
PROYECTO 1 - COMPRESIÓN DE SEÑALES DE AUDIO

202201534 – Josseline Griselda Montecinos Hernández

Resumen

La Programación Orientada a Objetos (POO) en Python brinda ventajas destacadas en el desarrollo de soluciones. Además de las estructuras de programación secuenciales, cíclicas y condicionales, la POO introduce clases y objetos, fomentando la abstracción y la estructuración lógica. La visualización de Tipos Abstractos de Datos se facilita con herramientas como Graphviz, que permite representar gráficamente las relaciones entre clases y sus interacciones, mejorando la comprensión del diseño. En conjunto, la implementación de la POO en Python con el respaldo de estructuras de control convencionales y la visualización de TDAs mediante Graphviz, junto con la capacidad de usar archivos XML como entrada, proporciona un enfoque sólido para desarrollar soluciones eficientes y escalables.

Palabras clave

POO
Matriz
Lista
Grafo
TDA

Abstract

Object Oriented Programming (OOP) in Python provides outstanding advantages in solution development. In addition to sequential, cyclic and conditional programming structures, OOP introduces classes and objects, encouraging abstraction and logical structuring. The visualization of Abstract Data Types is facilitated by tools such as Graphviz, which allows to graphically represent the relationships between classes and their interactions, improving the understanding of the design. Overall, implementing OOP in Python with the support of conventional control structures and visualizing ADTs using Graphviz, along with the ability to use XML files as input, provides a robust approach to developing efficient and scalable solutions.

Keywords

OOP
Matrix
List
Graph
ADT

Introducción

Este ensayo se sumerge en el panorama de la programación, explorando la Programación Orientada a Objetos (POO), los Tipos de Datos Abstractos (TDA), Extensión XML y la herramienta de visualización Graphviz. Inicialmente, se analiza Python, un lenguaje de programación de alto nivel utilizado en diversas aplicaciones. Se profundiza en la POO, abordando principios como encapsulación, abstracción, herencia y polimorfismo. Se examinan los TDA como modelos matemáticos para operar con datos, destacando su implementación mediante interfaces y encapsulación. Se introduce Graphviz como una herramienta poderosa para representar visualmente estructuras y relaciones complejas. En síntesis, esta investigación explora la esencia de la programación orientada a objetos, los tipos de datos abstractos y la visualización de datos, resaltando su relevancia en la creación y comprensión de sistemas informáticos y estructuras de datos en Python.

Desarrollo del tema

a. Python

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo, por ejemplo: Instagram, Netflix, Spotify, Panda3D, entre otros. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma. (LUCA, 2020)

```
setOfNumbers = []

print("How many random numbers do you want to generate?")
max = int(input())

for i in range (max):
    setOfNumbers.append(random.randrange(1,101,1))
setOfNumbers.sort()
print(setOfNumbers)

print("Which number do you want to find in the set of random numbers")
searchNumber = int(input())

firstPos = 0
lastPos = max-1
found = False

while (not found and firstPos <= lastPos):
    midPos = int((firstPos + lastPos)/2)

    if (searchNumber == setOfNumbers[midPos]):
        found = True
    else:
        if (searchNumber < setOfNumbers[midPos]):
            lastPos = midPos - 1
        else:
            firstPos = midPos + 1

if (found):
    print("Your item is in the list")
else:
    print("Your item is not in the list")
```

Figura 1. Búsqueda Binaria en python.

Fuente: MsDenning, 2016 Wikipedia.

a.1 Matrices en Python

Las matrices son una estructura de datos bidimensional donde los elementos se organizan en filas y columnas.

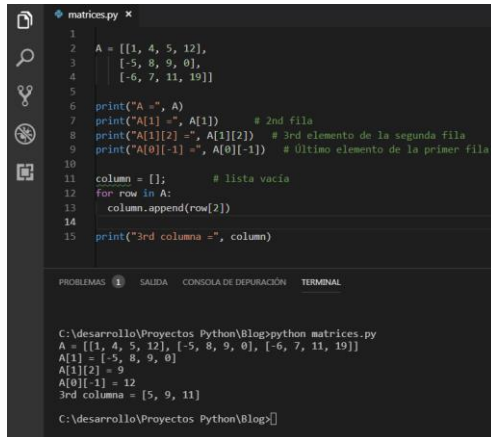
	4 Columnas				
	↓	↓	↓	↓	
	2	-5	-11	0	←
	-9	4	6	13	←
	4	7	12	-2	←
					3 Filas

Figura 2. Ejemplo de Matriz

Fuente: PythonDiario, 2019.

Python no tiene un tipo de dato incorporado para trabajar con matrices, sin embargo, podemos tratar la matriz como una lista de listas.

El uso de las listas anidadas para trabajar como una matriz funciona para tareas simples, sin embargo, a la hora de trabajar con matrices complejas podremos usar el paquete Numpy.



```
1 A = [[1, 4, 5, 12],
2       [-5, 8, 9, 0],
3       [-6, 7, 11, 19]]
4
5
6 print("A =", A)
7 print("A[1] =", A[1]) # 2nd fila
8 print("A[1][2] =", A[1][2]) # 3rd elemento de la segunda fila
9 print("A[0][-1] =", A[0][-1]) # Último elemento de la primer fila
10
11 column = [] # lista vacía
12 for row in A:
13     column.append(row[2])
14
15 print("3rd columna =", column)
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
C:\desarrollo\Proyectos Python\Blog\python matrices.py
A = [[1, 4, 5, 12], [-5, 8, 9, 0], [-6, 7, 11, 19]]
A[1] = [-5, 8, 9, 0]
A[1][2] = 9
A[0][-1] = 12
3rd columna = [5, 9, 11]
C:\desarrollo\Proyectos Python\Blog\[]
```

Figura 3. Ejemplo de Matriz como Lista

Fuente: PythonDiario, 2019.

b. POO

La Programación Orientada a Objetos (POO) representa un enfoque en la programación que se basa en el concepto de clases y objetos, formando un paradigma o modelo para desarrollar software. En lugar de un enfoque lineal, la POO organiza el código en unidades llamadas clases, que son como plantillas reutilizables, y a partir de estas clases se crean instancias individuales, conocidas como objetos. Esta metodología estructura un programa en componentes más manejables y simples, promoviendo el modularidad y la eficiencia en el diseño de software.

Históricamente han ido surgiendo distintos paradigmas de programación. Por un lado, los lenguajes secuenciales como COBOL o procedimentales como Basic o C, se enfocan más en la lógica que en los datos. Por otro lado, otros más modernos como Java, C# y Python, usan paradigmas para definir los programas, siendo la POO la más popular.

Con el paradigma de POO lo que se busca es dejar de centrarse en la lógica pura de los programas, para comenzar a pensar en objetos, lo que forma la base de dicho paradigma. Esto ayuda bastante en sistemas grandes, pues en lugar de pensar en funciones, se piensa en las relaciones o interacciones de los distintos elementos del sistema. (Darias, 2021)

b.1 Principios de la POO

- *Encapsulación:* presenta toda la información importante de un objeto dentro del mismo y solo expone la información elegida al mundo exterior.
Esta propiedad permite asegurar que la información de un objeto esté oculta para el mundo exterior, agrupando en una clase las características o atributos que tienen un acceso privado, y los comportamientos o métodos que cuenta con un acceso público.
- *Abstracción:* se produce cuando el usuario interactúa solo con los atributos y métodos seleccionados de un objeto, usando herramientas simplificadas de alto nivel para acceder a un objeto complejo.
En la POO, los programas suelen ser muy grandes y los objetos se comunican bastante entre sí. De este modo, la abstracción facilita el mantenimiento de un código de gran tamaño, donde pueden surgir distintos cambios con el paso del tiempo.
- *Herencia:* define relaciones jerárquicas entre clases, de modo que atributos y métodos

comunes puedan ser reutilizados. Las clases principales extienden atributos y comportamientos a las clases secundarias. Mediante la definición en una clase de los atributos y comportamientos básicos, pueden crearse clases secundarias, ampliando la funcionalidad de la clase principal y añadiendo atributos y comportamientos extra. Es una de las claves de la Programación Orientada a Objetos.

- *Polimorfismo*: El polimorfismo reside en diseñar objetos para compartir comportamientos, lo que permite procesar objetos de distintos modos. Es la capacidad de presentar la misma interfaz para distintas maneras subyacentes o tipos de datos. (Darias, 2021)

c. TDA's

Un Tipo de Dato Abstracto (TDA) o también conocido como Tipo Abstracto de Datos, se refiere a un modelo matemático que incluye un conjunto de operaciones definidas para trabajar con un conjunto específico de datos. Cuando se aplica en un programa informático, se presenta mediante una interfaz que actúa como una capa que cubre su implementación subyacente. La idea central es que los usuarios de un TDA solo necesiten interactuar con su interfaz, sin preocuparse por cómo se realiza internamente, ya que esta implementación puede variar con el tiempo. Sin la encapsulación, los cambios en la implementación podrían tener un impacto en los programas que utilizan el dato. Esto se

basa en el principio de Ocultación de Información, que salvaguarda al programa de decisiones de diseño que podrían cambiar con el tiempo.

Esto implica que el TDA puede ser realizado de varias maneras, siempre y cuando se mantenga coherente con la interfaz, los programas que lo emplean permanecen inalterados. Existe una distinción, a veces sutil, entre el Tipo de Dato Abstracto y la Estructura de Datos utilizada en su realización. Por ejemplo, un TDA que define una lista puede ser llevado a cabo mediante un Arreglo, una Lista Enlazada o incluso un Árbol binario de búsqueda. La lista en sí es un Tipo de Dato Abstracto con operaciones claramente establecidas (agregar elemento, añadir al final, introducir al principio, recuperar, eliminar, etc.), mientras que una lista enlazada es una estructura de datos basada en punteros o referencias (según el lenguaje) que puede utilizarse para crear una representación de una lista. La Lista Enlazada a menudo se emplea para representar una TDA Lista, a veces dando lugar a confusiones. (Blog Algoritmos y Programación, 2021)

d. XML

XML es el acrónimo de Extensible Markup Language, es decir, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.

El lenguaje de marcado es un conjunto de códigos que se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas. El lenguaje XML proporciona una plataforma para definir elementos para crear un formato y generar un lenguaje personalizado.

Un archivo XML se divide en dos partes: prolog y body. La parte prolog consiste en metadatos administrativos, como declaración XML, instrucción de procesamiento opcional, declaración de tipo de documento y comentarios. La parte del body se compone de dos partes: estructural y de contenido (presente en los textos simples).

El diseño XML se centra en la simplicidad, la generalidad y la facilidad de uso y, por lo tanto, se utiliza para varios servicios web. Tanto es así que hay sistemas destinados a ayudar en la definición de lenguajes basados en XML, así como APIs que ayudan en el procesamiento de datos XML - que no deben confundirse con HTML.

e. Graphviz

Es un conjunto de herramientas open-source realizado inicialmente en los laboratorios de investigación de AT&T para el dibujo de gráficos especificados en lenguaje de scripts DOT. Provee librerías para ser usadas por otras aplicaciones. Graphviz es software libre licenciado bajo CPL (Common Public License).

Sus aplicaciones son:

- Estructuras de datos.
- Estructuras de árbol.
- Representación de análisis social de redes.
- Diagramas entidad relación.
- Diagramas de redes.
- Diagramas de flujo.
- Diagramas de procesos.

Un grafo es un conjunto de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas que pueden ser orientados o no. Típicamente, un grafo se representa mediante una serie de puntos (los vértices) conectados por líneas (las aristas). (Gomez, 2014)

Graphviz es una colección de software para representar grafos, muy flexible, pero requiere un cierto esfuerzo de aprendizaje.

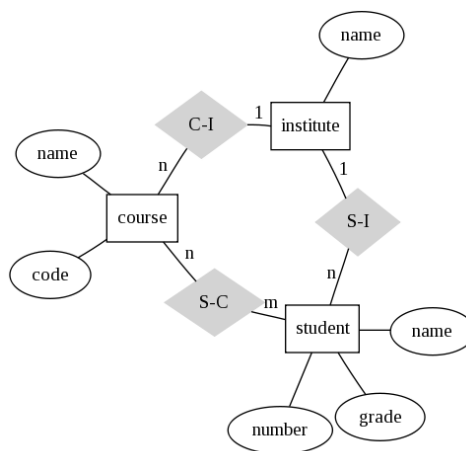


Figura 4. Ejemplo Diagrama Entidad Relación

Fuente: NEATO, 2009

Conclusiones

La visualización de Tipos Abstractos de Datos (TDA) mediante Graphviz añade claridad al diseño, lo que es esencial dada la creciente complejidad de las aplicaciones modernas. Asimismo, el uso de archivos XML como insumos confiere flexibilidad y permite configuraciones externas.

La idea central es que los usuarios de un TDA solo necesiten interactuar con su interfaz, sin preocuparse por cómo se realiza internamente, ya que esta implementación puede variar con el tiempo. Sin la encapsulación, los cambios en la implementación podrían tener un impacto en los programas que utilizan el dato.

Además, las listas simples enlazadas se presentan como una estructura de datos fundamental en Python, aportando flexibilidad y eficiencia en la gestión de información. Su implementación permite la creación de estructuras dinámicas y adaptables, ideales para situaciones en las que se requiere una gestión eficaz de datos variables. En resumen, las listas simples enlazadas en Python enriquecen la capacidad de representación y manipulación de datos, brindando una herramienta poderosa para el diseño de algoritmos y aplicaciones más versátiles y dinámicas.

Referencias bibliográficas

Blog Algoritmos y Programación. (2021). *TDA (Tipos de Datos Abstractos)*. Blogspot. Recuperado el 27 de agosto de 2023, de Blogspot: <http://blogalgoritmosyprogramacion.blogspot.com/12/07/tda-tipos-de-datos-abstractos.html>

Darias, S. (26 de noviembre de 2021). *Qué es la Programación Orientada a Objetos*. Intelequia.

Recuperado el 27 de agosto de 2023, de Intelequia: <https://intelequia.com/blog/post/qu%C3%A9-es-la-programaci%C3%B3n-orientada-a-objetos>

Gomez, O. (14 de marzo de 2014). *Visualizando Grafos usando Graphviz*. Osiux. Recuperado el 27 de agosto de 2023, de Osiux: <https://osiux.com/visualizando-grafos-graphviz.org>

LUCA. (24 de febrero de 2020). Python. Wikipedia. Recuperado el 27 de agosto de 2023, de Wikipedia: <https://es.wikipedia.org/wiki/Python>

Anexos

A				
	1	2	3	4
T	1	2	3	0
	2	0	0	6
	3	3	4	0
	4	1	0	1
	5	0	0	3

	1	2	3	4
1	1	1	0	1
2	0	0	1	1
3	1	1	0	1
4	1	0	1	1
5	0	0	1	1

LISTA DE PATRONES				
1 TIEMPO1	AMPLITUD1	VALOR1		
2 TIEMPO2	AMPLITUD2	VALOR2		

	1	2	3	4
1	1	1	0	1
2	0	0	1	1
3	1	1	0	1
4	1	0	1	1
5	0	0	1	1

	1	2	3	4
1	1	1	0	1
2	0	0	1	1
3	1	1	0	1
4	1	0	1	1
5	0	0	1	1

	1	2	3	4
1	1	1	0	1
2	0	0	1	1
3	1	1	0	1
4	1	0	1	1
5	0	0	1	1

	1	2	3	4
1	1	1	0	1
2	0	0	1	1
3	1	1	0	1
4	1	0	1	1
5	0	0	1	1

Figura 5. Análisis de matrices como celdas en Excel

Fuente: Creación propia, 2023

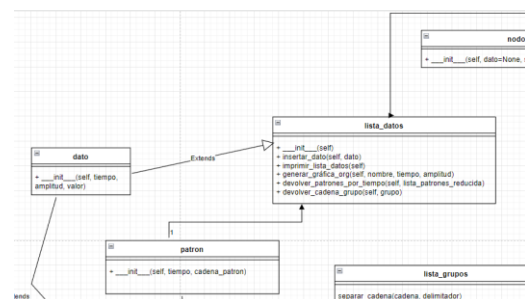


Figura 6. Diagrama de Clases Parte 1

Fuente: Creación propia, 2023

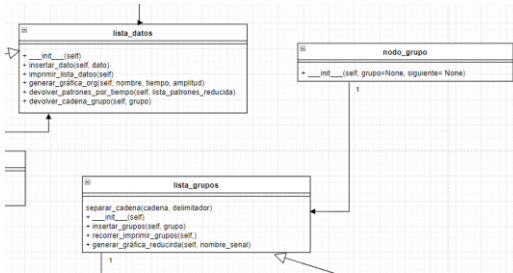


Figura 7. Diagrama de Clases Parte 2

Fuente: Creación propia, 2023

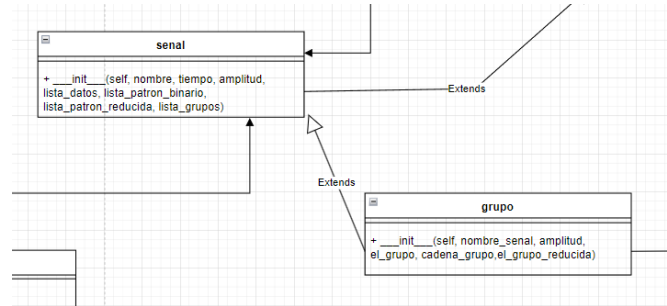


Figura 10. Diagrama de Clases Parte 5

Fuente: Creación propia, 2023

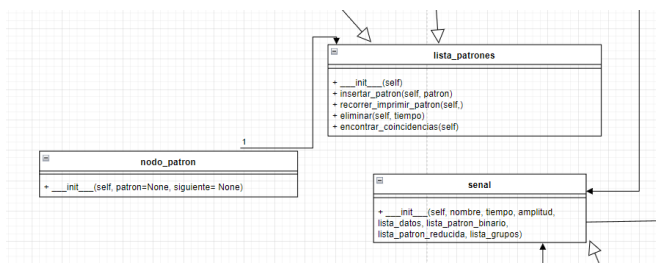


Figura 8. Diagrama de Clases Parte 3

Fuente: Creación propia, 2023

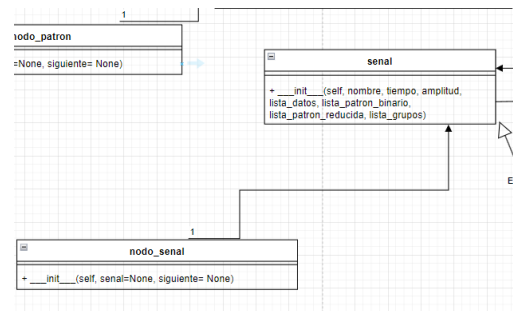


Figura 11. Diagrama de Clases Parte 6

Fuente: Creación propia, 2023

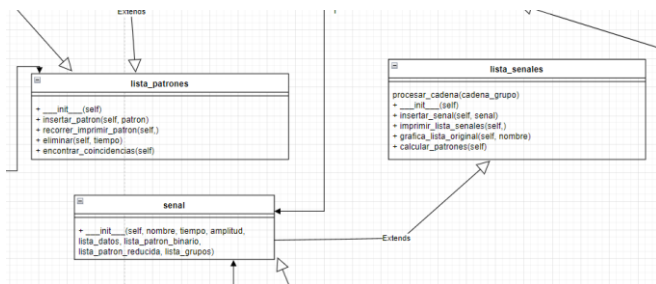


Figura 9. Diagrama de Clases Parte 4

Fuente: Creación propia, 2023

Enlace para Diagrama de clases completo:
<https://drive.google.com/file/d/1X9KLfsA23FGuSUQojuicgPkSIEfRkCG/view?usp=sharing>