Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2024-1

Publicación: 12 de Septiembre de 2024
Actividad evaluada

Actividad 3
Programación Funcional

Entrega

- **Lugar:** Repositorio personal de GitHub — Carpeta: `Actividades/AC3`
- **Fecha máxima de entrega:** 26 de Septiembre 23:59
- **Ejecución de actividad**: La Actividad será ejecutada **únicamente** desde la terminal del computador. **Los *paths* relativos utilizados en la Actividad deben ser coherentes con esta instrucción, y no pueden modificarse.**

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.

Flujo del programa

Esta actividad consta en completar 2 partes aplicando el paradigma de Programación Funcional. La primera referente a la carga de datos a partir de un archivo, preprocesar y retornarlos en un formato específico. Luego, la segunda parte es completar una serie de funciones para consultar los datos. Todas estas partes serán corregidas exclusivamente mediante el uso de *tests*.

Finalmente, debes asegurarte de entregar, como mínimo, el archivo que tenga el *tag* de `Entregar` en la siguiente sección. Los demás archivos no es necesario subir, pero tampoco se penalizará si se suben al repositorio personal.

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. Al finalizar la actividad, debes asegurar que este archivo esté subido en el lugar de entrega correspondiente.
- No modificar `utilidades.py`: Contiene la definición de *namedtuples* y funciones necesarias para cargar y manejar la información.
- No modificar `main.py`: Contiene código para ejecutar las diferentes funciones sin el uso de *tests*.
- No modificar `tests_publicos`: Carpeta que contiene diferentes `.py` para ir probando si lo desarrollado hasta el momento cumple con lo esperado. **En la última hoja del enunciado se encuentra un anexo de cómo ejecutar los *tests* por parte o todos**.

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 la siguientes funciones del archivo `funciones.py`:

- Modificar `def cargar_generos(ruta: str) -> Generator:`

  Esta función generadora recibe un `str` con la **ruta de un archivo** con información de los géneros. Es importante destacar que este argumento es la **ruta**, por ejemplo, `archivos/generos.csv`, `tests_publicos/generos.csv`, `generos_11.csv`, es decir, no es únicamente el nombre del archivo. El trabajo de esta función generador es abrir el archivo con *encoding* `"UTF-8"` e ir entregando instancias de `Genero` según el contenido del archivo.

  Debes asegurarte que los atributos de cada género sean guardados como su tipo de dato correspondiente y de utilizar la *namedtuple* correspondiente del archivo `utilidades.py`.

- Modificar `def cargar_peliculas(ruta: str) -> Generator:`

  Esta función generadora recibe un `str` con la **ruta de un archivo** con información de las películas. Es importante destacar que este argumento es la **ruta**, por ejemplo, `archivos/peliculas.csv`, `tests_publicos/archivo.csv`, `peliculas_2.csv`, es decir, no es únicamente el nombre del archivo. El trabajo de esta función generador es abrir el archivo con *encoding* `"UTF-8"` e ir entregando instancias de `Pelicula` según el contenido del archivo.

  Debes asegurarte que los atributos de cada película sean guardados como su tipo de dato correspondiente y de utilizar la *namedtuple* correspondiente del archivo `utilidades.py`.

2. Parte 2 – Consultas

Para poder manejar el *DCC Max*, deberás completar una serie de consultas que trabajarán sobre los datos cargados.

Importante: En esta segunda parte de la actividad, se espera que apliquen exclusivamente los contenidos de Programación Funcional y un uso correcto de generadores. Para lograr este objetivo, se espera que apliquen correctamente diferentes funciones como `map`, `filter` y `reduce`, uso de funciones anónimas (`lambda`) e *itertools*.

Además, para forzar la aplicación exclusiva de los contenidos, es que 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.

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

- Modificar `def obtener_directores(generador_peliculas: Generator) -> Generator:`

  Recibe un generador con instancias de `Pelicula` y retorna un objeto de tipo `map` con los nombres de todos los directores, sin importar si están repetidos.

- Modificar `def obtener_estrenos(generador_peliculas: Generator, estreno: int) -> Generator:`

  Recibe un generador con instancias de `Pelicula` y un `int` con un año de entreno. Esta función retorna un generador con el título de todos las películas cuyo año de entreno sea igual o mayor al entregado en el parámetro `estreno`.

- Modificar `def obtener_str_titulos(generador_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 (`", "`). Si no hay películas por concatenar, se retorna un string vació (`""`)

- Modificar `def filtrar_peliculas(generador_peliculas: Generator, director: str | None, rating_min: float | None, rating_max: float | None) -> Generator:`

Recibe un generador con instancias de `Pelicula`, además, puede recibir el nombre de un director, un *rating* mínimo o un *rating* máximo. Retorna un `Generator` 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(generador_peliculas: Generator, generador_generos: Generator, genero: str | None) -> Generator:`

  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 `Generator` 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, entones 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`.

- **Modificar** `def filtrar_titulos(generador_peliculas: Generator, director: str, rating_min: float, rating_max: float) -> str:`

  Recibe un generador con instancias de `Pelicula`, el nombre de un director, un *rating* mínimo o un *rating* máximo. Esta función primero filtra las películas para seleccionar solo aquellas que tengan el mismo director que el indicado, tengan un *rating* igual o mayor al `rating_min` y un *rating* igual o menor al `rating_max`.

  Finalmente, retorna un *string* con todos los títulos de las películas filtradas. Los títulos deben estar concatenados por una coma y un espacio (`", "`). Si no hay películas por concatenar, se retorna un string vació (`""`)

Notas

- **No puedes hacer *import* de otras librerías externas a las entregadas en el archivo a completar.**
- 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 `main.py`.
- 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`.
- Utilizar librerías *built-ins*.

Ejecución de *tests*

En esta actividad se provee de varios archivos `.py` los cuáles contiene diferentes *tests* que ayudan a validar el desarrollo de la actividad. **Importante:** En esta Actividad los *tests* también verificarán que no se usen los ciclos `for` y `while`, o uso de estructuras prohibidas.

Para ejecutar estos *tests*, **primero debes posicionar tu terminal/consola en la carpeta de la actividad (`Actividades/AC3`)**. Luego, desde esta misma, debes escribir el siguiente comando para ejecutar todos los *tests* de la actividad:

- `python3 -m unittest discover tests_publicos -v -b`

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

- `python3 -m unittest -v -b tests_publicos.test_cargar_datos`
  Para ejecutar solo el subconjunto de *tests* relacionado a la parte 1.
- `python3 -m unittest -v -b tests_publicos.test_obtener_directores`
  Para ejecutar solo el subconjunto de *tests* relacionado a la consulta de `obtener_directores`.
- `python3 -m unittest -v -b tests_publicos.test_obtener_estrenos`
  Para ejecutar solo el subconjunto de *tests* relacionado a la consulta de `obtener_estrenos`.
- `python3 -m unittest -v -b tests_publicos.test_obtener_str_titulo`
  Para ejecutar solo el subconjunto de *tests* relacionado a la consulta de `obtener_str_titulos`.
- `python3 -m unittest -v -b tests_publicos.test_filtrar_peliculas`
  Para ejecutar solo el subconjunto de *tests* relacionado a la consulta de `filtrar_peliculas`.
- `python3 -m unittest -v -b tests_publicos.test_filtrar_peliculas_genero`
  Para ejecutar solo el subconjunto de *tests* relacionado a la consulta de `filtrar_peliculas_por_genero`.
- `python3 -m unittest -v -b tests_publicos.test_filtrar_titulos`
  Para ejecutar solo el subconjunto de *tests* relacionado a la consulta de `filtrar_titulos`.

Importante: recuerda que si `python3` no funciona, probar con el comando específico de tu computador. Este puede ser `py`, `python`, `py3` o `python3.11`.