Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 11111100011-1

11110 de mayo de 11111100011
Actividad Sumativa

Actividad 1000

Paths, I/O y serialización

Introducción

Recuerda, el mensaje se lee de abajo hacia arriba
antes que los mafiosos, puede que esto nos lleve al Santo Video
pintor Antonini Da Ossa. Es tu misión descubrir el mensaje
han encontrado el mensaje que quiso dejar su autor, el famoso
Última Tarea", pero al parecer, unos esbirros del Tini Tamburini
de la actividad. En el DCC se encuentra un cuadro llamado "La
Este mensaje esta escondido para que solo lo lean personas

La Misión

Se han descubierto mensajes escondidos en las obras de artistas italianos. Dentro de estos mensajes, se encontraron tres tareas, que se han transcrito en este enunciado y debes completar. Puedes realizar las tres partes en el orden que desees, pero solo completándolas todas podrás acceder al mensaje secreto.

Recuerda que las 3 partes de esta misión son independientes entre sí. No necesitas terminar una para poder hacer las demás.

Paths (`parte_paths.py`)

Para asegurar que los secretos de la hermandad queden fuera del alcance del malvado Tini Tamburini, decides inspirarte en las Nueve Pizzas del Gran Polea (Divina DCComedia): Pepperoni, PolloBBQ, Queso, Vegetariana, Napolitana, Hawaiiana, MeatLovers, Jaiba y SinQueso.

A ti, como gran conocedor del uso de *paths*, se te encarga crear lo necesario para ocultar del Tini la imagen `la_ultima_tarea.jpg`. Para esto, deberás crear otra serie de funciones con las siguientes funcionalidades:

- `generar_carpetas(lista)`: Utilizando los nombres de las Nueve Pizzas obtenidas a partir de la lista entregada, esta función debe encargarse de crear 9 carpetas[1], con los nombres de cada pizza,

[1] El comando `makedirs()` explicado en la sección Notas podría serte de gran utilidad para crear un directorio

dentro de otra carpeta llamada `Boveda`. Asimismo, cada una de estas carpetas creadas deberán contener otras 5 carpetas, cuyos nombres también vienen de la lista, cuidando de que los nombres sean todos distintos y se elijan de manera aleatoria[2].

Figura 1: Ejemplo de directorios posibles.

- `generar_path()`: Para asegurar que las obras no sean encontradas con facilidad, esta función debe retornar el *string* de una ruta cualquiera (aleatoria) de las que fueron creadas a partir de la función `generar_carpetas(lista)`. Por ejemplo: `Boveda/Pepperoni/Hawaiiana`.
- `esconder_obras(imagen, ruta)`: Esta función debe ser capaz de generar una copia de la imagen entregada[3], dado su nombre y extensión, a la ruta correspondiente. La ruta recibida es de la forma retornada por la función `generar_path()` del punto anterior. Además, solo puede existir una única copia de la imagen dentro de la carpeta `Boveda`, por lo cual debe eliminar cualquier otra copia que se haya encontrado creada previamente[4].

Serialización (`parte_serializacion.py`)

La segunda parte de la misión, es conseguir el mensaje dentro de las descripciones de las obras. Para lograr recuperar el mensaje, te piden crear las siguientes funciones:

- `cargar_obras(ruta_obras)`: El archivo `operas.json` contiene toda la información de las obras. Las obras contienen algunos datos que no son relevantes para la solución de este misterio. Por esto, se te entrega el archivo `caratteristicas.json`[5] con las características que sí deberás obtener de las obras, mediante la función `obras_hook`. Esta función deberá retornar una lista de objetos de la clase `Obra`. e
- `generar_mensaje(lista_obras)`: Tu función debe serializar cada una de las obras utilizando **pickle**. Durante este proceso deberás agregar, a cada obra, el atributo *messaggio* con un mensaje

[2] ***Hint***: Los métodos `sample()` y `choice()` de la librería `random` te pueden ser de utilidad en esta sección.
[3] ***Hint***: Copiar es equivalente a escribir los datos nuevamente, en otra ruta
[4] El método `remove` de `os` puede ser de ayuda.
[5] Los nombres estan en *Italiano*. *Opera* = Obra. *Caratteristica* = Características.

generado por 3 palabras al azar del atributo *descrizione*. Los archivos generados deben ser guardados con el formato de nombre `<nome>-<autore>.opera`, donde `nome` y `autore` son el nombre de la obra y el nombre de su respectivo autor, en la carpeta `Obras`.

Mensaje secreto (+2.5 décimas de *bonus*): Para obtener el mensaje, debes sacar el 13.er caracter[6] de cada descripción de obra donde el autor sea Antonini Da Ossa, tras ordenarlas de forma cronológica. Para esto, debes completar la función `mensaje_bonus(lista_obras)`.

I/O (`parte_io.py`)

Haciendo gala de la cautela por la cual son conocidos los alumnos de Programación Avanzada, decides crear un mecanismo de defensa en caso de que el malvado Tini Tamburini esté cerca de encontrar el mensaje oculto. Para esto, debes modificar el archivo `la_gioconcha.jpg`, haciéndolo ilegible para un mortal común y corriente. Para lograrlo, debes implementar:

- `cargar_imagen(ruta)`: Esta función recibe la ruta de la imagen y debe encargarse de abrir el archivo y retornarlo en forma de *bytes*, para luego poder esconder la imagen usando la siguiente función.
- `esconder_imagen(imagen)`: Será la encargada de esconder la imagen utilizando el algoritmo que se detallará a continuación. Debe recibir los *bytes* obtenidos al utilizar la función `cargar_imagen(ruta)` y retornar los *bytes* modificados.
  - Primero, debe invertir el orden en que está el arreglo que recibiste (y que representa a la imagen). Por ejemplo: `[1,2,3,4] ->[4,3,2,1]`
  - Después de la transformación anterior, cada 2 elementos del arreglo de *bytes*, se debe invertir su orden. En caso que el arreglo tenga una cantidad impar de *bytes*, no mover el último elemento. Por ejemplo: `[1,2,3,4,5] ->[2,1,4,3,5]`.
- `guardar_imagen(imagen)`: Esta función debe recibir la imagen codificada del paso anterior y guardarla, en formato `<nombre>.uwu`, donde nombre debe ser un número al azar entre 1 y 100 en su representación binaria[7]

Mensaje secreto (+2.5 décimas de *bonus*): Deberás escribir la función `arreglar_imagen(ruta_imagen)`, la cual debe revertir el proceso completo de `esconder_imagen(imagen)` devolviendo la imagen sana y salva. Debes aplicar esta función al archivo `supersecreto.gato` que te ha sido entregado, y guardar el *output* en formato `.jpg`. En este archivo se encuentra tu segunda pista.

Notas

- En los archivos `parte_paths.py`, `parte_serializacion.py` y `parte_io.py` se encuentran los esqueletos de las funciones y clases de la actividad, junto al código para ejecutarlos de forma correcta, por lo que sólo debes completar cada función para finalizar la actividad.
- Las imágenes a utilizar son `la_ultima_tarea.jpg` y `la_gioconcha.jpg`.
- Cada sección es independiente de la otra. Es decir, puedes completarlas en el orden que estimes conveniente.

[6] Siendo el primer caracter, el caracter 0

[7] El método `bin` convierte un `int` a su representación binaria.

- Para las secciones Serialización (`parte_serializacion.py`) y *Paths* (`parte_paths.py`), tengan ojo con el manejo de errores que podrían surgir si es que los archivos y/o directorios ya existen y viceversa. En particular, el método `makedirs()` de la librería `os` posee un parámetro llamado `exist_ok` que te facilita el manejo de estos tipos de casos.
- Está estrictamente **prohibido** utilizar la librería `shutil`.
- Una vez obtengas los dos **mensajes secretos** de la parte Serialización (`parte_serializacion.py`) e I/O (`parte_io.py`), debes unir ambos mensajes mediante un "/" para llegar al santo video.

Requerimientos

- (2.00 pts) *Paths*
  - (1 pts) La función `generar_carpetas(lista)` logra crear las carpetas correctamente.
  - (0.5 pts) La función `generar_path()` retorna una ruta que cumple con lo requisitos pedidos.
  - (0.5 pts) La función `esconder_obras(imagen, ruta)` logra generar una copia del archivo original a la ruta correspondiente.
- (2.00 pts) Serialización
  - (1 pts) La función `cargar_obras(ruta_obras)` es implementada de manera correcta (Método de deserialización pedido).
  - (1 pts) La función `generar_mensaje(lista_obras)` es implementada de la forma correcta (Método de serialización pedido).
- (2.00 pts) I/O
  - (0.6 pts) La función `cargar_imagen(ruta)` devuelve correctamente los datos del archivo abierto
  - (0.8 pts) La función `esconder_imagen(imagen)` aplica correctamente el algoritmo pedido.
  - (0.6 pts) La función `guardar_imagen(imagen)` guarda correctamente el archivo.
- (+0.5 pts) *Bonus*
  - (+0.25 pts) Implementa la función `mensaje_bonus(lista_obras)` de forma correcta.
  - (+0.25 pts) Implementa la función `arreglar_imagen(ruta_imagen)` de forma correcta.
  - (+ **entretención personal**) Une los mensajes obtenidos de los dos mensajes anteriores.

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC08`
- **Hora del *push*:** 16:30