



TECNOLÓGICO  
NACIONAL DE MÉXICO



ING. SISTEMAS COMPUTACIONALES

INTELIGENCIA ARTIFICIAL 11-12 A.M

DOCENTE: ZURIEL DATHAN MORA FELIZ

UNIDAD 1

ARBOL BINARIO PYTHON

JIMENEZ VELZQUEZ ZETH ODIN ALFONSO

## Clase Nodo

La clase `Nodo` representa un nodo en el árbol binario. Cada nodo tiene:

- `dato`: el valor almacenado.
- `izquierda`: referencia al nodo hijo izquierdo.
- `derecha`: referencia al nodo hijo derecho.

```
class Nodo:
    def __init__(self, dato):
        self.dato = dato
        self.izquierda = None
        self.derecha = None
```

## Clase Arbol

La clase `Arbol` representa un árbol binario de búsqueda (ABB). Contiene:

- Un nodo raíz.
- Métodos privados para agregar, buscar e imprimir el árbol en diferentes órdenes.
- Métodos públicos para interactuar con el árbol.

### Constructor

```
def __init__(self, dato):
    self.raiz = Nodo(dato)
```

- Inicializa el árbol con un nodo raíz.

### Método privado `__agregar_recursivo`

```
def __agregar_recursivo(self, nodo, dato):
    if dato < nodo.dato:
        if nodo.izquierda is None:
            nodo.izquierda = Nodo(dato)
        else:
```

```

        self.__agregar_recursivo(nodo.izquierda, dato)
    else:
        if nodo.derecha is None:
            nodo.derecha = Nodo(dato)
        else:
            self.__agregar_recursivo(nodo.derecha, dato)

```

- Si dato es menor que `nodo.dato`, lo coloca en la izquierda.
- Si dato es mayor o igual, lo coloca en la derecha.
- Se llama recursivamente hasta encontrar una posición disponible.

## Métodos de recorrido

Estos métodos imprimen el árbol en distintos órdenes.

### ***Inorden (Izquierda - Nodo - Derecha)***

```

def __inorden_recursivo(self, nodo):
    if nodo is not None:
        self.__inorden_recursivo(nodo.izquierda)
        print(nodo.dato, end=", ")
        self.__inorden_recursivo(nodo.derecha)

```

- Imprime los nodos en orden ascendente.

### ***Preorden (Nodo - Izquierda - Derecha)***

```

def __preorden_recursivo(self, nodo):
    if nodo is not None:
        print(nodo.dato, end=", ")
        self.__preorden_recursivo(nodo.izquierda)
        self.__preorden_recursivo(nodo.derecha)

```

- Primero el nodo, luego los hijos.

### ***Postorden (Izquierda - Derecha - Nodo)***

```

def __postorden_recursivo(self, nodo):
    if nodo is not None:
        self.__postorden_recursivo(nodo.izquierda)

```

```
self.__postorden_recursivo(nodo.derecha)
print(nodo.dato, end=" ", " ")
```

- Primero los hijos, luego el nodo.

## Método `__buscar`

```
def __buscar(self, nodo, busqueda):
    if nodo is None:
        return None
    if nodo.dato == busqueda:
        return nodo
    if busqueda < nodo.dato:
        return self.__buscar(nodo.izquierda, busqueda)
    else:
        return self.__buscar(nodo.derecha, busqueda)
```

- Si `nodo.dato` es igual a `busqueda`, lo encuentra.
- Si `busqueda` es menor, busca a la izquierda.
- Si `busqueda` es mayor, busca a la derecha.

## Métodos públicos

Estos métodos usan los privados para interactuar con el árbol.

```
def agregar(self, dato):
    self.__agregar_recursivo(self.raiz, dato)
```

- Llama a `__agregar_recursivo` para insertar un nodo.

```
def inorden(self):
    print("Imprimiendo árbol inorden: ")
    self.__inorden_recursivo(self.raiz)
    print("")
```

- Llama a `__inorden_recursivo` y lo imprime.

```
def preorden(self):
    print("Imprimiendo árbol preorden: ")
    self.__preorden_recursivo(self.raiz)
```

```
print("")
```

- Llama a `__preorden_recursivo` y lo imprime.

```
def postorden(self):  
    print("Imprimiendo árbol postorden: ")  
    self.__postorden_recursivo(self.raiz)  
    print("")
```

- Llama a `__postorden_recursivo` y lo imprime.

```
def buscar(self, busqueda):  
    return self.__buscar(self.raiz, busqueda)
```

- Llama a `__buscar` y retorna el nodo si existe.

## Uso del árbol (main)

### Creación del árbol con nombres

```
from arbol import Arbol  
  
arbol = Arbol("Luis")  
arbol.agregar("María José")  
arbol.agregar("Maggie")  
arbol.agregar("Leon")  
arbol.agregar("Cuphead")  
arbol.agregar("Aloy")  
arbol.agregar("Jack")
```

- Se crea un árbol con nombres en orden aleatorio.

### Agregar un nodo con input

```
nombre = input("Ingresa algo para agregar al árbol: ")  
arbol.agregar(nombre)
```

- Permite al usuario agregar un nombre al árbol.

## Imprimir el árbol en distintos órdenes

```
arbol.preorden()  
arbol.inorden()  
arbol.postorden()
```

- Imprime el árbol en preorden, inorden y postorden.

## Búsqueda de un nodo

```
busqueda = input("Busca algo en el árbol: ")  
nodo = arbol.buscar(busqueda)  
if nodo is None:  
    print(f"{busqueda} no existe")  
else:  
    print(f"{busqueda} sí existe")
```

- Permite buscar un nombre en el árbol.

## Creación de un árbol con números

```
arbol_numeros = Arbol(5)  
arbol_numeros.agregar(1984)  
arbol_numeros.agregar(60)  
arbol_numeros.agregar(10)  
arbol_numeros.agregar(20)  
arbol_numeros.agregar(10)  
arbol_numeros.agregar(25)  
arbol_numeros.agregar(59)  
arbol_numeros.agregar(64)  
arbol_numeros.agregar(10)  
arbol_numeros.agregar(19)  
arbol_numeros.agregar(23)  
arbol_numeros.agregar(18)  
arbol_numeros.agregar(1)  
arbol_numeros.agregar(2013)
```

- Se crea un árbol binario con números.

## Imprimir el árbol

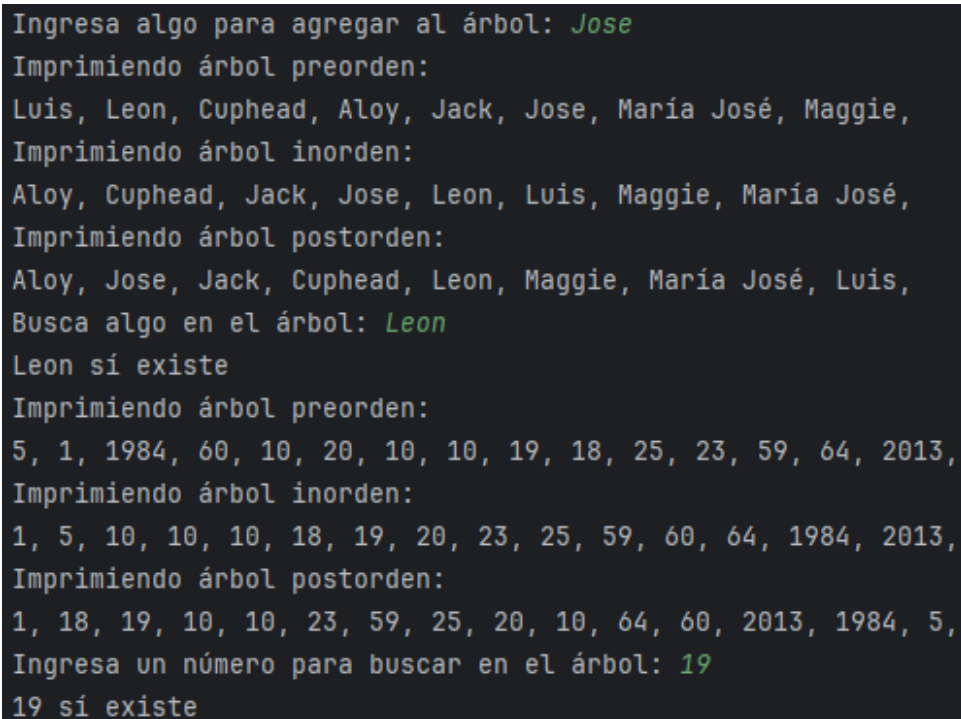
```
arbol_numeros.preorden()  
arbol_numeros.inorden()  
arbol_numeros.postorden()
```

- Se imprimen los números en los distintos órdenes.

## Buscar un número

```
busqueda = int(input("Ingresa un número para buscar en el árbol:  
"))  
n = arbol_numeros.buscar(busqueda)  
if n is None:  
    print(f"{busqueda} no existe")  
else:  
    print(f"{busqueda} sí existe")
```

- Permite buscar un número en el árbol.



```
Ingresa algo para agregar al árbol: Jose  
Imprimiendo árbol preorden:  
Luis, Leon, Cuphead, Aloy, Jack, Jose, María José, Maggie,  
Imprimiendo árbol inorden:  
Aloy, Cuphead, Jack, Jose, Leon, Luis, Maggie, María José,  
Imprimiendo árbol postorden:  
Aloy, Jose, Jack, Cuphead, Leon, Maggie, María José, Luis,  
Busca algo en el árbol: Leon  
Leon sí existe  
Imprimiendo árbol preorden:  
5, 1, 1984, 60, 10, 20, 10, 10, 19, 18, 25, 23, 59, 64, 2013,  
Imprimiendo árbol inorden:  
1, 5, 10, 10, 10, 18, 19, 20, 23, 25, 59, 60, 64, 1984, 2013,  
Imprimiendo árbol postorden:  
1, 18, 19, 10, 10, 23, 59, 25, 20, 10, 64, 60, 2013, 1984, 5,  
Ingresa un número para buscar en el árbol: 19  
19 sí existe
```

En este screen es un caso de éxito tanto en el nombre como en el número donde se encontró dicho elemento en el árbol.

```
Ingresa algo para agregar al árbol: Zeth
Imprimiendo árbol preorden:
Luis, Leon, Cuphead, Aloy, Jack, María José, Maggie, Zeth,
Imprimiendo árbol inorden:
Aloy, Cuphead, Jack, Leon, Luis, Maggie, María José, Zeth,
Imprimiendo árbol postorden:
Aloy, Jack, Cuphead, Leon, Maggie, Zeth, María José, Luis,
Busca algo en el árbol: chuyon
chuyon no existe
Imprimiendo árbol preorden:
5, 1, 1984, 60, 10, 20, 10, 10, 19, 18, 25, 23, 59, 64, 2013,
Imprimiendo árbol inorden:
1, 5, 10, 10, 10, 18, 19, 20, 23, 25, 59, 60, 64, 1984, 2013,
Imprimiendo árbol postorden:
1, 18, 19, 10, 10, 23, 59, 25, 20, 10, 64, 60, 2013, 1984, 5,
Ingresa un número para buscar en el árbol: 999
999 no existe
```

En la siguiente screen es el caso contrario donde no se encontró ni el nombre ni el número.