



Actividad 3

Iterables y Generadores

Entrega

- **Lugar:** Repositorio personal de GitHub — Carpeta: **Actividades/AC3**
El código debe estar en la rama (*branch*) por defecto del repositorio: **main**.
- **Fecha máxima de entrega:** 13 de Mayo 20:00

Introducción

Preparándose para disfrutar su fin de semana, todos los miembros del DCC se proponen descansar viendo sus películas favoritas. Con un montón de cabritas saladas y listos para empezar una buena noche de cine, notan que una entidad maligna ha atacado todos los servicios de *stream* presentes en el internet y ahora nadie puede disfrutar de su preciado tiempo libre.

Agobiados por esta situación, tú junto a un grupo estudiantes del DCC se deciden a programar su propia plataforma de *stream*: **DCC Max**. Para ayudarlos, estarás encargado de: cargar la información de las películas y sus categorías, implementar una serie de consultas para operar sobre la información cargada y, finalmente, implementar el catálogo de *DCC Max* para así poder revisar todas las películas disponibles.

Archivos

En el directorio de la actividad encontrarás los siguientes archivos:

Archivos de datos

- **No modificar** `archivos/peliculas.csv`: Este archivo contiene la información de las películas disponibles. El formato del archivo es:

`id,titulo,director,año_estreno,rating_promedio`

donde `id` y `año_estreno` corresponden a números enteros, mientras que `rating_promedio` corresponde a un número decimal.

- **No modificar** `archivos/generos.csv`: Este archivo contiene la información de todos los géneros de las películas del archivo anterior. Una misma película puede estar relacionada con uno o más géneros. El formato del archivo es:

`genero,id_pelicula`

donde `id_pelicula` corresponde a un número entero.

Archivos de código

- **Entregar** **Modificar** `funciones.py`: Contiene las funciones necesarias para cargar y manejar la información.
- **No modificar** `utilidades.py`: Contiene la definición de *namedtuples* y funciones necesarias para cargar y manejar la información.
- **No modificar** `parametros.py`: Contiene parámetros necesarios para ejecutar el programa.
- **No modificar** `test.py`: Contiene el código necesario para ejecutar los *tests* relacionados a las funcionalidades.

Flujo del programa

Esta actividad consta de tres partes, en las cuales se te pedirá que implementes funciones que permitan cargar información, realizar consultas y manejar un catálogo de películas.

1. Parte 1 – Cargar datos

Para que puedas implementar correctamente las funcionalidades, te entregamos las siguientes *namedtuples* ya implementadas en el módulo `utilidades.py`:

- **No modificar** `Pelicula`: Posee los atributos `id_pelicula` (`int`), `titulo` (`str`), `director` (`str`), `estreno` (`int`) y `rating` (`float`).
- **No modificar** `Genero`: Posee los atributos `genero` (`str`) y `id_pelicula` (`int`).

Para cargar los datos y utilizar las *namedtuples* anteriores, deberás completar las siguientes funciones del archivo `funciones.py`:

- **Modificar** `def cargar_peliculas(ruta: str) -> Generator`:
Esta función generadora recibe un `str` con la ruta del archivo y va retornando instancias de `Pelicula` según el contenido del archivo.
- **Modificar** `def cargar_generos(ruta: str) -> Generator`:
Esta función generadora recibe un `str` con la ruta del archivo y va retornando instancias de `Genero` según el contenido del archivo.

2. Parte 2 – Consultas

Para poder manejar el *DCC Max*, deberás completar una serie de consultas que trabajarán sobre los datos cargados. **Importante:** Se encuentra **estrictamente prohibido** el uso de: ciclos `for` y `while`; estructuras de datos `list`, `tuple`, `dict`, `set`; cualquier tipo de estructura creada por comprensión; y otras librerías diferentes a las dadas en los archivos base.

Para lograr implementar las funciones anteriores, ten en mente el uso de funciones `lambda`, pero considera que el énfasis está en el uso de las funciones `map`, `filter` y `reduce`.

Las funciones a completar en el archivo `funciones.py` son:

- **Modificar** `def obtener_directores(generator_peliculas: Generator) -> set:`
Recibe un generador con instancias de `Pelicula` y retorna un objeto de tipo `set` con los nombres de todos los directores.

Para transformar un generador en un `set`, puede utilizar la función ya implementada:

- **No modificar** `def obtener_unicos(generator: Generator) -> set:`
Recibe un generador y retorna un `set` con la misma información entregada por el generador, pero sin elementos duplicados.

Importante: Esta función, únicamente se puede utilizar en la función `obtener_directores`.

- **Modificar** `def obtener_str_titulos(generator_peliculas: Generator) -> str:`
Recibe un generador con instancias de `Pelicula` y retorna un `string` con todos los títulos de las películas concatenados por una coma y un espacio `(" , ")`.
- **Modificar** `def filtrar_peliculas(generator_peliculas: Generator, director: str | None, rating_min: float | None, rating_max: float | None) -> filter:`
Recibe un generador con instancias de `Pelicula`, además, puede recibir el nombre de un director, un `rating` mínimo o un `rating` máximo. Retorna un objeto de tipo `filter` con las películas filtradas.

Las películas se filtran de forma que, en caso de haberse indicado:

- El nombre de un director: se filtran las películas de forma que solo queden las películas que tengan el mismo director que el indicado.
 - Un `rating` mínimo: se filtran las películas de forma que solo queden las películas que tengan un `rating` equivalente o mayor al entregado.
 - Un `rating` máximo: se filtran las películas de forma que solo queden las películas que tengan un `rating` equivalente o menor al entregado.
- **Modificar** `def filtrar_peliculas_por_genero(generator_peliculas: Generator, generator_generos: Generator, genero: str | None) -> filter:`
Recibe un generador con instancias de `Pelicula`, un generador con instancias de `Generos` y puede recibir el nombre de un género de película. Retorna un objeto de tipo `filter` que contiene todos los pares del generador de películas y el generador de géneros que:
 1. Correspondan a la misma película, es decir, que ambos elementos del par tengan el mismo id de película.
 2. El género corresponda al indicado en el *input*. Si no se indica un género, entonces solo se deben retornar todos los pares que cumplen con el punto 1.

Para lograr lo anterior, deberás investigar y utilizar la función `product` de la librería `itertools`.

3. Parte 3 – *DCC Max*

En esta última parte, deberás aplicar tus conocimientos de **iterables personalizados**, para implementar la clase `IteradorDCCMax`, que corresponde al iterador de la clase `DCCMax`.

Ambas clases se encuentran en el archivo `funciones.py` y la intención de estas es poder recorrer las películas de forma ordenada según su año de estreno y *rating* promedio. Para esto:

1. El iterador entregará las películas según su año de estreno ordenadas de forma ascendente.
2. En caso de que hayan 2 o más películas con el mismo año de estreno, se ordenarán las películas según su *rating* promedio de forma descendente.

Importante: Para lograr lo anterior, se encuentra **estrictamente prohibido** el uso de ciclos `for` y `while`.

Los métodos a completar en las clases `DCCMax` e `IteradorDCCMax` son:

- `class DCCMax`: Esta clase recibe como argumento una lista de películas.
 - **Modificar** `def __iter__(self)`: Este método retorna una instancia `IteradorDCCMax` con los datos correspondientes.
- `class IteradorDCCMax`: Esta clase recibe como argumento una lista de instancias de `Pelicula` y guarda una copia de esta como atributo (`peliculas`).
 - **Modificar** `def __iter__(self)`: Este método debe retornar a la instancia misma del iterador (`self`).
 - **Modificar** `def __next__(self)`: Este método es el encargado de encontrar y entregar la siguiente instancia de película. Una vez que no hayan más elementos que retornar, se levanta la excepción correspondiente de este iterador.

Notas

- No puedes hacer *import* de otras librerías externas a las entregadas en el archivo.
- Cada una de las tres partes de la actividad puede realizarse en el orden que estimen conveniente.
- Recuerda que la ubicación de tu entrega es en **tu repositorio de Git**. En la rama (*branch*) por defecto del repositorio: `main`.
- Recuerda que esta evaluación presenta corrección **automatizada**. Si entregas un código que se cae al momento de correr los *tests*, será evaluado con 0 puntos.
- Puedes probar tu código con los *tests* y ejecutando `funciones.py`, donde podrás visualizar todas las funcionalidades de `DCCMax`.
- Si aparece un error inesperado, ¡léelo! Intenta interpretarlo y/o buscarlo en Google.

Objetivo de la actividad

- Implementar una función generadora, utilizando correctamente `yield`.
- Aplicar conocimientos de iterables utilizando funciones `map`, `filter` y `reduce`.
- Crear un iterable y un iterador personalizado definiendo correctamente los métodos `__iter__` y `__next__`.

- Investigar y utilizar librerías *built-ins*.

Ejecución de *tests*

En esta actividad se provee de `test.py` que contiene diferentes *tests* que ayudan a validar el desarrollo de la actividad. Para ejecutar este archivo, desde la terminal/consola debes escribir `python3 test.py` y se ejecutarán todos los *tests* de la actividad. Los puntos finales se asignarán según un conjunto de test privados. **Importante:** En esta Actividad los *tests* también verificarán que no se usen los ciclos `for` y `while`.

En cambio, si deseas ejecutar un subconjunto de *tests*, puedes ejecutar lo siguiente en la terminal/consola:

- `python3 -m unittest -v test.TestCargarDatos`: para ejecutar solo el subconjunto de *tests* relacionado a cargar películas y géneros en la Parte 1.
- `python3 -m unittest -v test.TestConsultas`: para ejecutar solo el subconjunto de *tests* relacionado a obtener directores y títulos de la Parte 2.
- `python3 -m unittest -v test.TestFiltros`: para ejecutar solo el subconjunto de *tests* relacionado a filtrar películas de la Parte 2.
- `python3 -m unittest -v test.TestIterables`: para ejecutar solo el subconjunto de *tests* relacionado a las clases `DCCMax` e `IteradorDCCMax` de la Parte 3.

Importante: recuerda que si `python3` no funciona, probar con el comando específico de tu computador. Este puede ser `py`, `py3` o `python`.