Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2025-2

Publicación: 13 de noviembre de 2025
Actividad evaluada

Actividad 7
Tópicos Avanzados I

Entrega

- **Lugar:** Repositorio personal de GitHub — Carpeta: `Actividades/AC7`
- **Fecha máxima de entrega:** 13 de noviembre 17:20
- **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.**

Importante: Antes de comenzar, comprueba que Git este funcionando correctamente en tu repositorio privado. Para esto, **sube los archivos base de la actividad de inmediato** (*add*, *commit*, *push*). Se espera que en esta actividad (así como en las demás actividades y tareas) utilices Git a lo largo de **todo tu desarrollo** como una herramienta, no sólo como un método de entrega. Es por esto que recomendamos enfáticamente que vayas subiendo tus cambios constantemente (***push***), ya que **problemas de último minuto** relacionados con la entrega y Git **no serán considerados**.

Objetivo de la actividad

- Utilizar la librería `pandas` para cargar y procesar información.
- Aplicar conocimientos de representación y búsqueda en Grafos.
- Aplicar conocimientos de Expresiones Regulares mediante el uso de la librería `regex`.
- Probar código mediante la ejecución de *test*.

Introducción

Durante un fin de semana tranquilo, te sienta a tomar un té viendo las fotos de todas tus lindas y adorables mascotas que conociste gracia a tu aplicación para compartir fotos.

En eso, notas que una de ellas es Pepa, una tortuga que ha vivido más de 70 años. Asombrado/a por la longevidad de este reptil, te propones investigar los distintos eventos que ella puede haber vivido durante su vida: ¿Qué fue primero...? ¿Pepa o Google? ¿Pepa o el Internet moderno? ¿Pepa o el DCC?

Como buen Chileno, no esperas más y empiezas a investigar los distintos sismos que han ocurrido en nuestro país, para así descubrir los eventos telúricos que ha vivido nuestra querida tortuga.

Flujo del programa

Esta actividad consta de un único archivo en el deberás completar 6 funciones utilizando los contenidos de Tópicos Avanzados I, por lo que se espera que resuelvas las distintas partes de la actividad con un correcto uso de los contenidos de dicha semana. Todas las funciones serán corregidas exclusivamente mediante el uso de *tests*.

Deberás asegurarte de entregar -como mínimo- los archivos que tenga el *tag* de Entregar en la siguiente sección, asegurando que se mantenga la estructura de directorios original de la actividad. Los demás archivos no es necesario subir, pero tampoco se penalizará si se suben al repositorio personal.

Archivos y entidades

En el directorio de la actividad encontrarás los siguientes archivos:

- Entregar Modificar `main.py`: Archivo que contiene las distintas funciones a completar.
- No modificar `sismos.csv`: Archivo que contiene la información de distintos Sismos que han ocurrido en Chile. Presenta la siguiente información:
  - **Fecha y hora local:** Texto correspondiente a la fecha y hora en la que ocurrió el sismo.
  - **Magnitud Ms:** Magnitud de onda superficial, corresponde a un *float* que se usan para describir la fuerza o el "tamaño" general de un sismo.
  - **Magnitud Mw:** Magnitud de momento, corresponde a un *float* que indica mediante una escala logarítmica la energía total que se libera en un sismo.
  - **Profundidad:** Indica mediante un *int* la profundidad del sismo en kilómetros.
  - **Efecto:** Texto que indica los efectos que tuvo el sismos. Estos pueden ser: T (Tsunami), TM (Tsunami moderado), TD (Tsunami destructor)
  - **Coordenadas:** Texto que indica las coordenadas donde ocurrió el sismo.
  - **Ubicación:** Texto que representa la ubicación geográfica donde ocurrió el sismo. Ordenan la información de forma jerárquica, indicando país, región, provincia, comuna y otras sub-unidades de organización.

Parte I. Cargar información

Antes de poder trabajar la información de los sismos, deberemos cargar su información. Deberás completar la siguiente función:

- Modificar `def cargar_sismos(path_archivo: str) -> pd.DataFrame:`

  Lee el contenido del archivo indicado en `path_archivo` y retorna un *DataFrame* con dicha información.

  Antes de retornar el *DataFrame*, la función se debe transformar:

  - La columna "Ubicación" en una lista de *strings*, separando el texto original según los punto y coma (`;`). Debes asegurar que se mantenga el orden original de las distintas partes del texto.
  - La columna "Fecha y hora local" a un objeto de tipo `datetime`. Para esto, te recomendamos investigar la función `to_datetime` de la librería Pandas y el formato de *strings* que utiliza la librería `datetime` para transformar textos a fechas.

Parte II. Consultas sismos

Ahora que hemos cargado los datos en un *DataFrame* podemos realizar consultas sobre la información que tenemos de los sismos. Deberás completar las siguientes funciones:

- Modificar `def filtrar_sismos_magnitud(df: pd.DataFrame, magnitud: float, operacion: Optional[str] = '>') -> pd.DataFrame:`

  Recibe un *DataFrame* y retorna todos los sismos que su "Magnitud Ms" cumplan la magnitud y operación recibidas como *input*.

  Los valores que puede recibir como operación son:

  - `'>'`: "Magnitud Ms" mayor estricto que magnitud recibida.
  - `'='`: "Magnitud Ms" igual a magnitud recibida.
  - `'<'`: "Magnitud Ms" menor estricto que magnitud recibida.

- Modificar `def estadisticas_efectos(df: pd.DataFrame) -> pd.Series:`

  Recibe un *DataFrame* y retorna un objeto *Series* que resume los distintos de efectos que ha tenido cada sismo (considerando también los `'-'`) y el porcentaje asociado a cada uno de estos.

  Recomendamos investigar los método `groupby` y `count` asociados a los objetos *DataFrame*.

Parte III. Grafo ubicaciones

Ahora que sabes cómo realizar consultas en el *DataFrame*, te propones avanzar a un nuevo desafió y empezar a utilizar la información jerárquica que presenta la "Ubicación" de cada sismos.

Para esto, crearás un grafo que guarde toda las relaciones presentes en la columna "Ubicación" y utilizarás ese gráfico para identificar lugares que hayan presentado sismos en el mar. En base a lo anterior, deberás completar los siguientes métodos:

- Modificar `def crear_grafo_ubicaciones(df: pd.DataFrame) -> dict[set]:`

  Recibe un *DataFrame* y retorna un grafo formado a partir de la información contenida en la columna "Ubicación".

  El grafo retornado debe representar las conexiones mediante un diccionario de adyacencia, donde cada llave se un *string* y cada valor un set de *strings*.

- Modificar `def ubicacion_tuvo_sismo_mar(grafo: dict[set], ubicacion: str) -> bool:`

  Recibe un grafo representado mediante un diccionario de adyacencia, y una ubicación correspondiente a uno de los nodos de dicho grafo. Retorna un booleano que indica si desde dicha ubicación ha presentado sismos en el mar.

Parte IV. Corrupción archivos

Finalmente, has notado que la información descargada de ... a veces llega corrompida, por lo que te planteas utilizar Expresiones Regulares para reparar esta información y que no interfiera con todo el trabajo previo:

- `def limpiar_header_corrompido(header: str) -> str:`

  Recibe un texto corrompido y retorna el texto reparado. Al reparar un texto, se deben eliminar los siguientes elementos adicionales, los cuales pueden aparecer una o más veces en el texto:

  - Dos o más letras mayúsculas seguidas
  - Números
  - Caracteres especiales `!#$%&?*+`

Notas

- **No puedes hacer *import* de otras librerías externas a las entregadas en el archivo.**
- Recuerda que la ubicación de tu entrega es en **tu repositorio de Git**. En la rama (*branch*) por defecto del repositorio: `main`.
- Se recomienda completar la actividad en el orden del enunciado.
- 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.
- Si aparece un error inesperado, ¡léelo y revisa el código del *test*! Intenta interpretarlo y/o buscarlo en Google.

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. Para ejecutar estos *tests*, **primero debes posicionar tu terminal/consola en la carpeta de la actividad (`Actividades/AC7`)**. 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`
  Para ejecutar solo el subconjunto de *tests* de la Parte I.
- `python3 -m unittest -v -b tests_publicos.test_consultas`
  Para ejecutar solo el subconjunto de *tests* de la Parte II.
- `python3 -m unittest -v -b tests_publicos.test_grafo`
  Para ejecutar solo el subconjunto de *tests* de la Parte III.
- `python3 -m unittest -v -b tests_publicos.test_regex`
  Para ejecutar solo el subconjunto de *tests* de la Parte IV.

Importante: recuerda que si `python3` no funciona, probar con el comando específico de tu computador. Este puede ser `py`, `python`, `py3` o `python3.12`.