Universidad de San Carlos de Guatemala

Facultad de ingeniería

Escuela de ciencias y sistemas

ING. Byron Rodolfo Zepeda Arévalo

AUX. Erwin Fernando Vásquez

Manual Técnico – IPC2

Nombre: Frander Oveldo Carreto Gómez

Carné: 201901371

Fecha de entrega: 06/23

Introducción

El presente Manual Técnico tiene como objetivo proporcionar a los desarrolladores y usuarios avanzados una comprensión detallada de la implementación y funcionalidades de la aplicación de procesamiento de datos de señales y generación de gráficas.

La aplicación, desarrollada en Python, ofrece una solución integral para cargar, procesar y visualizar datos contenidos en archivos XML. Los datos representan señales con información de tiempo (t), amplitud (A) y valor. La herramienta permite generar gráficas interactivas basadas en estos datos, facilitando la comprensión y el análisis de las señales. Además, el programa brinda la capacidad de escribir archivos de salida XML con los datos procesados y ofrece funciones adicionales, como la visualización de información del estudiante.

Requisitos de Software

- Python 3.x
- Biblioteca ElementTree (incluida en la biblioteca estándar de Python)
- Biblioteca Graphviz (debe instalarse previamente, consulte la sección de instalación de Dependencias)

Requisitos de Hardware

- Equipo con suficiente capacidad de procesamiento para ejecutar Python y las operaciones de procesamiento de datos.
- Espacio en disco suficiente para almacenar archivos XML y gráficos generados.

Instalación de Dependencias

- La biblioteca Graphviz debe instalarse antes de utilizar la aplicación. Puede instalarla utilizando pip

Arquitectura y Diseño

Estructura del Código

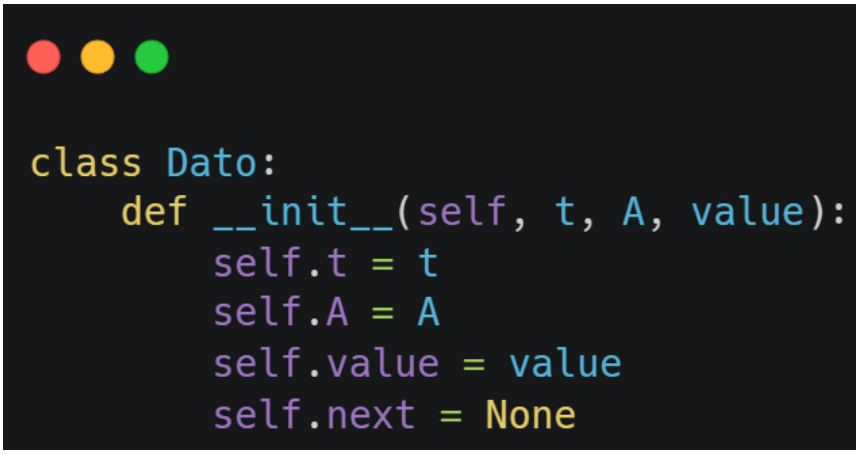
La aplicación está estructurada en clases, incluyendo las clases Dato y Senal para representar los datos de las señales, y funciones que gestionan la carga, procesamiento y visualización de los datos. El programa utiliza las bibliotecas ElementTree para el análisis XML y Graphviz para la generación de gráficas.

Clases y Funciones Principales

- Dato: Clase que representa un dato de señal con tiempo (t), amplitud (A) y valor.

- Senal: Clase que representa una señal con nombre, una lista enlazada de datos y una tabla para almacenar los datos.
- cargar_datos_desde_xml(xml_string): Función que carga datos desde un archivo XML y devuelve una lista de señales.
- mostrar_datos_y_tablas(senales): Función que muestra datos de señales y tablas.
- main(): Función principal que maneja el menú y las opciones de usuario.

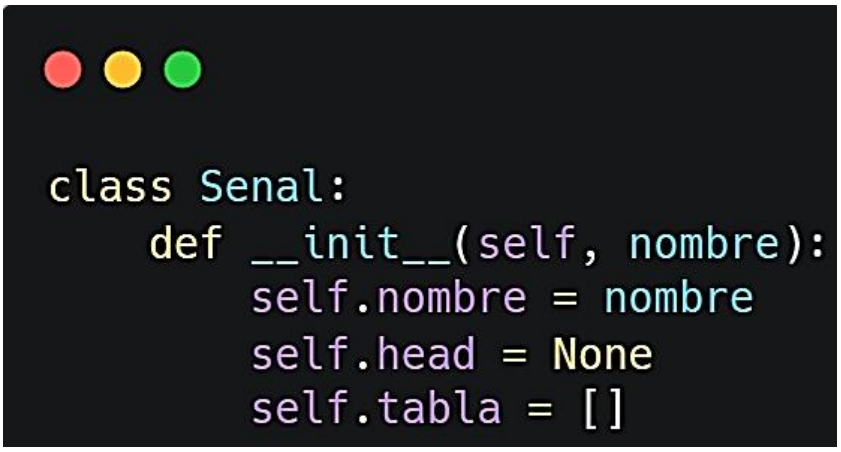
Clase Dato



```
class Dato:
    def __init__(self, t, A, value):
        self.t = t
        self.A = A
        self.value = value
        self.next = None
```

- La clase Dato tiene una estructura sencilla pero fundamental para representar los datos de señales. Cada objeto de esta clase tiene tres atributos: t para el tiempo, A para la amplitud y value para el valor del dato. El atributo next se utiliza para enlazar los datos en una lista enlazada.

Clase Senal



```
class Senal:
    def __init__(self, nombre):
        self.nombre = nombre
        self.head = None
        self.tabla = []
```

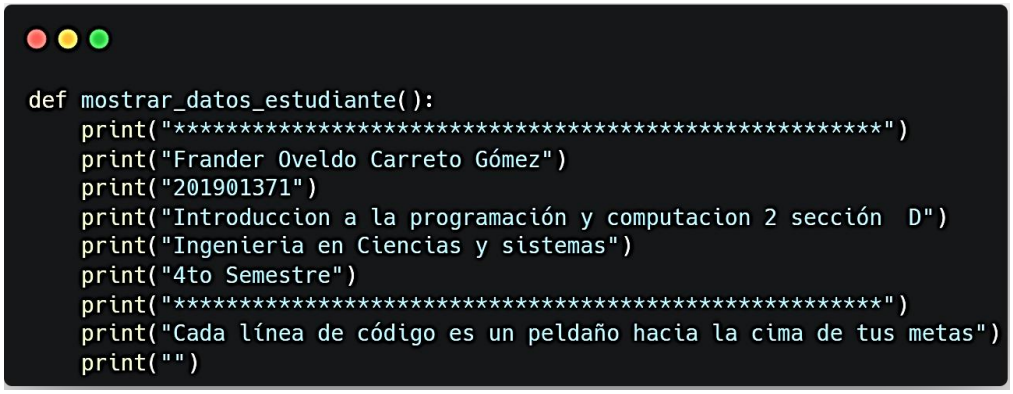
- La clase Senal representa una señal individual. Tiene un atributo nombre para almacenar el nombre de la señal, un atributo head que apunta al

primer dato de la señal en la lista enlazada, y un atributo tabla que es una estructura de tabla utilizada para organizar los datos de la señal.

Uso de la Biblioteca ElementTree y Graphviz

- **Biblioteca ElementTree**
  La biblioteca xml.etree.ElementTree se utiliza para analizar archivos XML. Permite a la aplicación navegar por la estructura jerárquica del XML y extraer la información necesaria, como los datos de señales y sus atributos. Esta biblioteca es fundamental para la funcionalidad de carga de datos desde archivos XML.

- **Biblioteca Graphviz**
  La biblioteca graphviz se utiliza para generar gráficas interactivas basadas en los datos de señales. Esta biblioteca permite crear visualizaciones claras y efectivas que facilitan la comprensión de los datos. Se utiliza para mostrar la estructura de los datos de señales de manera gráfica, lo que resulta útil en la generación de gráficas para su visualización.



```
def mostrar_datos_estudiante():
    print("*****************************************************")
    print("Frander Oveldo Carreto Gómez")
    print("201901371")
    print("Introduccion a la programación y computacion 2 sección  D")
    print("Ingenieria en Ciencias y sistemas")
    print("4to Semestre")
    print("*****************************************************")
    print("Cada línea de código es un peldaño hacia la cima de tus metas")
    print("")
```

Conclusión

- Facilidad de Uso: La aplicación ha sido diseñada pensando en la facilidad de uso tanto para desarrolladores como para usuarios finales. La interfaz de usuario basada en un menú simplifica la navegación y el acceso a las funciones principales, lo que permite a los usuarios cargar, procesar y visualizar datos de señales sin dificultad.

- Eficiencia en el Procesamiento: La estructura de clases y funciones implementadas en el código proporciona una solución eficiente para la carga y procesamiento de datos de señales. La utilización de listas enlazadas y tablas facilita la organización y manipulación de datos, lo que se traduce en un procesamiento rápido y preciso.

- Generación de Gráficas Interactivas: La capacidad de generar gráficas interactivas a partir de los datos de señales ofrece a los usuarios una herramienta poderosa para visualizar y analizar la información. Esto facilita la identificación de patrones y tendencias en los datos, lo que puede ser de gran utilidad en diversas aplicaciones.

- Escritura de Archivos de Salida: La aplicación permite a los usuarios generar archivos de salida XML con los datos procesados, lo que brinda flexibilidad y compatibilidad con otros sistemas o herramientas de análisis. Esto facilita la exportación de resultados para su posterior uso.