Explicación de Conceptos en el Código loteria.py

## Clase (class) y su función

Una clase es una plantilla para crear objetos en Python. En este código, ManejadorDeArchivos es una clase que agrupa funcionalidades relacionadas con el manejo de archivos JSON.

class ManejadorDeArchivos:

# métodos aquí

**Función:** Organiza y encapsula todas las operaciones relacionadas con archivos (cargar, guardar, formatear) en un solo lugar, haciendo el código más modular y mantenible.

## @staticmethod

Un método estático es un método que pertenece a la clase en lugar de a una instancia de la clase. No necesita acceso a self.

@staticmethod

def CargarboletosActivos():

# código

**Por qué se repite:** Cada método que no necesita acceder a los datos de una instancia específica se marca como @staticmethod. En este caso, todos los métodos son estáticos porque la clase solo agrupa funciones utilitarias sin necesidad de estado interno.

## .append()

Es un método de las listas que añade un elemento al final:

historial.append(registro) # Añade 'registro' al final de la lista 'historial'

## json.dump()

Función que escribe datos Python en un archivo en formato JSON:

json.dump(boletos, file, indent=4) # Escribe 'boletos' en 'file' con sangrado de 4 espacios

## .items()

Método de los diccionarios que devuelve pares clave-valor:

for aciertos, cantidad in estadisticas.items():

# Itera sobre las claves y valores del diccionario

## Operador //

División entera (redondea hacia abajo):

CantidadMAX = int(Dinero // Precioboleto) # Calcula cuántos boletos se pueden comprar

## .get()

Método de diccionarios que obtiene un valor por clave, con un valor por defecto si la clave no existe:

Premios.get(aciertos, "No ganó") # Devuelve el premio o "No ganó" si no hay premio para esos aciertos

## random.sample()

Genera una lista de elementos únicos seleccionados al azar:

random.sample(range(1, 50), 6) # 6 números únicos entre 1 y 49

## Uso de b en comprensión de diccionarios

Es una variable temporal en la comprensión:

Usuarios = {b['id']: b['nombre'] for b in boletosActivos}

Esto crea un diccionario donde las claves son los IDs y los valores son los nombres, iterando sobre cada boleto (b) en boletosActivos.

## None

Representa la ausencia de valor. En las funciones, se usa para indicar que no se devolvió nada válido:

if not boletosUsuario:

return None # La función termina sin resultado

## (ValueError, IndexError)

Son tipos de excepciones:

* ValueError: Cuando un valor no es del tipo esperado
* IndexError: Cuando se intenta acceder a un índice que no existe

except (ValueError, IndexError):

# Maneja ambos tipos de errores

## set()

Estructura que almacena elementos únicos. La operación & es intersección:

len(set(Numeroboleto) & set(ganadores)) # Cuenta cuántos números coinciden entre el boleto y los ganadores

## headers y tablefmt="grid"

Parte de la función tabulate():

* headers: Títulos de las columnas
* tablefmt: Formato de la tabla ("grid" crea bordes alrededor de las celdas)

print(tabulate(tabla, headers=["nombre", "id", ...], tablefmt="grid"))

## Resumen de conceptos clave

| **Concepto** | **Explicación** | **Ejemplo en el código** |
| --- | --- | --- |
| **class** | Plantilla para crear objetos | class ManejadorDeArchivos |
| **@staticmethod** | Método que no necesita instancia | @staticmethod def CargarboletosActivos() |
| **.append()** | Añade elemento a lista | historial.append(registro) |
| **json.dump()** | Escribe datos en formato JSON | json.dump(boletos, file) |
| **//** | División entera | Dinero // Precioboleto |
| **.get()** | Obtiene valor de diccionario con valor por defecto | Premios.get(aciertos, "No ganó") |
| **random.sample()** | Selección aleatoria sin repetición | random.sample(range(1,50), 6) |
| **set()** | Conjunto de elementos únicos | set(Numeroboleto) & set(ganadores) |

Estos conceptos trabajan juntos para crear un sistema de lotería que maneja boletos, sorteos y estadísticas de manera organizada.