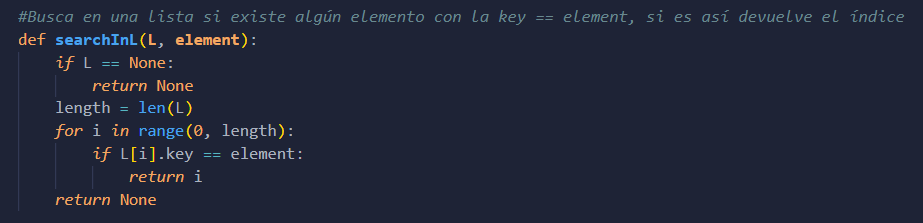
**TP TRIE**

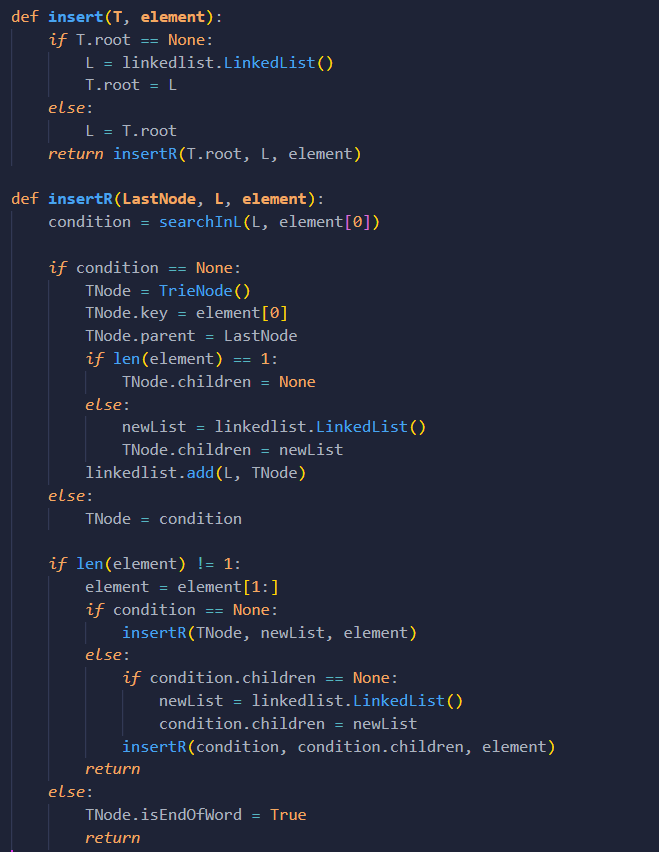
**Tomás Rando - 14004 - LCC**

Para la realización del TP hice dos implementaciones. Una con listas de python y otra con listas LinkedList. En la segunda mitad del tp se encuentran las implementaciones con LinkedList y en GitHub se encuentran los códigos correspondientes junto a un archivo main.py que contiene algunos casos de prueba para ambas implementaciones.

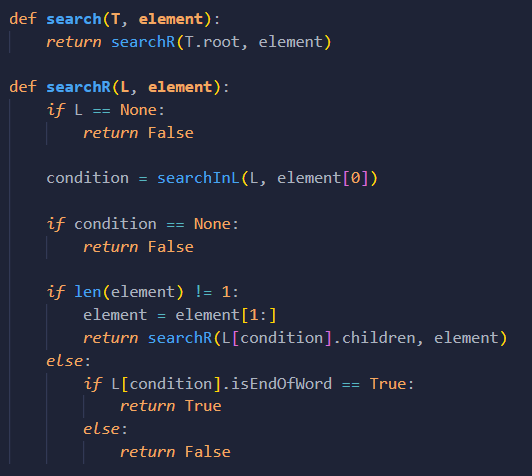
**Punto 1)**

****

**insert(T, element)**



**search(T, element)**

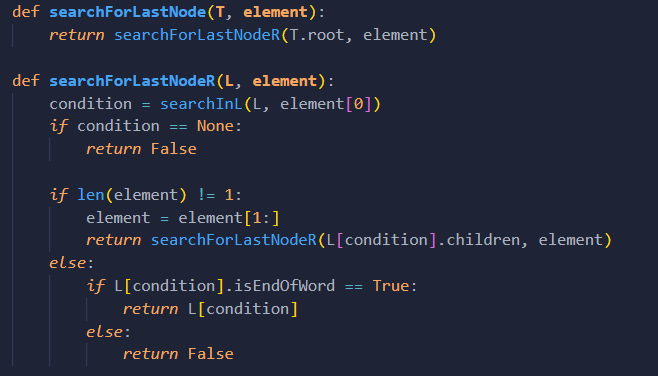


**Punto 2)**

Una versión del search() cuya complejidad sea O(m) sería una versión del Trie implementado mediante arrays en vez de listas enlazadas (utilizando posiciones conocidas de antemano para insertar las letras). De esta manera, el acceso a cada letra sería de O(1) y solo importaría recorrer la palabra, que se realizaría en O(m), siendo m la longitud de la palabra

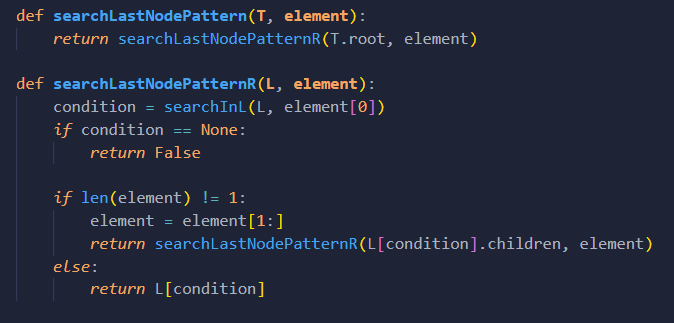
**Punto 3)**

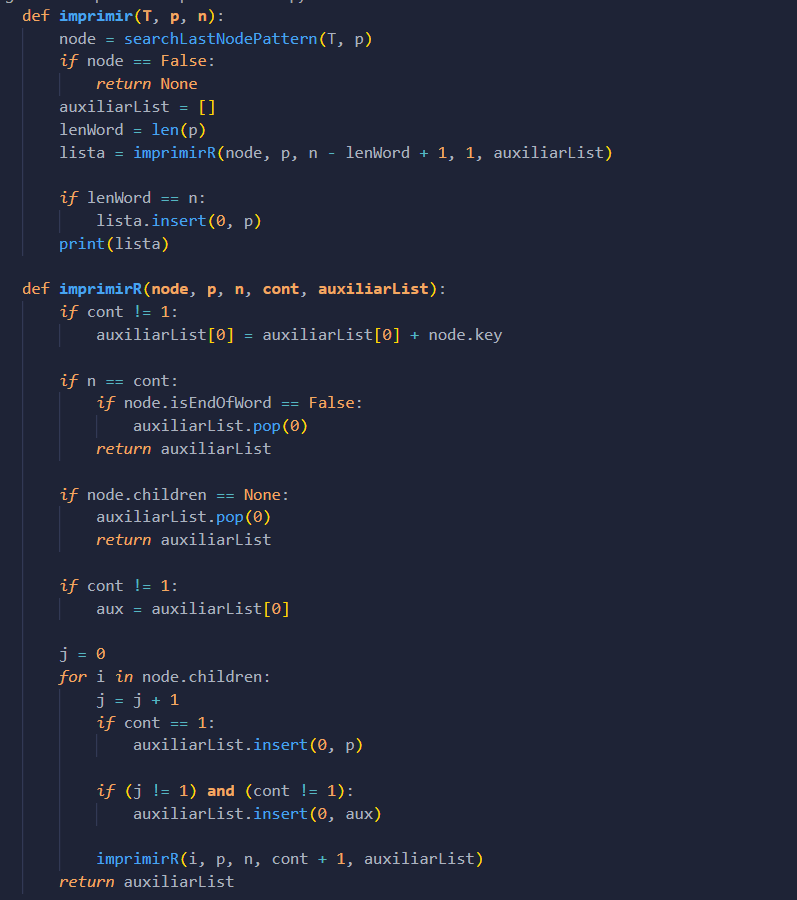
**delete(T, element)**





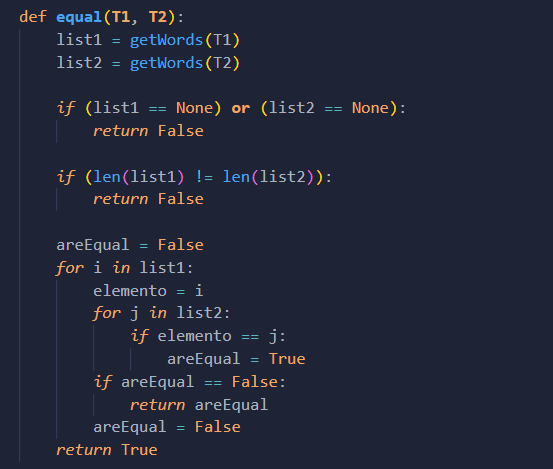
**Punto 4)**





**Punto 5)**

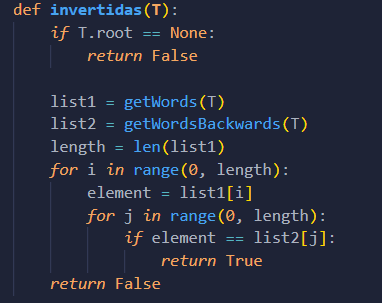




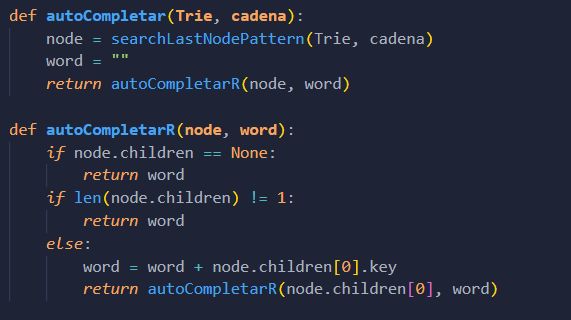
La complejidad es de O(m^2), siendo m la cantidad de palabras de los Trie, ya que se comparan las dos listas al final usando dos bucles anidados que recorren la longitud de las 2 listas.

**Punto 6)**



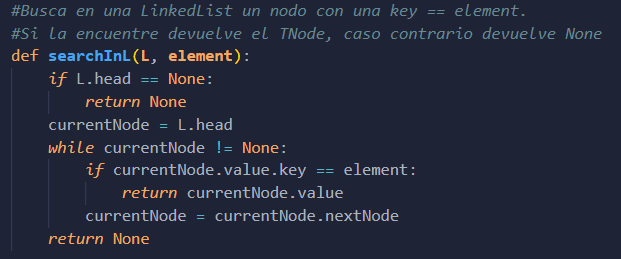


**Punto 7)**

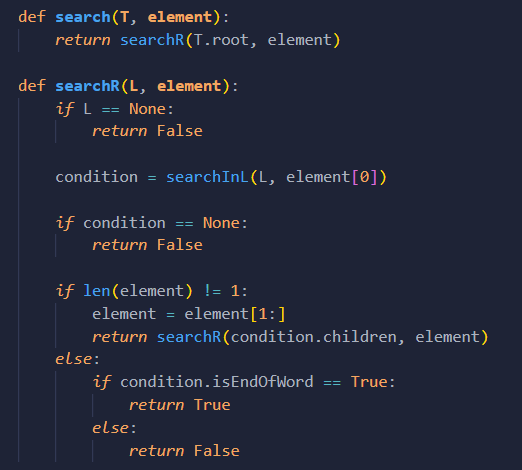


**IMPLEMENTACIONES CON LINKEDLIST**

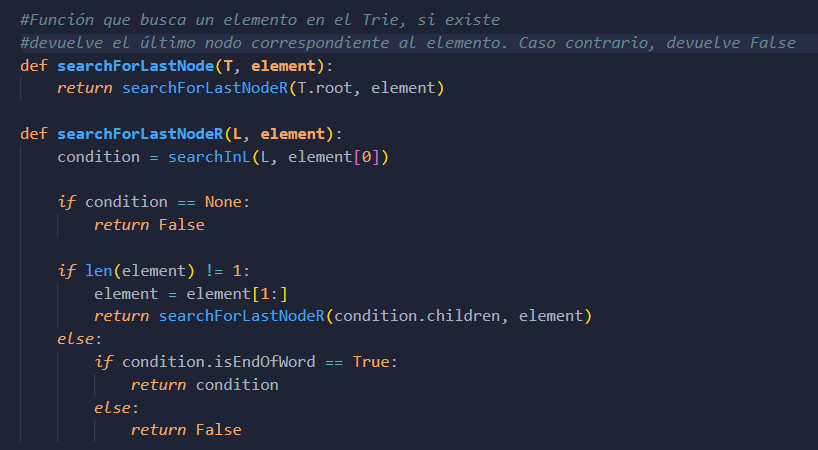
**Punto 1)**





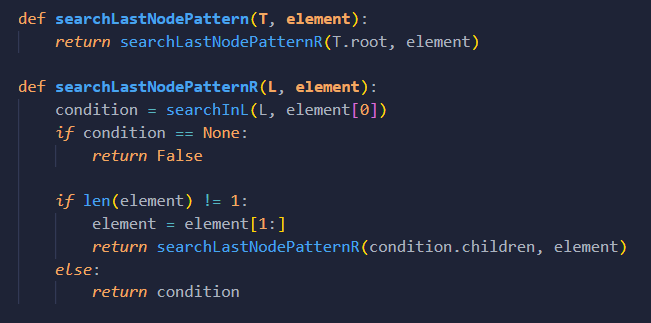


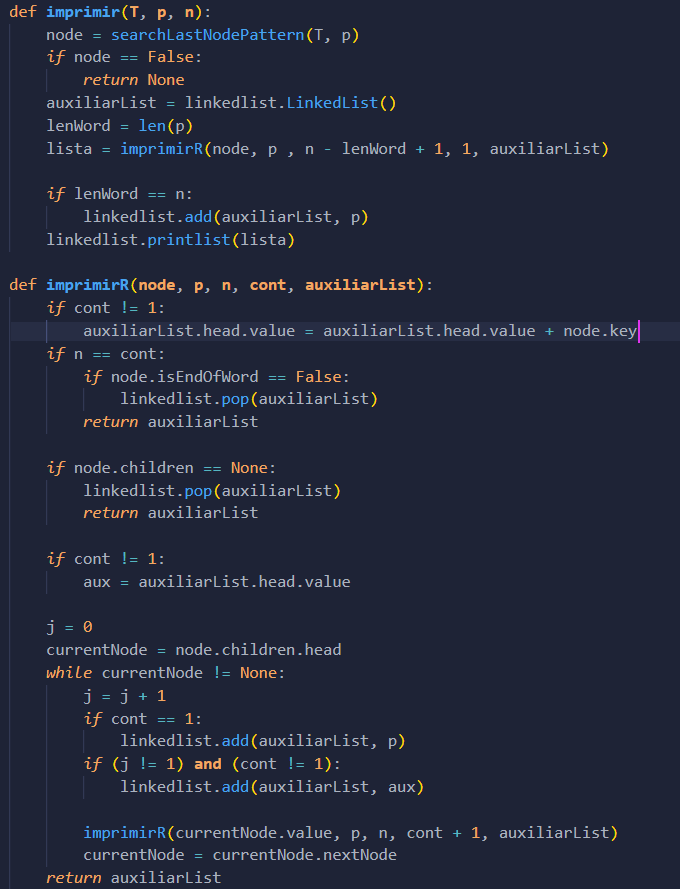
**Punto 3)**





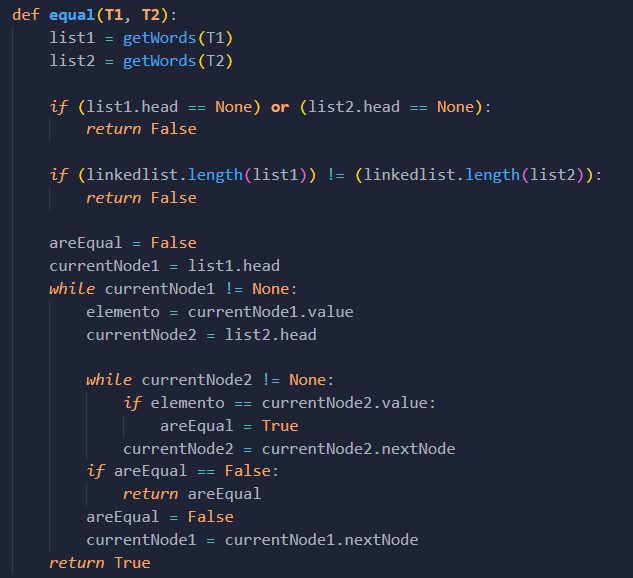
**Punto 4)**





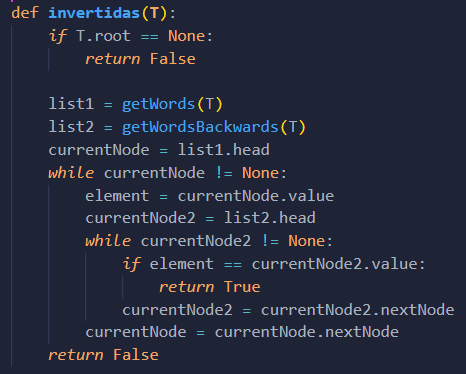
**Punto 5)**



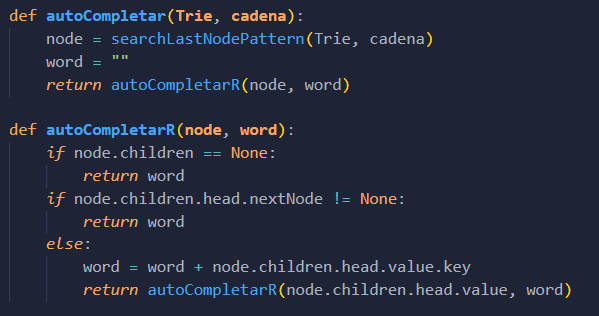


**Punto 6)**





**Punto 7)**



Link al código: <https://github.com/Kilxz/algoritmos2/tree/main/practicas/tp-trie/code>