite salir del sistema guardando los cambios realizados.

Flujo de Operación del Sistema

- Inicio del Menú Principal: El usuario es recibido con un mensaje de bienvenida y las opciones disponibles en el menú principal.
- Selección de Opción: El usuario selecciona una opción ingresando el número correspondiente.
- Ejecución de la Opción Seleccionada: El sistema ejecuta la función correspondiente según la opción seleccionada.
- Actualización de Datos: Cualquier cambio realizado se actualiza en las estructuras de datos internas y se guarda en el archivo JSON.
- Cierre del Sistema: Al seleccionar la opción de salir, el sistema guarda todos los datos en el archivo JSON y finaliza la ejecución.

Fundamentos de informática
Alumno: Mendoza Santiago Ricardo José
Profesora: Segovia Wenddy
Fecha: 23-06-2024
Comisión: 10

Bibliografía:

- Sweigart, Al. "Automate the Boring Stuff with Python: Practical Programming for Total Beginners." No Starch Press, 2015.

- Matthes, Eric. "Python Crash Course: A Hands-On, Project-Based Introduction to Programming." No Starch Press, 2019.

- Salvatierra, José. "Aprende Python en un fin de semana." Independently published, 2018.