PROYECTO 1

202110511 – Fredy Alexander Esteban Pineda

Resumen

En este proyecto como solución a una problemática de agrupación de señales de audio representadas en archivos de entrada, necesitamos hacer uso de Soluciones orientadas en POO. Como ya sabemos estamos en la era de la tecnología y se busca que las computadoras y programas puedan resolver la mayor cantidad de tareas en el menor tiempo posible, ya que este tipo de programas pueden ser de mucha utilidad para compañías que procesan una gran cantidad de información y desean manejarla de la manera más precisa posible. Por lo que por medio de métodos como Listas buscamos reducir estas señales de audio a su máxima expresión por medio de un sistema de matrices en los cuales se nos serán representadas las señales, implementando el uso de lecturas de archivos de Python que es el lenguaje en el cual nos estaremos desarrollando.

Palabras clave

Python

POO

parámetro

variable

Lista

Abstract

Traducir al idioma inglés, el resumen redactado en la columna de la izquierda.

La traducción debe ser revisada con un profesional en ingeniería con amplios conocimientos del idioma inglés, en caso que en forma personal no se posean.

Evitar la utilización del traductor de google u otra similar.

El abstract y las keywords deben abarcar solamente esta columna.

Keywords

python

POO

parameter

variable

List

Introducción

Para adentrarnos en este tema es importante conocer el lenguaje en el cual nos vamos a desarrollar que en este caso es Python. Este es un lenguaje de alto nivel, flexible y muy poderoso en el cual podemos manejar archivos que es en lo que nos enfocaremos en este proyecto. Para el manejo de datos de dichos archivos necesitaremos usar Listas, Python ya cuenta con estructuras de listas integradas, sin embargo nosotros tendremos que crear la nuestra con su propia estructura. Por medio de sentencias tendremos que verificar la información de los archivos que manejaremos para poder dar un resultado mucho más compacto y sin errores que puedan llegar a la mala interpretación de los datos.

Desarrollo del tema

Python

Como ya lo habíamos mencionado anteriormente python es un lenguaje de alto nivel utilizado para muchas cosas como desarrollo de aplicaciones web, Inteligencia artificial, automatizar tareas entre otras cosas. Lo que hace a Python un lenguaje tan amigable es la sintaxis que utiliza ya que es sencilla de comprender y puede ser una muy buena opción para alguien que esté deseando aprender a programar.

Este fue un lenguaje creado a finales de la década de 1980 por el desarrollador Gruido van Russum, este lenguaje ha ganado mucha popularidad y ha tenido un gran crecimiento por medio de todas las versiones que ha tenido a lo largo de estos años.

Unas de las características más importantes de python y por lo que es muy reconocido podríamos decir que son las siguientes.

-Su sintaxis es simple y fácil de leer

-Es multiparadigma

-Tiene una biblioteca muy amplia de funciones

-Comunidad Activa y colaborativa

Por lo que podemos decir que Python es un lenguaje muy versátil que ha ganado terreno en la industria de la programación, la facilidad y su ambiente amigable es lo que hace que muchos programadores se decantan por él. Una vez ya sabemos el lenguaje en el cual nos vamos a desenvolver creo que podemos empezar a listar los elementos que nos van a ser de mucha utilidad en este proyecto.

Listas simple en Python

Estas son una estructura de datos esencial en la programación. Son de las estructuras más versátiles que posee python y más utilizadas, por lo cual su uso será indispensable para el manejo de datos de los archivos que se nos dan.

Estas listas, son flexibles, eficientes y fáciles de utilizar. además que nos sirven para el almacenamiento de Datos, recorrer los datos guardados así como guardarlos en una ubicación específica y para muchos algoritmos está es la base. Por lo que podemos hacernos la pregunta de si las listas simples son la mejor opción de resolución para nuestro programa.

Estructura de una lista simple:

Para saber cómo utilizar una lista simple debemos entender su estructura así podremos sacarle el mayor provecho. Una lista simple consta de dos componentes esenciales.

-Nodo: estas son las estructuras que van conectadas entre sí lo que forma una lista, estos nodos tienen dos partes fundamentales.

Datos o Valor: En esta parte se guarda la información que se desea guardar en la lista y puede ser de cualquier tipo.

Referencia al siguiente nodo: Aquí lo que se tiene es una referencia al siguiente nodo lo que hace que todos estén conectados en orden.

Figura 1. Estructura de una lista simple en python

Fuente: Jóse Pedro Pascual, 2020, analisisyprogramacionoop

Es importante aclarar que el nodo Inicial generalmente nos va servir como referencia ya que este será el punto de partida para acceder a los siguiente datos de la lista hasta el nodo final.

Como podemos observar de esta manera la estructura de una lista es más clara con lo que ya con esto tenemos en la herramienta fundamental que estaremos utilizando para el manejo de datos de nuestro archivo.

Lectura de Archivos en Python

Como sabemos para esta práctica necesitamos saber cómo leer archivos ya que necesitamos manejar la información que estos tienen adentro para poder procesarlo en nuestro programa.

Python nos ofrece una variedad de herramientas y funciones que ya están incorporadas que facilitan esta tarea.

Para Abrir un archivo es necesario utilizar la función open ya esta función debemos pasarle dos parámetros, el nombre del archivo que vamos a abrir y el modo en el que vamos a abrirlo. Esto significa que se puede abrir el archivo en modo lectura, escritura, entre otros.

Luego aparte de abrir el archivo necesitamos leer cada línea del archivo para así poder extraer la información que queremos . Para esto podemos utilizar un bucle ‘for’ lo que nos va permitir iterar a través de las líneas del archivo que hayamos abierto así como el ejemplo que se muestra en La figura 2 abriendo un archivo y recorriendolo línea por línea.

```python
with open("mi_archivo.txt", "r") as archivo:
    for linea in archivo:
        print(linea)
```

Figura 2. Ejemplo de apertura de archivo y recorrido del mismo.

Fuente: Elaboración propia

y así como esta hay muchas más opciones que nos ofrece python para poder manejar archivos.

Graphviz

Así como es necesario manejar la información de una forma precisa también es necesario mostrarla de una forma clara y ordenada y que sea comprensible para el usuario. Esta es una herramienta que se utiliza para la visualización de grafos y estructuras de datos relacionadas, y existe una interfaz para trabajar con Graphviz que se llama 'pydot' que nos permite ver estas estructuras.

Graphviz es un conjunto de herramientas de código abierto que se utiliza para presentaciones visuales de grafos y estructuras. Se pueden crear diagramas de flujo, mapas conceptuales y otros. En nuestro caso nos permitirá visualizar las matrices que son ingresadas en los archivos y que las ubicamos por medio de señales. Una vez ingresadas se hará el manejo de la información para formar las matrices reducidas de las señales y estás deberán de mostrarse de manera gráfica en el programa.

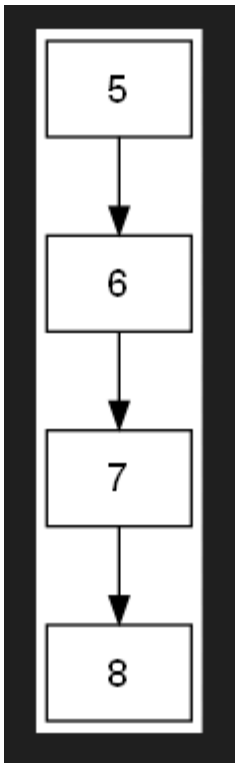

Figura 3. ejemplo de La herramienta Graphviz para visualizar una Lista

Fuente: Elaboración propia

Conclusiones

Para la solución de dicho proyecto es necesario la utilización de una estructura de datos como lo es una lista simplemente enlazada, de otra manera no podríamos manejar la información de una manera correcta.

El manejo de archivos tiene que ir de la mano con la utilización de una lista simple la cual nos permitirá ir guardando la información que se va leyendo línea por línea en el archivo para luego manipularla.

La herramienta Graphviz es fundamental para mostrar el resultado de la matrices de forma gráfica para que el usuario pueda comprender cómo está almacenada la información y cuales son los resultados.

Anexos

```
class ListaSimple():
    id = 0
    def __init__(self):
        self.nodoInicio = None
        self.nodoFinal = None
        self.size = 0
```

Estructura de Lista Simple

```
class Nodo():
    def __init__(self,id, dato):
        self.id = id
        self.dato = dato
        self.siguiente = None
```

Estructura de Nodo

```
class Senal:
    def __init__(self, nombre, tiempoMaximo, amplitudMaxima):
        self.nombre = nombre
        self.tiempoMaximo = tiempoMaximo
        self.amplitudMaxima = amplitudMaxima
        self.tiempos = ListaSimple()
        self.crearListaTiempo()
        self.imprimir()
```

Estructura de la Clase Senal

```
class Graph():
    def __init__(self, nombreArchivo, direccion):
        self.nombreArchivo = nombreArchivo
        self.dot = graphviz.Digraph('structs', file
        self.dot.attr(rankdir=direccion)
```

Estructura de la clase Graph

```
class Amplitud:
    def __init__(self, amplitud, dato=0):
        self.amplitud = amplitud
        self.dato = dato
```

Estructura de la clase Amplitud

```
class Tiempo:
    def __init__(self, tiempo, amplitud):
        self.tiempo = tiempo
        self.amplitud = amplitud
        self.listaAmplitudes = ListaSimple()
        self.llenarListadoAmplitudes()
        self.imprimir()
```

Estructura de la Clase Tiempo

Referencias bibliográficas

C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.

Documentation. (s/f). Graphviz. Recuperado el 7 de septiembre de 2023, de https://graphviz.org/documentation/

Python tutorial. (s/f). W3schools.com. Recuperado el 7 de septiembre de 2023, de https://www.w3schools.com/python/

Stack Overflow - where developers learn, share, & build careers. (s/f). Stack Overflow. Recuperado el 7 de septiembre de 2023, de https://stackoverflow.com/

Welcome to. (s/f). Python.org. Recuperado el 7 de septiembre de 2023, de https://www.python.org/