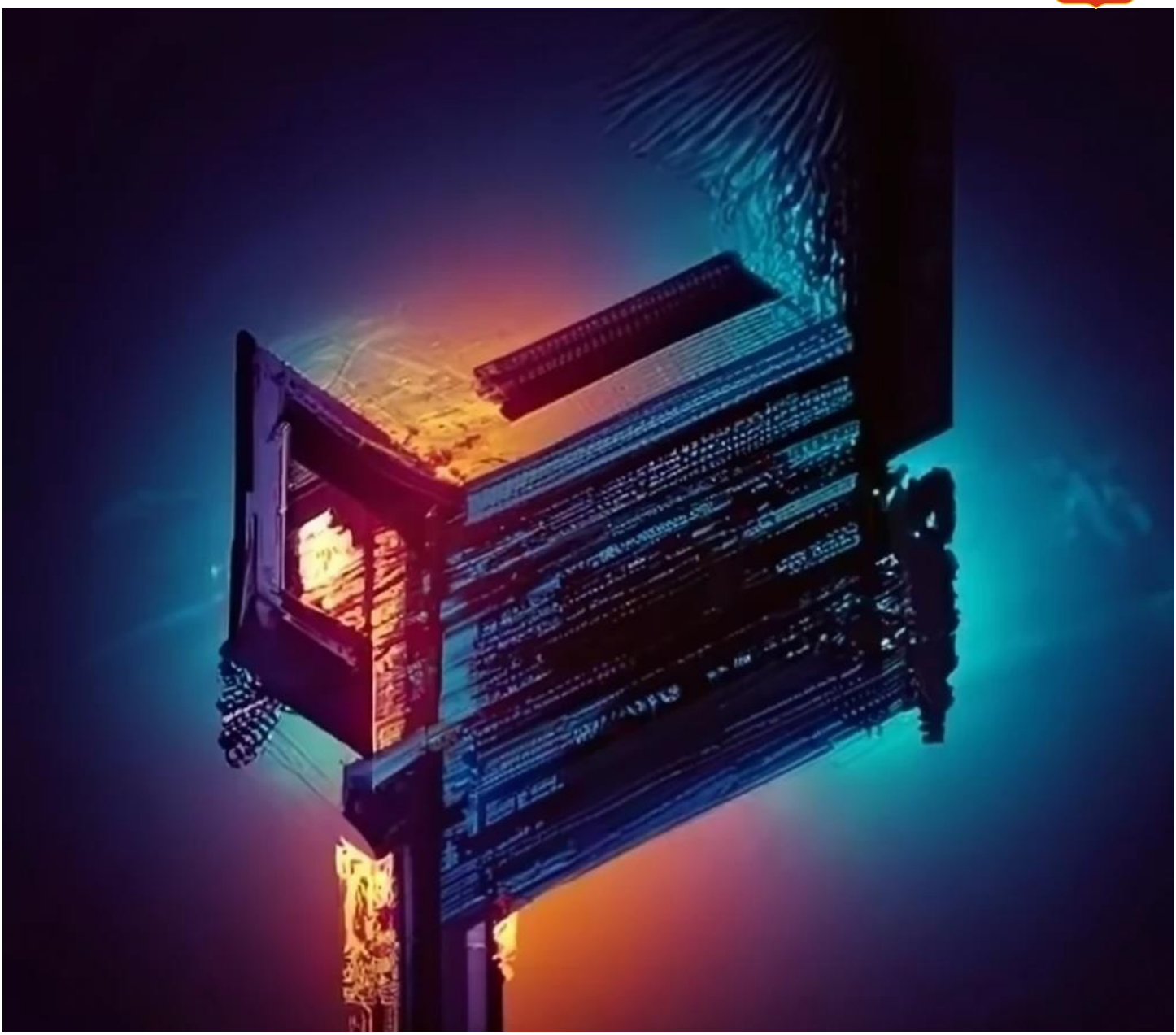
Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías



Sección: D06 Profesor: MICHEL EMANUEL LOPEZ FRANCO
Tema: Generar un programa que sea capaz de restaurar el estado de ejecución.
Ciclo: 2024A

Introducción

Esta es la ejecución de lo que se conoce como checkpointing dando a conocer ejemplos de lo que existe en el entorno de programación dentro de los lenguajes y se verá a continuación la utilización de uno de ellos como lo es el pickle en Python.

El módulo pickle implementa protocolos binarios para serializar y deserializar una estructura de objetos de Python. "Decapado" es el proceso mediante el cual una jerarquía de objetos de Python se convierte en un flujo de bytes, y "deseleccionado" es la operación inversa, mediante la cual un flujo de bytes (de un archivo binario o de un objeto similar a bytes) se convierte nuevamente en una jerarquía de objetos.

Contenido

Se muestra a continuación la utilización de lo que es pickle para la demostración de programas con checkpointing, mostrando un código sencillo y su salida usado en el lenguaje de python.

```python
import pickle

class ProgramState:
    def __init__(self):
        self.variable1 = 0
        self.variable2 = "Hola"

    def display_state(self):
        print(f"Variable1: {self.variable1}, Variable2: {self.variable2}")

def save_checkpoint(state, filename="checkpoint.pkl"):
    with open(filename, "wb") as file:
        pickle.dump(state, file)

def load_checkpoint(filename="checkpoint.pkl"):
    with open(filename, "rb") as file:
        return pickle.load(file)

# Ejemplo de uso
if __name__ == "__main__":
    # Crear una instancia del estado del programa
    current_state = ProgramState()
    current_state.display_state()

    # Guardar un checkpoint
    save_checkpoint(current_state, "initial_checkpoint.pkl")
```

```python
    # Modificar el estado del programa
    current_state.variable1 = 42
    current_state.variable2 = "Mundo"
    current_state.display_state()

    # Guardar otro checkpoint después de la modificación
    save_checkpoint(current_state, "modified_checkpoint.pkl")

    # Restaurar desde el primer checkpoint
    restored_state = load_checkpoint("initial_checkpoint.pkl")
    print("\nRestaurado desde el primer checkpoint:")
    restored_state.display_state()

    # Restaurar desde el segundo checkpoint
    restored_state = load_checkpoint("modified_checkpoint.pkl")
    print("\nRestaurado desde el segundo checkpoint:")
    restored_state.display_state()
```

con el muestrario de lo que es la salida de este ejemplo sencillo

```
exico/Tolerante a fallas/CheckPoints.py"
Variable1: 0, Variable2: Hola
Variable1: 42, Variable2: Mundo

Restaurado desde el primer checkpoint:
Variable1: 0, Variable2: Hola

Restaurado desde el segundo checkpoint:
Variable1: 42, Variable2: Mundo
PS C:\Users\USER\Downloads\Micro Analizador Lexico>
```

Viendo desde más perspectivas dentro de la informática y la computación también tenemos otros puntos para poder hacer una restauración de versiones pero estos implican diferentes cosas por lo cual esta es solo una demostración de cómo puede ser en programación.

Sistema de Control de Versiones: Puedes utilizar sistemas de control de versiones como Git para gestionar versiones de tu código y revertir a versiones anteriores si es necesario. Esto no solo te permite revertir el código, sino también cualquier otro archivo o recurso en tu proyecto.

Snapshots en Aplicaciones de Virtualización: En entornos de virtualización, como máquinas virtuales o contenedores, puedes tomar snapshots periódicos del estado de la máquina virtual o del contenedor. Esto te permite revertir a un estado anterior si algo sale mal o si necesitas regresar a una configuración previa.

Sistemas de Bases de Datos: Muchas bases de datos ofrecen funcionalidades para realizar copias de seguridad y restauración del estado de la base de datos en diferentes puntos en el tiempo. Por ejemplo, puedes hacer copias de seguridad incrementales o completas y restaurar la base de datos a un estado anterior si es necesario.

Historial de Versiones en Aplicaciones: En aplicaciones que trabajan con datos, como editores de texto, procesadores de documentos, o incluso juegos, puedes implementar un historial de versiones que registre los cambios realizados y permita al usuario revertir a versiones anteriores.

Sistemas de Control de Transacciones: En aplicaciones que realizan operaciones críticas o transaccionales, puedes implementar sistemas de control de transacciones que permitan deshacer operaciones en caso de fallos o errores.

Referencias

- *pickle — Python object serialization*. (s/f). Python Documentation. Recuperado el 6 de febrero de 2024, de (https://docs.python.org/3/library/pickle.html)