### **Examen: Construyendo el Sistema de Recomendación de PythonFlix**

**Contexto**: Eres parte del equipo de ingeniería de **PythonFlix**, una plataforma de streaming que necesita construir y mejorar su sistema de recomendación. Vas a trabajar con estructuras de datos como listas, tuplas, diccionarios y sets para procesar ratings, historiales y perfiles de usuario.

#### **1. Catálogo de contenido (Listas y tuplas)**

Tienes una tupla inmutable con las series más vistas:

catalogo\_popular = ("Breaking Python", "Data Science 101", "Machine Love", "Debug Wars")

Convierte esta tupla en una lista y realiza lo siguiente:

* Agrega una nueva serie: "Neural Networks"
* Reemplaza "Debug Wars" por "AI Rebellion"
* Ordena la lista alfabéticamente

#### 

#### 

#### **2. Filtrado de géneros (Slicing)**

Tienes una lista con los géneros disponibles:

generos = ["Drama", "Comedia", "Acción", "Sci-Fi", "Documental", "Terror", "Romance"]

Realiza:

* Un *slicing* que obtenga los géneros del medio
* Un *slicing* que obtenga los tres primeros géneros
* Los géneros ordenados en orden inverso (usando slicing)

#### **3. Puntajes de usuarios (Listas por comprensión)**

Tienes una lista de ratings otorgados por un usuario a distintas series:

ratings = [4.5, 3.0, 5.0, 2.5, 4.0, 1.5]

Con listas por comprensión, crea:

* Una lista con solo los ratings mayores a 3.5
* Una lista con los ratings convertidos a una escala del 1 al 100
* Una lista de strings que diga "Alta" si rating > 4, "Media" si entre 3 y 4, "Baja" si menor

#### **4. Historial de visualización (Diccionarios)**

Tienes un diccionario que representa el historial de un usuario:

historial = {

"Breaking Python": 3,

"Machine Love": 5,

"Data Science 101": 2

}

Realiza lo siguiente:

* Aumenta en 1 la cantidad de veces que vio "Data Science 101"
* Añade la serie "AI Rebellion" con 1 visualización
* Elimina "Machine Love" del historial
* Muestra todas las series que el usuario vio más de 2 veces

#### **5. Usuarios únicos por serie (Sets)**

Tienes la siguiente información:

usuarios\_serie\_A = {"ana", "luis", "pedro", "maria"}

usuarios\_serie\_B = {"maria", "pedro", "carla", "lucas"}

Usando sets:

* Encuentra qué usuarios vieron ambas series
* Qué usuarios vieron solo la serie A
* Qué usuarios únicos vieron al menos una de las dos

#### **6. Recomendaciones compartidas (Diccionario + Sets)**

Tienes una estructura con recomendaciones por usuario:

recomendaciones = {

"ana": {"Machine Love", "AI Rebellion"},

"luis": {"AI Rebellion", "Debug Wars"},

"pedro": {"Debug Wars", "Machine Love"}

}

Encuentra con código:

* Las series que fueron recomendadas a todos
* Las series recomendadas a al menos dos personas
* Las series recomendadas solo a una persona

#### **7. Transformar ratings (Tuplas + comprensión de diccionario)**

Tienes una lista de tuplas con (serie, rating):

puntuaciones = [("Breaking Python", 4.0), ("AI Rebellion", 5.0), ("Debug Wars", 2.0)]

Crea un diccionario en el formato:

{

"Breaking Python": "Buena",

"AI Rebellion": "Excelente",

"Debug Wars": "Regular"

}

Regla:

* Rating >= 4.5 → "Excelente"
* 3 <= rating < 4.5 → "Buena"
* < 3 → "Regular"

#### **8. Usuarios y géneros favoritos (Diccionario + slicing)**

Cada usuario tiene una lista de sus géneros preferidos (ordenados por preferencia):

gustos = {

"ana": ["Sci-Fi", "Drama", "Documental"],

"luis": ["Comedia", "Acción", "Sci-Fi"],

"pedro": ["Drama", "Sci-Fi", "Romance"]

}

Realiza:

* Una lista con el primer género favorito de cada usuario
* Una lista con todos los géneros mencionados sin repeticiones
* ¿Cuál género aparece más veces en la primera posición?

#### **9. Resumen de votos por serie (Diccionario + listas)**

Tienes una lista de votos como pares (serie, puntaje):

votos = [("Breaking Python", 5), ("AI Rebellion", 4), ("Breaking Python", 4), ("Debug Wars", 3), ("AI Rebellion", 5)]

Crea un diccionario donde la clave sea la serie y el valor una lista con todos sus votos. Ejemplo:

{

"Breaking Python": [5, 4],

...

}

Luego, calcula el promedio de rating de cada serie.

#### **10. Motor básico de recomendaciones (Integrador)**

Define una función recomendar\_series(usuario, historial, catalogo) que:

* Reciba el nombre de usuario, su historial en forma de diccionario {serie: vistas} y un catálogo con todas las series disponibles.
* Devuelva una lista de series del catálogo que **no haya visto aún**.

Ejemplo:

usuario = "ana"

historial = {"AI Rebellion": 3, "Debug Wars": 1}

catalogo = ["AI Rebellion", "Debug Wars", "Machine Love", "Data Science 101"]

Resultado:

["Machine Love", "Data Science 101"]