

Reto Práctico: "Gestor de Torneos Cortos de Futbol "

Descripción

Desarrolla un programa en Python que gestione torneos de futbol. El programa permitirá a los organizadores de torneos:

- crear un torneo con 4 equipos,
- sortear los cruces para definir los partidos de cada semana
- registrar los resultados de los partidos
- mostrar las clasificaciones y estadísticas.

Los torneos serán siempre cuadrangulares todos contra todos. El ganador es el que más puntos obtenga así (3 puntos por victoria, 1 punto por empate) si hay equipos empatados en puntos se desempata con diferencia de gol y si persiste el empate se desempata por goles a favor.

Requisitos Técnicos

Estructura del Proyecto

El proyecto debe estar organizado en la siguiente estructura de carpetas:

```
tournament/  
├── main.py  
├── db_manager.py  
├── classes/  
│   ├── __init__.py  
│   ├── team.py  
│   ├── tournament.py  
│   └── match.py  
├── logs/  
│   └── tournament.log  
└── data/  
    ├── tournament.json  
    └── matches.csv
```

Descripción de Archivos y Módulos:

main: Este archivo será el punto de entrada del programa. Aquí se manejará la interacción con el usuario y se integrarán los diferentes módulos para gestionar equipos, torneos, y resultados.

classes/team.py: Define la clase `Team`, que representa a un equipo en el sistema. Encapsula toda la lógica de negocio para la creación del equipo. Debe incluir atributos como nombre, alias, victorias, derrotas, y puntos.

classes/tournament.py: Define la clase `Tournament`, que maneja la creación y gestión de torneos, así como la inscripción de equipos.

classes/match.py: Define la clase `Match`, que registra los partidos entre equipos, incluyendo los resultados y la actualización de las estadísticas de los equipos.

db_manager.py: Contiene funciones para guardar y cargar los datos de los torneos, equipos y resultados desde/hacia archivos JSON o CSV.

Logs/tournament.log: Log de eventos.

Data/ directorio para guardar los archivos json y csv creados.

Consideraciones Adicionales

- Log de evento. Se debe utilizar el módulo *logging* para llevar el log de errores. Recuerde utilizar `logging.basicConfig` para definir la ubicación de log.
- Manejo de Errores: Asegúrate de manejar posibles errores como intentar registrar un resultado para un equipo no existente o intentar crear un torneo sin equipos inscritos. Las excepciones de lógica de negocio deben generar una excepción en la clase correspondiente y ser capturada por programa que invoca a la clase.
- Estilo y nombramiento según [PEP 8](#): especialmente en los siguientes temas:
 - o Usa **snake_case** para nombres de variables y funciones
 - o Usa **CamelCase** para nombres de clases
 - o Las constantes se deben nombrar usando **mayúsculas sostenidas** y guiones bajos para separar palabras.
 - o Los comentarios deben ser claros y precisos. Usa `#` para comentarios en línea y `"""` para docstrings en funciones y clases.
 - o Coloca todas las importaciones al inicio del archivo.
 - o Las importaciones deben ordenarse en grupos: primero las librerías estándar, luego las librerías de terceros, y finalmente tus propios módulos.
 - o Cada importación debe estar en una línea separada

Historias de usuario

General: se requiere la funcionalidad backend, no es necesario una interfaz gráfica. Utilice un programa Python para crear los objetos y ejecutar las funciones.

	Historia	Criterios aceptación
1	Como organizador necesito validar si el nombre de un equipo es adecuado para el torneo.	<ul style="list-style-type: none">• El nombre no debe exceder los 40 caracteres (o excepción)• El nombre debe ser de máximo 3 palabras• Se debe generar un alias con las iniciales de cada palabra (para los equipos con nombre de 3 palabras)• Se debe generar un alias con la inicial de la primera palabra y 2 de la segunda palabra para equipos con nombre de 2 palabras• Y se debe utilizar como alias las 3 primeras letras si en nombre tiene una única palabra. <p>Tareas sugeridas</p> <ul style="list-style-type: none">- Crear clase <i>Team</i> con todos sus atributos (name, alias, matches, wins, losses, draws, goals, goals_against, points).- El constructor recibe el nombre- Crear una función para calcular el alias con las reglas definidas en la historia- Hacer las validaciones y levantar la excepción <code>ValueError</code> cuando no se cumplen las reglas de negocio para los nombres-
2	Como organizador del torneo necesito crear un torneo con un nombre y una lista de equipos inscritos para administrar los equipos, encuentros y resultados	<ul style="list-style-type: none">• Debe recibir 4 equipo o levantar una excepción• El nombre del torneo no debe exceder los 40 caracteres (o excepción)• El sistema debe asignar un ID tipo <i>UUID 4</i> utilizando la librería <i>uuid</i>.• El torneo se crea con <i>status</i> = "Active"• Todos los torneos son todos contra todos (round robin). Utilizar un parámetro con valor default en el <i>init</i> de la clase• Al crear el torneo se debe guardar en data/tournament.json como se ve en el siguiente ejemplo <pre>{ "name": "xxxxxxx xxxxxx xxxx", "id": "bd65600d-8669-4903-8a14- af88203add38", "type": "Round robin",</pre>

		<pre> "status": "Active", "teams": [{ "name": "abc efg hij", "alias": "aeh", "matches": "0", "losses": "0", "wins": "0", "goals": "0", "goals against": "0", "draws": "0", "points": "0", }, { "name": "123Orion", "alias": "123", "matches": "0", "losses": "0", "wins": "0", "goals": "0", "goals against": "0", "draws": "0", "points": "0", }], "timestamp": "2024-08-29 07:00:01" "node": "xxxxxx" "python version": "x.xx" } </pre> <ul style="list-style-type: none"> • Al crear o actualizar el archivo debe incluirse la fecha y hora (timestamp), el nombre del computador y versión de python que ejecuta la actualización (usar modulo platform) • La clase Tournament debe devolver un diccionario • Una función en el módulo db_manager.py guarda los datos del diccionario en el archivo json • Se deben generar excepciones de equipos con alias no único, número de equipos diferente a 4. • Se deben generar excepciones si ya existe un archivo tournament.json y está en active • Controlar además las excepciones relacionadas con uso de archivos. • Utilice el logger para dejar registrado la creación a nivel INFO
--	--	---

		<p>Tareas sugeridas</p> <ul style="list-style-type: none"> - Crear Tournament. Con un constructor que reciba el nombre del torneo la lista de equipos - Guardar los equipos como un diccionario (atributo teams de la clase) tomando el alias como key y el objeto Team como valor. - En el constructor obtener id con uuid4 - El tipo de torneo es se recibe como parámetro del constructor con valor default "Round robin" - Valide la regla de 4 equipos. Levante una Excepción ValueError si no se cumple - Complete la función get_teams_data para retornar los datos de la clase como un diccionario - En db_manager cree la función para escribir los datos al archivo tournament.json - Debe generar error si el archivo ya existe
3	Como organizador del torneo necesito programar los partidos de cada fecha	<ul style="list-style-type: none"> • Se deben generar los cruces o partidos para cada una fecha. • Cada partido tiene un Status que inicia en Scheduled, y pasa a Played cuando se ingresa el resultado. • Ordene la lista de equipos de forma aleatoria y luego realice los cruces por las posiciones (0 a 3): <ul style="list-style-type: none"> ○ Semana 1, Partido 1, 0 vs 1 ○ Semana 1, Partido 2, 2 vs 3 ○ Semana 2, Partido 1, 0 vs 2 ○ Semana 2, Partido 2, 1 vs 3 ○ Semana 3, Partido 1, 0 vs 3 ○ Semana 3, Partido 2, 1 vs 2 • Organizar los partidos es una función de la clase <i>Tournament</i> que devuelve una lista de objetos <i>Match</i> • El listado partidos se guardan en el archivo data/matches.csv con los datos: Week (1, 2 o 3), Match (1 o 2), Team 1, Team 2, Goals team 1, Goals team 2, Status (Scheduled / Played) • La función que crea el archivo debe estar en el módulo db_manager.py • Utilice el logger para dejar registrado la creación a nivel INFO • Manejar las excepciones de archivos

		<p>Tareas sugeridas</p> <ul style="list-style-type: none"> - Utilizar la clase Match - Crear la función create_schedule que realice los cruces de los equipos y los inserte en una lista de objetos Match en el atributo matches. - La función retorna una lista de diccionarios con los datos de los partidos. - En db_manager cree la función para escribir los datos al archivo matches.csv. Recibe la lista de diccionarios y guarda en CSV
4	Como organizador necesito ingresar los resultados de los partidos para conservar los datos	<ul style="list-style-type: none"> • Se deben solicitar los resultados de los partidos en orden de semana (week), partido (match) • Los goles de cada equipo, que deben ser enteros positivos • Al ingresar los resultados debo poder detener el ingreso de datos o seguir con el siguiente encuentro • Para guarda una fecha se debe tener resultado de los 2 partidos de esa fecha. No se guardan datos de un solo partido • Los datos deben actualizar el archivo data/matches.json y data/tournamet.json • La función que crea el archivo debe estar en el módulo db_manager.py • Utilice el logger para dejar registrado la creación a nivel INFO • Manejar las excepciones de archivos • Si se ingresan resultados de las 3 fechas, el torneo se considera finalizados y se debe actualizar el estado en data/tournament.json • Si se ingresan resultados de las 3 fechas, el torneo se considera finalizados y se debe actualizar el estado en data/tournament.json status = "Finished"
5	Como organizador necesito seguir ingresando resultados conservando la información de fechas anteriores para ingresar fecha a fecha los datos	<ul style="list-style-type: none"> • Se debe solicitar información en el orden de las fechas (1 a 3) iniciando por la menor fecha que no tenga datos de resultados • Utilice el logger para dejar registrado el ingreso de los resultados a nivel INFO • Se debe actualizar el puntaje de los equipos

		•
6	Como organizador necesito listar las posiciones de los equipos según los resultados ingresados	<ul style="list-style-type: none"> • Listar los equipos ordenados por puntos, diferencia de goles, goles a favor. • Equipos empatados en estas 3 variables deben compartir la misma posición.