

Generar un programa que sea capaz de restaurar el estado de ejecución

Alumno: Oswaldo Gael Cervantes Castoño

Código: 219747468

Computación Tolerante a Fallas

Sección D06

Profesor: Michel Emanuel López Franco



Objetivo: Generar un programa que sea capaz de restaurar el estado de ejecución.

Desarrollo:

Para el desarrollo de esta actividad decidí hacer uso del módulo “pickle” en Python, el cual permite serializar el estado de un objeto en formato binario, de modo que podemos recuperar la información exacta del objeto modificado en tiempo de ejecución para reconstruirlo en la misma ejecución o incluso en una nueva sin el riesgo de perder la información que se genera en programas que manejan cantidades considerables de datos los cuales queremos mantener en la mayor cantidad posible en situaciones críticas como en el caso de cierres inesperados.

Ejemplo codificado

Para la codificación de esta actividad utilicé una agenda telefónica que hice hace algunos semestres, la cual guarda información de contacto de alumnos de la Universidad de Guadalajara para posteriormente visualizarlos en pantalla.

La clase utilizada para generar objetos en tiempo de ejecución es la siguiente:

```
class Entrada:
    def __init__(self, nombre, apellido, codigo, telefono, carrera):
        self.__nombre = nombre
        self.__apellido = apellido
        self.__codigo = codigo
        self.__telefono = telefono
        self.__carrera = carrera

    def get_nombre(self): return self.__nombre
    def get_apellido(self): return self.__apellido
    def get_codigo(self): return self.__codigo
    def get_telefono(self): return self.__telefono
    def get_carrera(self): return self.__carrera
```

En un segundo archivo construí la clase “Agenda”. En dicho archivo se importa la clase “Entrada” así como el módulo “pickle”. En el constructor de la clase “Agenda” se busca en el directorio actual un archivo con información de algún checkpoint guardado previamente, si se encuentra entonces por medio del método .load() del módulo “pickle” se obtiene la lista de contactos del archivo binario y se le asigna el contenido a la lista de

la agenda para poder seguir trabajando sobre ella, en caso de no encontrarse el archivo con el backup se produce una excepción y en el bloque de “except” se inicializa la lista de contactos como vacía para poder insertar información nueva.

```
class Agenda:
    def __init__(self):
        try:
            with open("user_data", "rb") as agenda:
                self.__entradas = pickle.load(agenda)
            agenda.close()
        except:
            self.__entradas = []
```

La clase también cuenta con un método para realizar “checkpoints” en determinados momentos de la ejecución del programa, dicha función se define de la siguiente manera:

```
def __do_checkpoint(self):
    #Serializar el estado del objeto (lista de contactos)
    if len(self.__entradas) > 0:
        with open("user_data", "wb") as agenda:
            pickle.dump(self.__entradas, agenda)
        agenda.close()
```

Si la longitud de la lista es mayor que 0 (contiene al menos un elemento) entonces se crea un archivo binario y por medio del método .dump() del módulo “pickle” se convierte el estado actual de la lista de contactos en una serie de bytes que se guardan en dicho archivo. Finalmente se cierra el archivo creado.

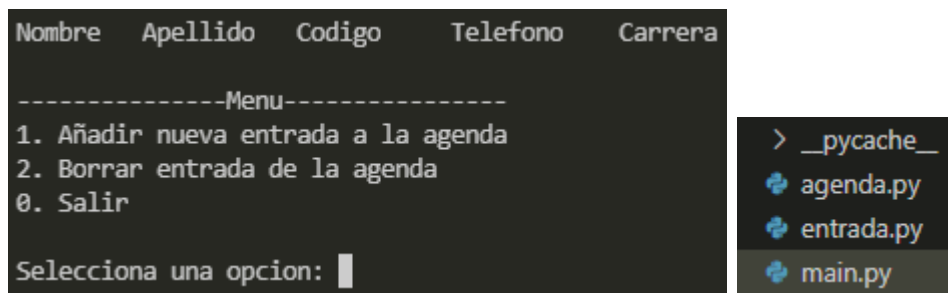
La clase cuenta con métodos para crear y borrar una entrada (contacto), en ambos casos, al final de cada método se invoca el método “__do_checkpoint()” para serializar el estado actual del objeto.

```
def agregar_entrada(self, nombre, apellido, codigo, telefono, carrera):
    entrada = Entrada(nombre, apellido, codigo, telefono, carrera)
    self.__entradas.append(entrada)
    self.__do_checkpoint()
```

```
def eliminar_entrada(self, codigo):
    if len(self.__entradas) == 0:
        print("\nNo hay elementos en la agenda...")
    else:
        index = 0
        encontrado = False
        for entrada in self.__entradas:
            if entrada.get_codigo() == codigo:
                index = self.__entradas.index(entrada)
                self.__entradas.pop(index)
                print("\nRegistro borrado de la agenda...")
                encontrado = True
                break
        if not encontrado:
            print("\nNo se encontraron coincidencias...")
        else:
            self.__do_checkpoint()
```

Pruebas del código

La primera prueba consiste en inicializar la agenda y añadirle datos para posteriormente cerrarla y volver a recuperar la información justo al iniciar de nuevo la ejecución de la aplicación:



```
Nombre  Apellido  Codigo  Telefono  Carrera

-----Menu-----
1. Añadir nueva entrada a la agenda
2. Borrar entrada de la agenda
0. Salir

Selecciona una opcion: 
```

> __pycache__
agenda.py
entrada.py
main.py

Podemos observar que la información de la agenda se muestra en todo momento pero al ser una agenda nueva simplemente se muestran los encabezados de la tabla y en la imagen de la derecha podemos observar que aún no se crea el archivo de respaldo de la información del usuario.

Procedemos a insertar algunos registros por medio de la consola para poder visualizar algo de contenido en la parte superior y al mismo observar que al momento de crear registros se genera el archivo de respaldo y este queda listo para cuando se requiera hacer un vaciado de su información en la lista de contactos.

```
Nombre  Apellido  Codigo  Telefono  Carrera
Gael    Cervantes  219747468  3317972728  INCO
Aide    Flores    218767566  3311334876  INCO
Kevin   Fernandez  234354677  3321343245  INNI
Miguel   Madrigal  213245678  3311234532  INDU

-----Menu-----
1. Añadir nueva entrada a la agenda
2. Borrar entrada de la agenda
0. Salir

Selecciona una opcion: 
```

```
> __pycache__
+ agenda.py
+ entrada.py
+ main.py
+ user_data
```

Podemos observar que se ha generado un archivo con el checkpoint, ahora procedemos a terminar la ejecución del programa:

```
Nombre  Apellido  Codigo  Telefono  Carrera
Gael    Cervantes  219747468  3317972728  INCO
Aide    Flores    218767566  3311334876  INCO
Kevin   Fernandez  234354677  3321343245  INNI
Miguel   Madrigal  213245678  3311234532  INDU

-----Menu-----
1. Añadir nueva entrada a la agenda
2. Borrar entrada de la agenda
0. Salir

Selecciona una opcion: Traceback (most recent call last):
  File "C:\Users\Gael\Desktop\Restaurar estado de ejecucion\main.py", line 15, in <module>
    opt = input("\nSelecciona una opcion: ")
KeyboardInterrupt
^C
C:\Users\Gael\Desktop\Restaurar estado de ejecucion>
```

Ejecutamos de nuevo el programa y podemos observar que la información se restaura:

```
C:\Users\Gael\Desktop\Restaurar estado de ejecucion>python main.py
Nombre  Apellido  Codigo  Telefono  Carrera
Gael    Cervantes  219747468  3317972728  INCO
Aide    Flores    218767566  3311334876  INCO
Kevin   Fernandez  234354677  3321343245  INNI
Miguel   Madrigal  213245678  3311234532  INDU

-----Menu-----
1. Añadir nueva entrada a la agenda
2. Borrar entrada de la agenda
0. Salir

Selecciona una opcion: 
```

Link al repositorio con el código fuente: <https://github.com/gaelcervantes65/Generar-un-programa-que-sea-capaz-de-restaurar-el-estado-de-ejecuci-n.git>

Conclusiones

Como conclusiones para la realización de esta actividad puedo mencionar que pude aprender mucho con la adaptación del código sobre el que trabajé ya que en primera instancia dicha actividad la había programado para que la serialización se hiciera sobre archivos de texto guardando cada campo de un objeto con separadores o delimitadores de campo, lo cual si bien es sencillo, requiere de algunas líneas de código y en lo personal considero que no es un método tan “limpio” u ordenado ya que depende mucho del criterio del programador; en cambio, por medio del uso del módulo “pickle” pude ahorrar muchas líneas de código y encontré un método estándar que funciona de forma muy práctica y eficaz que serializa el objeto como una serie de bytes que se pueden recuperar y asignar de nueva cuenta a un objeto de la misma clase.

Fuentes de información

- How to checkpoint a long-running function pythonically? (2015, 8 diciembre). Stack Overflow. Recuperado 20 de febrero de 2022, de <https://stackoverflow.com/questions/34155841/how-to-checkpoint-a-long-running-function-pythonically>
- pickle — Python object serialization — Python 3.10.2 documentation. (s. f.). docs.python.org. Recuperado 20 de febrero de 2022, de <https://docs.python.org/3/library/pickle.html>