Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2020-1

19 de marzo de 2020
Actividad Formativa

Actividad Formativa 01

Estructuras de datos *built-ins*

Antes de comenzar...

Para esta, y todas las actividades del semestre, como equipo docente esperamos que sigas el siguiente flujo de trabajo:

- Lee el enunciado completo, incluyendo las notas. Puedes revisar los archivos subidos a medida que lees, o al final, como te acomode.
- Antes de comenzar a programar, copia y pega todos los archivos de la carpeta AF01 del *Syllabus* y pégalos en la misma carpeta de **tu repositorio personal**.
- Haz `git add`, `git commit` y `git push` de los archivos copiados inmediatamente, para comprobar que el uso de Git esté funcionando correctamente. (***Tip:*** Si no recuerdas como utilizar estos comandos, puedes revisar el enunciado de la AC00)
- En caso de encontrar un error, contacta a un ayudante para resolver el problema lo antes posible, en caso de no hacerlo, tu actividad podría no entregarse correctamente.
- Comienza a trabajar en tu actividad en tu repositorio y recuerda subir cada vez que logres un avance significativo (con los mismos comandos de Git de antes).
- Todas las actividades y tareas tienen una fecha y hora de entrega, en la cual automáticamente se recolecta el último *commit* ***pusheado*** en tu repositorio. Esto no quiere decir que solo se consideran los cambios de ese último *commit*, si no que todos los avances **hasta** ese *commit*. Luego, es importante realices `git push` de todos tus avances, antes de la fecha y hora de entrega. En este caso, es a las 16:50 de hoy.

Introducción

Durante la última semana se decretó estado de cuarentena en Chile, y como estar encerrado en casa es particularmente aburrido, el DCC (Departamento de Ciencias de la Computación) decidió aprovechar la oportunidad y crear su propia página para hacer *streaming* de anime. Tu deber como programador ~~otaku~~ estrella, es el de apoyar el desarrollo de la aplicación y terminarla, de manera que la gente tenga ganas de quedarse en sus casas, ~~se vuelva otaku~~ y no infecte a los demás.

Archivos

Para esta actividad se te hará entrega de los siguientes archivos:

- `main.py`: Este es el archivo principal del programa. Puedes ejecutarlo para probar el funcionamiento de tu programa completo. **Ya viene implementado, y no debe ser modificado**.
- `cargar.py`: En este archivo están las bases de funciones encargadas de cargar los datos. Deberás completarlas.
- `consultas.py`: En este archivo están las bases de funciones encargadas de consultar los datos. Deberás completarlas.
- `animes.csv`: En este archivo de texto se encuentran todas las series que tiene el sistema. Cada línea sigue el siguiente formato:

  ```
  nombre,rating,estudio,genero_1/.../genero_n
  ```

  Donde `nombre`, `genero_1,...,genero_n` y `estudio` son `str` y `rating` es `int`.
- `consultas.csv`: En este archivo de texto se encuentran todas las consultas a realizar. Cada línea sigue el siguiente formato:

  ```
  tipo_consulta;argumento_1;...;argumento_n
  ```

  Donde `tipo_consulta` es un `int` y los argumentos pueden ser de cualquier tipo.

Lectura de Archivos

Para poder leer los archivos deberás modificar las funciones presentes en el archivo `cargar.py`, sin embargo **no podrás hacer uso de clases para guardar esta información**. Las funciones son las siguientes:

- `def cargar_animes(ruta)`: Está función cargar los animés del sistema. Debe retornar un diccionario de `namedtuples`, donde las llaves son nombres de los distintos animé y el valor correspondiente una `namedtuple` con su información. También debes asegurarte que los atributos de cada anime sean guardados como el tipo de objeto correspondiente.

  Finalmente, hay un pequeño error con la base de datos, varios animes tienen generos repetidos. Por lo cual debes guardar estos como un `set` para solucionar el problema.
- `def cargar_consultas(ruta)`: Esta función carga las consultas que se le harán al sistema. Debe retornar una cola[1] de tuplas, donde las tuplas serán de la forma `(tipo_consulta, argumentos)` donde `argumentos` es una lista con los argumentos.

Consultas

La segunda parte de tu trabajo es implementar distintas consultas para poder extraer información relevante de la base de datos entregada. En específico tendrás que implementar las siguientes consultas:

- `def cantidad_animes_genero(animes)`: Esta función recibirá una lista de (tuplas de) animes. Deberás retornar el número de animes por género como un `dict`, donde cada llave es un genero y su valor es la cantidad de animes con ese género, un ejemplo seria:

  ```
  {"genero_1": 3, "genero_2": 2 ,..., "genero_n": 1}
  ```

[1] Recuerda que para crear colas puedes utilizar `deque` del módulo `collections`

- `def generos_distintos(anime, animes):` Está función recibirá un anime y una lista de animes. Debes retornar un `set` con todos los géneros que el anime no tiene pero que sí están presentes en los géneros de la lista de animes.

  Por ejemplo, si la respuesta a esta consulta es `"Shonen"` y `"Moe"` significa que el anime entregado no tenía esos géneros, pero estos si estaban presentes en la lista de animés.

- `def promedio_rating_genero(animes):` Esta función recibirá una lista de animes y deberás guardar en un `dict` el *rating* promedio de los animes por cada genero y retornarlo, un ejemplo sería:

```
{"Seinen": 6.5, "Shonen": 4.2, ...}
```

Notas

- Se agregaron *prints* en el código base para que puedas revisar tu avance.
- Si aparece un error inesperado, ¡leelo! Intenta interpretarlo.
- Siéntete libre de agregar *prints* en cualquier lugar para revisar objetos, modificaciones, etc... Es una muy buena herramienta para arreglar errores.
- Se imprimen las consultas y sus resultados. Puedes revisar en base a los argumentos la correctitud de tus implementaciones.

Requerimientos

Este puntaje es solo referencial. Al ser una actividad formativa, no hay nota directa asociada.

- (2.00 pts) `cargar.py`
  - (1 pts) `def cargar_animes`: Completa el código y carga los animes de manera correcta. (0.5 pts) Usa la estructura de datos adecuada (0.5).
  - (1 pts) `def cargar_consultas`: Carga las consultas de manera correcta (0.5 pts) y usa la estructura de datos adecuada (0.5 pts).
- (4.00 pts) `consultas.py`
  - (1 pts) `def cantidad_animes_genero`: Recibe la estructura e imprime de manera correcta el numero de animes esperado.
  - (1.5 pts) `def generos_distintos`: Usa las estructuras pedidas (0.5 pts) y obtiene los animes esperados (1.0 pts).
  - (1.5 pts) `def promedio_rating_genero`: Demostrar manejo en la estructura pedida (1.0 pts) e implementa un código que permita realizar un cruce de información de manera correcta. (0.5 pts)

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AF01/`
- **Hora del *push*:** 16:50