Examen Módulo 3

16/05/2022

Instrucciones

El presente examen se compone de un solo problema con varias partes incrementales. Tendrás un tiempo de 110 minutos para resolverlo, más 10 minutos para entrega en línea. Te recomendamos revisar rápidamente los requisitos de la solución abajo, y luego ir a la sección titulada "Observaciones sobre la implementación" antes de comenzar a resolver el examen.
Para entregar tu examen, haz un archivo `zip` con nombre cuyo formato debe ser `RUT_sin_verificador.zip`, que contenga el programa desarrollado. En cada enunciado de pregunta se especifica cuál debe ser el nombre de archivo. Entrega tu archivo zip en el buzón disponible en Canvas.

Wandes Music (6 puntos + 1 bonus)

En este examen desarrollarás un programa para organizar una biblioteca de música (colección discográfica), similar a cómo funcionan aplicaciones como Spotify o Apple Music.

Representación de datos

La biblioteca musical tiene la siguiente representación en Python:

Listado 1: Representación de la biblioteca

```
biblioteca = {
        'generos' : [],    # Lista de strings con generos musicales
        'artistas' : {},   # Diccionario con informacion de artistas (la clave es nombre de artista)
        'discos' : [],     # Lista de diccionarios con info de discos
        'canciones' : []   # Lista de diccionarios con info de canciones
    }
```

El siguiente ejemplo muestra cómo es representado un artista en la biblioteca:

Listado 2: Representación de un artista

```
artista = { 'nombre' : 'The Who',       # String con nombre del artista
            'idx_genero' : 26,          # Índice del género en la biblioteca
            'idx_discos': [7,33,21,99]  # Índices de discos en biblioteca
          }
```

En la biblioteca, la clave del diccionario 'artistas' es el nombre del artista, el cual se repite en el campo 'nombre' en el diccionario con la información del artista.
Los discos también son representados como diccionarios en la biblioteca:

Listado 3: Representación de un disco

```
disco = {
          'nombre' : "Who's Next",  # String con nombre del disco
          'fecha' :  '06/11/1971',  # String con formato DD/MM/AAAA
          'idx_canciones' : [543,544,545,546,...] # Índices de caciones en biblioteca
        }
```

Finalmente, cada canción es almacenada en la biblioteca también como diccionario:

Listado 4: Representación de una canción

```
cancion = {
            'titulo' : 'Behind Blue Eyes',  # String con el titulo de la canción
            'duracion' : '00:03:41'         # Duración en HH:MM:SS
          }
```

Funciones a implementar

Se te entrega un código base que implementa toda la interfaz de usuario del programa, y algunas funciones utilitarias. Para mayores detalles sobre el código base, revisa más abajo la sección titulada "Observaciones sobre la implementación".
Debes completar las funciones del código base que se describen a continuación:

1. [1.0 pt] `crear_artista(nombre, genero, biblioteca)`: Si el artista ya existe en la biblioteca (el `nombre` ya está en la biblioteca), retorna `False`. En caso contrario, si el género no existe en la biblioteca, lo agrega. Después crea un diccionario con los datos de nombre y genero del artista y lo incorpora a la biblioteca. Retorna `True`.

2. [1.0 pt] `crear_disco(nombre, fecha, artista, biblioteca)`: Crea un disco con la representación en el Listado 3 y lo agrega a la biblioteca. Además, agrega el disco a la lista de índices de discos del artista respectivo (ver `idx_discos` en el Listado 2). Retorna el índice con que ha quedado guardado el disco en la lista de discos de la biblioteca.

3. [1.0 pt] `mostrar_disco(idx_disco, biblioteca)`: Despliega toda la información de un disco, incluyendo el artista y su género musica, el título del disco y su fecha de lanzamiento, y luego la lista de canciones, incluyendo el número de pista (suponer que el orden de índices en lista `idx_canciones` de un disco es el orden de las pistas), y duración de cada canción. Ejemplo:

   ```
   Artista: Limos / Género: drum and bass
   Disco: Complainer (remixes) / Fecha: 11/2007
   1. Rebel Assault 00:06:20
   2. Just a Moment 00:03:33
   3. Voice 00:04:45
   4. Forever 00:04:05
   5. Como haré para olvidar 00:07:07
   6. The Rain 00:05:29
   7. Contact 00:02:19
   8. "Le Fils des étoiles : La Vocation, prélude à l'acte I" 00:12:25
   ```

4. [1.0 pt] `buscar_idx_disco_por_idx_cancion(idx_cancion, biblioteca)`: Dado el índice `idx_cancion` de una canción en la biblioteca, busca el disco en `biblioteca` cuyo campo `idx_canciones` (ver Listado 3) contiene `idx_cancion`. Retorna una lista con dos elementos: (1) el índice del disco en la biblioteca, y (2) el diccionario con los datos del disco.

5. [1.0 pt] `buscar_por_subpalabra(subpalabra, biblioteca)`: Esta función de búsqueda retorna un diccionario con la siguiente estructura:

Listado 5: Resultados de búsqueda

```
    resultados = { 'artistas' : [],
                   'idx_discos' : [],
                   'idx_canciones' : [] }
```

La lista asociada a la clave `artistas` es una lista de strings, que debe contener los nombres de todos los artistas cuyo nombre contenga `subpalabra`. Por ejemplo, si `subpalabra` es `Jam`, la lista podría contener `'Perl Jam'`. Si no hay coincidencias, la lista queda vacía. La lista `idx_discos` en el diccionario en el Listado 5 debe contener los índices de todos los discos en la biblioteca cuyo nombre contenga la `subpalabra` buscada, o quedar vacía de no existir coincidencias. Finalmente, la lista `idx_canciones` debe contener los índices de todas las canciones en la biblioteca cuyo título coincida con `subpalabra`, o quedar vacía si no hay coincidencias.
Consejo: Usar la función `find` de strings, que retorna el índice en donde comienza la ocurrencia del substring buscado, o -1 si no existe ninguna ocurrencia.

6. [1.0 pt] `buscar_artistas_por_genero(genero, biblioteca)`: Busca en la biblioteca todos los artistas cuyo género coincide con `genero`. Agrega los artistas encontrados (diccionarios de acuerdo a Listado 2) a la lista y retorna la lista. La lista retornada puede ser vacía si el género no existe o no se encuentran artistas relacionados con el género.

7. [1.0 pt] - BONUS `obtener_lista_aleatoria_reproduccion_por_genero(genero, tiempo_min, biblioteca)`: Retorna una lista con índices de canciones (según `biblioteca`) escogidos al azar, cuyos artistas corresponden a `genero`, y en donde la duración total de las canciones escogidas, agregada la última canción, supera `tiempo_min` en segundos.
Consejos: (1) Obtener todos los artistas por género requerido. (2) Para cada artista, escoger aleatoriamente sus discos y canciones (usar `random.choice(l)` que escoge y retorna a azar un elemento de la lista `l`), e ir manteniendo registro del tiempo total. (3) Para convertir la duración de las canciones en formato string `HH:MM:SS` a segundos, puedes usar la función `tiempo_a_segundos(hhmmss)` que recibe un string con tiempo y retorna los segundos equivalentes (`int`). (4) Iterar por los artistas y discos hasta que se satisface la condición del tiempo mínimo. Limitar las iteraciones a un máximo (p.ej., 1000) para evitar un loop infinito.

Observaciones sobre la implementación

1. Puedes ver una demostración de funcionamiento del programa aquí `https://youtu.be/5X-6UQakBHQ`. Evidentemente, las funciones que aparecen en el menú dependen de las funciones que tú debes implementar en este examen. El programa del código base tiene mínima funcionalidad.

2. No está permitido modificar la firma (nombre y parámetros) de las funciones que debes implementar. Sí está permitido agregar otras funciones.

3. Puedes desactivar la interfaz de usuario comentando la última línea del programa con código `mostrar_menu()`. Ello te ayudará a probar las funciones que implementes sin necesidad de utilizar toda la interfaz.

4. Si necesitas usar una biblioteca musical llena de datos para probar tus funciones, puedes usar la biblioteca `bb.biblioteca` en tu programa. Esta biblioteca tiene miles de canciones y cientos de discos.

Sistema de Evaluación

Cada ítem es evaluado con una escala de 1-5 (1: no presenta solución, 2: presenta esbozo de solución incompleto o con errores graves, 3: solución incompleta o imprecisa, 4: error(es) mínimo(s), 5: excelente) y luego un ponderador (1:0, 2:0,25, 3:0,5, 4:0,75, 5:1,0) es aplicado al puntaje del ítem. El puntaje otorgado a la solución se calcula como la suma de los puntajes parciales obtenidos, aplicándose los ponderadores. La nota final es el puntaje calculado más el punto base.