**BNF:**

**Fortran:**

<https://techpubs.jurassic.nl/library/manuals/3000/007-3694-006/sgi_html/ch01.html>

**Python:**

<https://docs.python.org/3/reference/grammar.html>

**Historias:**

**Fortran:**

Fortran es un lenguaje de alto nivel utilizado en matemáticas y áreas científicas. Sus orígenes se remontan a finales de 1953, cuando IBM buscaba una alternativa más práctica al lenguaje ensamblador para programar su computadora central IBM 704. En 1956, se publicó un manual dirigido al público porque los clientes dudaban en usar un lenguaje de alto nivel a menos que su compilador pudiera generar código con un rendimiento comparable al del código ensamblador hecho a mano. A pesar del escepticismo inicial, Fortran redujo en veinte veces el número de sentencias necesarias para operar una máquina, ganando rápidamente aceptación. Fortran fue ampliamente adoptado por científicos para escribir programas numéricamente intensivos, lo que incentivó a los desarrolladores a crear compiladores más rápidos y eficientes. La inclusión de tipos de datos y aritmética de números complejos amplió su rango de aplicaciones, haciéndolo especialmente adecuado para ingeniería eléctrica y otras aplicaciones técnicas. Hacia 1960, había versiones de Fortran disponibles para las computadoras IBM 709, 650, 1620 y 7090. La creciente popularidad de Fortran impulsó a otros fabricantes de computadoras a ofrecer compiladores Fortran para sus máquinas. Para 1963, existían más de 40 compiladores Fortran. Debido a esto, Fortran es considerado el primer lenguaje de programación ampliamente utilizado y soportado en diversas arquitecturas de computadoras. Su desarrollo estuvo estrechamente vinculado con la evolución temprana de la tecnología de compiladores, y muchos avances en teoría y diseño de compiladores fueron motivados por la necesidad de generar código eficiente para programas en Fortran.

**Python:**

Python es un lenguaje de programación de alto nivel, interpretado, cuya filosofía destaca por la legibilidad de su código. Es un lenguaje multiparadigma, ya que admite la programación orientada a objetos, la programación imperativa y, en menor medida, la programación funcional. Además, es interpretado, dinámico y multiplataforma. Python fue creado en 1989 en los Países Bajos por Guido van Rossum como sucesor del lenguaje ABC. ABC es un lenguaje que comparte algunas características clave con Python, como la no necesidad de declarar variables. Sin embargo, a Van Rossum no le gustaba la falta de extensibilidad de ABC, lo que le causaba problemas al interactuar con el sistema operativo Amoeba, con el que trabajaba a diario. Esto lo llevó a crear su propio lenguaje, modificando esta característica como idea principal. El nombre Python proviene de la afición de su creador por los humoristas británicos Monty Python.

**Comparación de Funciones de Ordenamiento (Sort):**

**Fortran:**

En Fortran, las funciones de ordenamiento no están incluidas en la biblioteca estándar, por lo que generalmente se implementan manualmente o se usan bibliotecas externas.

Ejemplo de ordenamiento de burbuja en Fortran:

|  |
| --- |
| program ordenamiento\_de\_burbuja  implicit none  integer, parameter :: cantidad = 10  integer :: i, j, aux  integer :: vector(cantidad) = (/ 10, 2, 8, 6, 7, 5, 4, 3, 1, 9 /)  do i = 1, cantidad -1  do j = 1, cantidad -i  if (vector(j) > vector(j+1)) then  aux = vector (j)  vector(j) = vector (j+1)  vector(j+1) = aux  end if  end do  end do  print \*, 'El vector ordenado:'  print \*, vector  end program ordenamiento\_de\_burbuja |

**Python:**

Python incluye funciones de ordenamiento muy eficientes en su biblioteca estándar, como sorted() y el método sort() para listas , que usan el algoritmo Timsort:

Ejemplo de ordenamiento con sorted() en Python:

|  |
| --- |
| vector = [10, 2, 8, 6, 7, 5, 4, 3, 1, 9]  vector\_ordenado = sorted(vector)  print("'El vector ordenado:")  print(vector\_ordenado) |

Timsort es un algoritmo de ordenamiento híbrido que combina técnicas de ordenamiento por inserción y por mezcla (merge sort). Primero, toma el array original y lo divide en una cierta cantidad de subarrays llamados runs. Estos runs son secuencias ya ordenadas de manera ascendente o descendente, ya que las descendentes se pueden reordenar fácilmente invirtiendo los valores. Estas secuencias se conocen como runs naturales. Además, se define un tamaño mínimo de run que, en la práctica, es una constante fija (por ejemplo, 32 elementos).

Primero, se identifican las runs naturales y luego se dividen o fusionan para cumplir con el tamaño mínimo, asegurando que se mantenga lo más cercanas posible al orden ascendente o descendente. Luego, se ordenan los runs: se comparan dos elementos adyacentes y, si el primero es más pequeño que el segundo, se sigue con el segundo comparado con el tercero. Si el tercero es más pequeño que el segundo, se invierten sus posiciones y se compara el tercero con el primero. Si están ordenados, se dejan; si no, se invierten. Este proceso continúa desde la ubicación de la modificación osea se sigue comparando el segundo con el cuarto. Este proceso continúa hasta tener todo el run ordenado.

Una vez ordenados todos los runs, se comparan los primeros elementos de cada run, y el más pequeño se coloca como el primer elemento de la lista final ordenada. Luego, se vuelve a comparar los primeros elementos restantes junto con el segundo de la run de donde se tomó el primero, y así sucesivamente hasta terminar con todas las runs o que quede solo una, la cual se coloca al final de la lista.

Otra característica clave de Timsort es que, en caso de encontrar una run que esté total o parcialmente ordenada, utiliza métodos que consumen menos recursos. Para ello, primero analiza los primeros elementos y, si no hay cambio de orden, toma un valor promedio y analiza si los siguientes son más grandes, asegurando que no hay un valor menor más adelante en la lista.

**Comparación de Rendimiento**

Para comparar el rendimiento de las funciones de ordenamiento en Fortran y Python, se realizó una medición del tiempo que tarda en ordenar cada programa una lista con 100000 elementos aletorios.

Facilidad de Uso: Python proporciona funciones de ordenamiento listas para usar y muy eficientes, mientras que en Fortran es común tener que implementar el algoritmo manualmente o depender de bibliotecas externas.

Rendimiento: Aunque Python es más fácil de usar y tiene un código más limpio, Fortran generalmente puede ser más rápido para tareas computacionalmente intensivas debido a su optimización para cálculos numérico.