Implementación de Árboles Binarios en Python

Alumnos:

Alex Pereyra – [lexthus@gmail.com](mailto:lexthus@gmail.com)  
 Lorenzo Ojeda – [lorenzomkt.0@gmail.com](mailto:lorenzomkt.0@gmail.com)

Materia: Programación I

Profesor: AUS Bruselario, Sebastián  
Tutora: Cimino, Virginia

Fecha de entrega: 09/06/2025

Contenido

[1. Introducción 3](#_Toc200213250)

[2. Marco Teórico 3](#_Toc200213251)

[3. Caso Práctico 3](#_Toc200213252)

[4. Metodología Utilizada 3](#_Toc200213253)

[5. Resultados Obtenidos 3](#_Toc200213254)

[6. Conclusiones 3](#_Toc200213255)

[7. Bibliografía 3](#_Toc200213256)

[8. Anexos 4](#_Toc200213257)

# 1. Introducción

Este trabajo aborda la implementación de árboles binarios en Python utilizando listas como estructura principal. Se eligió este enfoque por su valor didáctico y su aplicabilidad en escenarios donde se requiere una jerarquización eficiente de datos. Los objetivos del trabajo incluyen comprender la estructura de los árboles binarios, realizar recorridos comunes, y representar visualmente el árbol.

# 2. Marco Teórico

Un árbol binario es una estructura jerárquica donde cada nodo tiene a lo sumo dos hijos: uno izquierdo y uno derecho. Cada nodo almacena un valor y enlaces a sus hijos. Los recorridos típicos son Preorden, Inorden y Postorden. Esta estructura es utilizada en múltiples áreas como bases de datos, sistemas de archivos y algoritmos de búsqueda.

# 3. Caso Práctico

Se construyó un árbol binario con nodos A-G y se implementaron funciones para insertarlos y recorrerlos. También se generó una función para visualizar el árbol rotado 90 grados, facilitando su interpretación jerárquica. El código completo se encuentra en el archivo arbol\_binario.py y en el repositorio GitHub.

# 4. Metodología Utilizada

El desarrollo se realizó con Python 3.x. Se usaron listas anidadas para representar nodos. Las funciones fueron desarrolladas en etapas: creación del árbol, inserciones, recorridos y visualización. Se realizaron pruebas unitarias con salida en consola para validar resultados.

# 5. Resultados Obtenidos

El árbol fue construido correctamente, y los recorridos generaron el orden esperado. La visualización rotada permitió validar visualmente la estructura. Los recorridos Preorden, Inorden y Postorden se imprimieron sin errores.

# 6. Conclusiones

La implementación de árboles con listas resulta efectiva para fines educativos. Permite abstraer la lógica sin requerir conocimientos avanzados de programación orientada a objetos. Se logró cumplir con todos los objetivos propuestos y se comprendieron los fundamentos de esta estructura.

# 7. Bibliografía

- Cormen, T. et al. (2009). Introduction to Algorithms.

- Python Software Foundation. https://docs.python.org/3/

- Miller, B. & Ranum, D. Problem Solving with Algorithms and Data Structures using Python.

# 8. Anexos

- Capturas de pantalla del árbol y los recorridos.

- Link al video:

- Repositorio GitHub: https://github.com/alexmp2602/tp-integrador-arboles-python