Trabajo Integrador Arboles en Python

**Datos Generales**

* *Alumnos:*   
  Giannoni Gabriel - gabrielgiannoni96@gmail.com

López Emanuel – emaguslopez@gmail.com

* *Tema Elegido*:

Árboles en Python

* *Materia*:

programación l

* *Profesor/a*:

Cinthia Rigoni

* *Fecha de Entrega*:

09/06/2025

**Índice**

1. Introducción
2. Marco Teórico
3. Caso Practico
4. Metodología Utilizada
5. Resultados Obtenidos
6. Conclusiones
7. Bibliografía
8. Anexos

**1.Introducción**

**¿Porque se eligió el tema?**

Se elegio el tema de árboles porque es un tema nuevo que desconocen los integrantes del grupo.   
   
 **¿Qué importancia tiene en la programación?**

Son ampliamente utilizados en informática para organizar datos de forma eficiente.   
   
 **¿Qué objetivos se propone alcanzar con el desarrollo del trabajo?**

Llevar adelante una investigación práctica y aplicada sobre conceptos fundamentales y avanzados del lenguaje Python, integrando teoría, casos de uso reales, desarrollo de software y difusión de los resultados obtenidos.

**2.Marco Teórico**

Los árboles en la programación es una estructura de datos no lineal y que organiza los elementos de forma jerárquica donde contiene listas enlazadas bidimensionalmente. Cada nodo tiene un valor y apunta a dos subárboles, uno a la izquierda y otro a la derecha.

**Algunos conceptos para entender un árbol**

Nodo: es un centro de información de datos que contiene los datos que se desea en su interior, puede contener cualquier tipo de datos, desde números hasta string, etc.

Rama: son las que conectan a los nodos.

**Clasificación de nodos**

La clasificación de nodos cumple dos criterios, en primer lugar, su relación con otros nodos (raíz, padre, hijo, hermano, etc.). Y en segundo lugar su posición en el árbol.

Nodo raíz: no tiene padre

Nodo hoja: no tiene hijos

Nodo rama: tiene padre e hijo

**Propiedades de los arboles**

*Longitud de camino*: se define como la cantidad de ramas que tenemos que recorrer para llegar de un nodo a otro.  
 *Profundidad*: se hace un análisis similar a la longitud de camino, pero se nombre solo un nodo, porque se sobre entiende que el otro extremo siempre es el nodo raíz.  
 *Nivel*: se calcula sencillamente la profundidad mas 1 unidad, si la profundidad de un nodo “x” es 2, el nivel es 3.  
 *Altura:* va a estar dada como la longitud de camino entre el nodo raíz y la hoja que este más lejos de él, si la longitud de camino de “x” es 2, la altura es 2.  
 *Grado*: es simplemente cuantos hijos tiene el nodo.  
 *Orden*: es la máxima cantidad de hijos que puede tener un árbol, es una restricción que ponemos antes de construir el árbol.  
Peso: es la cantidad de nodos que tiene el árbol. Interfaz de usuario gráfica, Diagrama

El contenido generado por IA puede ser incorrecto.

**Recorrido de un árbol**

Hay tres tipos de recorrido de árboles: Preorden, inorden, postorden.

*Preorden*:   
 El recorrido de Preorden se empieza por la raíz y va hacia las hojas, es el único método que empieza de la raíz, dando prioridad al subárbol izquierdo(hijo).  
Realiza tres pasos de forma recursiva:

* 1. Visita el nodo actual(N)
  2. Recorrer recursivamente el subárbol izquierdo(L) del nodo actual.
  3. Recorrer recursivamente el subárbol derecho(R)del nodo actual.

Dibujo animado de un personaje de caricatura

El contenido generado por IA puede ser incorrecto.

Ejemplo de recorrido finalizado (Ver flechas en color rojo):

['a', 'b', 'd', 'e', 'g', 'h', 'c', 'f']

función en Python:

Texto

El contenido generado por IA puede ser incorrecto.

*Inorden*:  
 El recorrido de Inorden primero empieza por la hoja más a la izquierda del subárbol izquierdo, luego visita al padre y luego a la hoja mas a la izquierda, así sucesivamente, una vez nos quedamos sin hojas del subárbol izquierdo, pasamos a la raíz y empezamos la misma secuencia, pero por el subárbol derecho.

Diagrama

El contenido generado por IA puede ser incorrecto.

Inorden: 

Función en Python:

Texto

El contenido generado por IA puede ser incorrecto.

*Postorden*:  
 El recorrido empieza de la hoja que esta más a la izquierda de todas, se recorren todos los hermanos posibles y luego al padre que los une. En esta parta ya terminamos de recorrer el subárbol izquierdo asique nos pasamos al derecho yendo a la hoja más a la izquierda nuevamente, recorremos hermanos y padre y al final de todo a la raíz.

  
Un dibujo de un perro

El contenido generado por IA puede ser incorrecto.

Postorden: [ ' d ' , ' g ' , ' h ' , ' e ' , ' b ' , ' f ' , ' c ' , ' a ' ]

Funcion en python

Texto

El contenido generado por IA puede ser incorrecto.

**Casos de uso**:

* Árboles genealógicos
* Bases de datos jerárquicos
* Algoritmo de búsqueda y toma de decisiones
* Organización Jerárquica de una empresa
* Fixture de torneos deportivos
* Sistema de archivos

**3.Caso Practico**

En el caso práctico veremos como es la sintaxis en Python de un árbol en cuanto a funcionalidades que necesitaremos para agregar y recorrer los elementos del árbol.

Funcionalidades a utilizar:  
Texto

El contenido generado por IA puede ser incorrecto.

Programa Principal

Texto

El contenido generado por IA puede ser incorrecto.

**4.Metodologia utilizada**

Se utilizará Python3 como entorno de desarrollo.  
Se implementará un algoritmo para crear un árbol y 3 nodos con una lista de tres elementos: valor, subárbol izquierdo, subárbol derecho.  
Se desarrollarán las funciones necesarias para la inserción de nodos, recorridos y visualización

**5.Resultados obtenidos**

Los resultados fueron los deseados, pudimos hacer los recorridos sin problemas, y pudimos verificar el uso correcto del recorrido al momento de utilizar la función.

Texto

El contenido generado por IA puede ser incorrecto.  
 Imagen ilustrativa de los resultados del algoritmo

**6.Bibliografía**

<https://www.guiadearbolado.com.ar/recorrido-en-orden-de-un-arbol-binario/>

<https://www-geeksforgeeks-org.translate.goog/binary-tree-in-python/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc>

<https://sites.google.com/site/programacioniiuno/temario/unidad-5---grafos/rboles>

<https://www-openbookproject-net.translate.goog/py4fun/bintree/bintree.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc>

<https://docs.python.org/es/3.10/library/ast.html>

<https://builtin-com.translate.goog/articles/tree-python?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc>

<https://runestone.academy/ns/books/published/pythoned/Trees/ImplementacionArbolBusqueda.html>