Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 0x7E3-0b10

0xE de noviembre de 0x7E3
Actividad Formativa

Actividad 0b1001

I/O + Serialización

Introducción

Atención alumno de Programación Avanzada. Este es un llamado de parte de suma importancia de parte de **PROGRA**.

Somos una organización que se dedica a eliminar una amenaza conocida como Tareos, seres malvados que históricamente han sido nuestro enemigo número uno, y está vez han vuelto y lograron destrozar nuestros computadores centrales. Es por esto por lo que necesitamos de tu ayuda, ya que con tus habilidades de programador podremos repararlos y así enfrentarnos a estos seres mediante nuestras unidades Docengelion y derrotarlos para siempre.

Nuestras tres computadoras han sido infectadas, y se les ha implantado un **mensaje oculto** que necesitamos decifrar. Para lograr esto tendrás que reparar los códigos de las computadoras Daniar, Enzohor y Pintosar, que poseen tres problemas diferentes de serialización. Si logramos conseguirlo, lograremos acabar con los ataques de los Tareos y estaremos agradecidos de por vida por salvarnos.

Estas tres computadoras trabajan de manera independiente, por lo que puedes repararlas en cualquier orden y no dependen una de la otra. Si completas las tres partes, puedes describir el secreto al final.

Daniar (Serialización JSON)

La computadora **Daniar** se dedica a controlar el estado de las Unidades Docengelion.

Dentro del archivo `daniar.py` encontrará la clase llamada `Docengelion` que contiene ciertos atributos importantes y las siguientes funciones:

- `def recibir_eva(ruta)`: esta función deberá cargar el archivo cuyo *path* es `ruta`, que contiene la información de las Unidades Docengelion, serializada como un archivo JSON. Deberá retornar una lista de instancias de `Docengelion` cuyos modelos correspondan a los del archivo.
- `def reparar_eva(docengelion)`: esta función recibe una instancia de `Docengelion` que debe ser reparada. Para eso, debes serializar en JSON la instancia recibida y realizar ciertos cambios **sin editar la instancia original** (es decir, debes usar un `JSONEncoder`), de la siguiente manera:
  - cambiar el atributo `estado` por el valor `'reparacion'`,
  - y agregar el atributo `registro_reparacion` con la fecha y hora guardado en un *string*.

  Dentro de la misma función, el objeto serializado será guardado en el archivo `Unidad-{modelo}.json` donde `modelo` es el atributo correspondiente de la instancia, este archivo debe quedar en una carpeta `Daniar`.

Enzohor (Serialización `pickle`)

La computadora **Enzohor** se dedica a controlar el estado de los pilotos al estar dentro de su Unidad Docengelion.

Dentro del archivo `enzohor.py` encontrará la clase llamada `Piloto` que contiene ciertos atributos importantes y las siguientes funciones:

- `def cargar_almas(ruta)`: esta función deberá cargar el archivo cuyo *path* es `ruta`, el cual contiene la información de los pilotos. El archivo a cargar fue serializado mediante `pickle`, entonces para que esta información sea cargada correctamente se debe definir el método `def __setstate__` dentro de la clase `Piloto`, **donde se hará uso de la segunda función** de **Enzohor**. Finalmente, la función debe retornar una lista de instancias de `Piloto`.
- `def aumentar_sincronizacion(estado):` esta función recibe el estado (`dict`) que será deserializado en la función anterior (es decir, será utilizada en el `__setstate__`). Esta función elimina todo *substring* presente en el atributo `alma`, que cumpla con estas 3 condiciones **simultáneamente**:
  - el *substring* comienza con una `E` (mayúscula), y solo contiene esa `E`
  - el *substring* termina con una `O` (mayúscula), y solo contiene esa `O`
  - el *substring* contiene al menos una `G` (mayúscula)

  Finalmente, la función debe retornar un nuevo **estado** con los cambios realizados.

  Por ejemplo:

  String original: `"Este stringEdsGGgofoOf es negro"`

  Identificamos el *substring*: `"Este stringEdsGGgofoOf es negro"`

  String resultante: `"Este stringf es negro"`

Bonus: RegEx

Enzohor posee una arquitectura de computador primigenia, por lo que su capacidad de procesamiento es baja y debe estar dedicada sólo a controlar el estado de los pilotos. Por lo que el uso de ***RegEx*** para `def aumentar_sincronizacion` será recompensado (moralmente).

Pintosar (Manejo de *bytes*)

La computadora **Pintosar** se dedica a controlar el estado de la comunicación entre las Unidades Docengelion y los pilotos.

La comunicación se ha corrompido debido a una falla en la sincronización de los pilotos quienes están teniendo visiones, corrompiendo todo registro visual de los combates.

Para recuperarlo, deberá completar la siguiente función dentro del archivo `pintosar.py`:

- `def reparar_comunicacion(ruta)`: Esta función recibe la ruta del archivo visual a reparar, para eso, deberá obtener los *bytes* del archivo y realizar el siguiente algoritmo:

  1. Deberá obtener *chunks* de 16 *bytes*.
  2. Utilizar el primer *byte* del *chunk* como pivote.
  3. Eliminar todos aquellos *bytes* del *chunk* cuyo valor numérico sea **mayor o igual** al pivote.
  4. Eliminar el pivote.

  Terminado esto, deberás escribir los *bytes* restantes del *chunk* en el archivo `Docengelion.bmp`.

Parte Final: Plan de Instrumentalización Programada (sin nota)

El secreto tras este plan se encuentra escondida en las tres partes de su misión:

Primero, repare el archivo `EVA.xdc` llamando a la función `reparar_comunicacion(ruta)` dentro de la computadora **Pintosar**.

Luego, con la información obtenida, obtenga el atributo `nucleo` del Docengelion correspondiente dentro de la computadora **Daniar**.

Finalmente obtenga el atributo `alma` del piloto Shinji Gonzalez dentro de la computadora **Enzohor**.

Al concatenar estos elementos, obtendrá el secreto.

Notas

- La librería `time` posee la función `strftime()`. Puedes encontrar ejemplo de su uso en este **enlace**.
- Para crear una carpeta dinámicamente en Python, recuerda que `os` tiene un método `makedirs`.
- Para probar su expresión regular (Regex), puede utilizar la herramienta online https://pythex.org/

Requerimientos

- (2.00 pts) Daniar (Serialización JSON)
  - (0.50 pts) Completa la función `recibir_eva` deserializando correctamente.
  - (0.50 pts) Completa la función `reparar_eva` serializando correctamente.
  - (1.00 pts) Completa la clase `DocengelionEncoder` correctamente.
- (2.00 pts) Enzohor (Serialización `pickle`)
  - (0.50 pts) Completa la función `cargar_almas` deserializando correctamente
  - (0.50 pts) Completa el `__setstate__` de la clase `Piloto` correctamente.
  - (1.00 pts) Completa la función `aumentar_sincronizacion` correctamente.
  - *Bonus* (0.50 pts) Completa la función `aumentar_sincronizacion` mediante sólo el uso de Regex.
- (2.00 pts) Pintosar (Manejo de *bytes*)
  - Completa la función `reparar_comunicacion` correctamente.

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC09/`
- **Hora del *push*:** 16:40

Auto-evaluación

Como esta corresponde a una actividad formativa, te extendemos la instancia de responder, después de terminada la actividad, una auto-evaluación de tu desempeño. Esta se habilitará a las **16:50 de jueves 0xE de noviembre** y tendrás plazo para responderla hasta las **23:59 del día siguiente**. Puedes acceder al formulario mediante el siguiente enlace:

`https://forms.gle/evT9FycXieceKw7N9`

El asistir, realizar la actividad y responder la auto-evaluación otorgará como bonificación al alumno 2 décimas para sumar en su mejor actividad sumativa del semestre.