**Método de la burbuja**

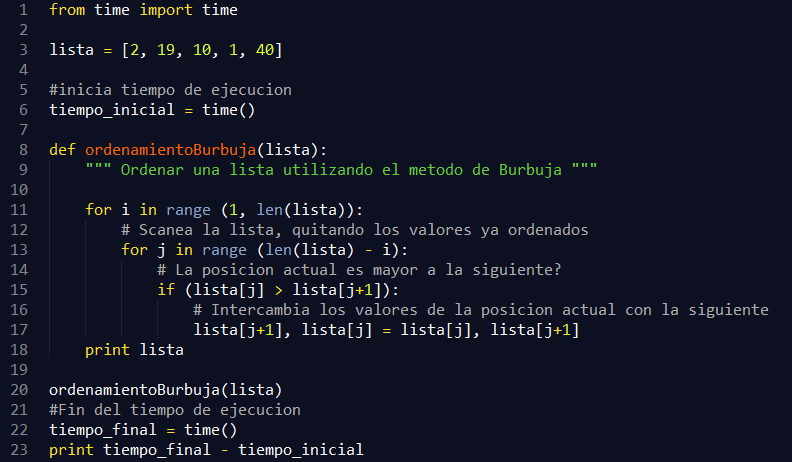
Este método consiste en acomodar el vector moviendo el mayor hasta la última casilla comenzando desde la casilla cero del vector hasta haber acomodado el número más grande en la última posición, una vez acomodado el más grande, prosigue a encontrar y acomodar el siguiente más grande comparando de nuevo los números desde el inicio del vector, y así sigue hasta ordenar todo los elementos el arreglo.

**Algoritmo**

**Diagrama de flujo**

****

**Código en C (Inserta screenshot de tu código)**

**Codigo en Python**

**Tiempo de ejecución 1**

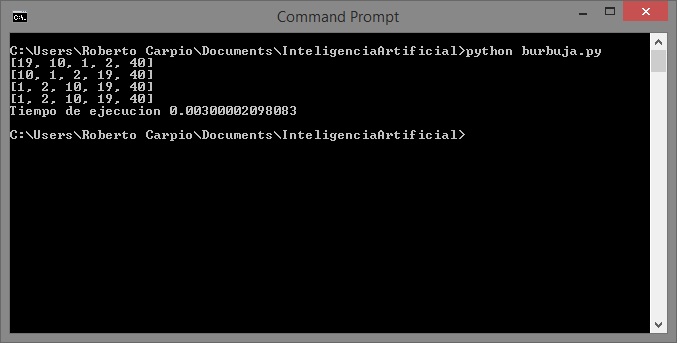
Procesador: Intel® Core™ i3-4030U CPU 1.90GHz 1.90Hz

Memoria Ram: 6.00GB

Tipo de Sistema: 64-bit, x64

Sistema Operativo: Windows 8.1

Tiempo de ejecución: 0.00300002098083



**Tiempo de ejecución 2 (Inserta el tiempo de ejecución en tu C en tu computadora)**

**Método de Inserción**

Para cada elemento de la lista después del primero, comparar los elementos con los anteriores desplazando una posición a la derecha a todos los elementos anteriores que cumplan con la comparación y luego colocar el elemento en la posición del último elemento anterior desplazado

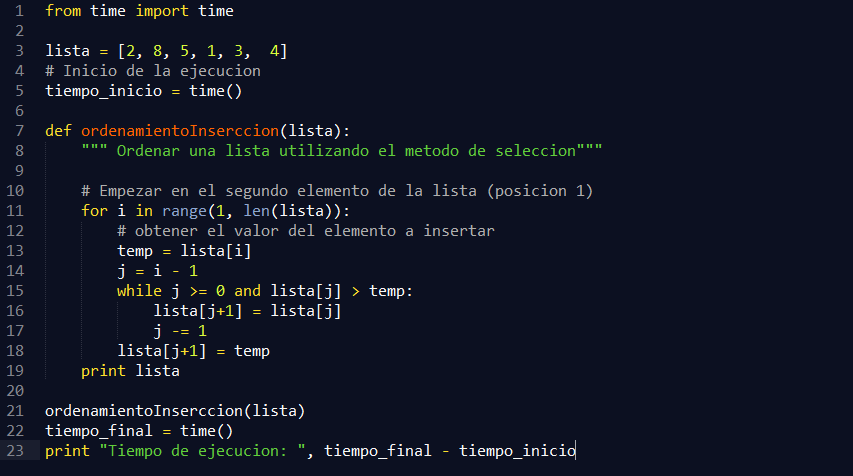
**Algoritmo**

***𝒔𝒊𝒂𝒓[u codigo)***

**Diagrama de flujo**

**Código en C (Inserta tu código)**

**Código en Python**



**Tiempo de ejecución 1**

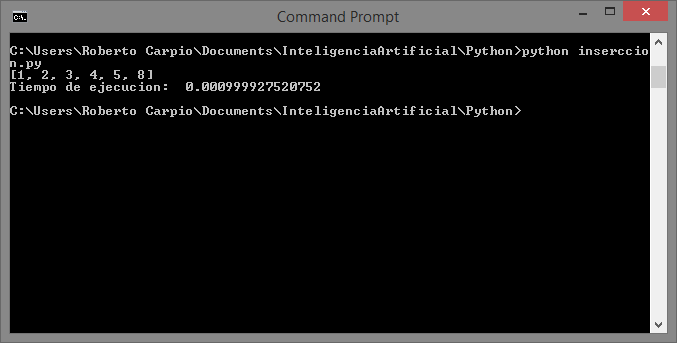
Procesador: Intel® Core™ i3-4030U CPU 1.90GHz 1.90Hz

Memoria Ram: 6.00GB

Tipo de Sistema: 64-bit, x64

Sistema Operativo: Windows 8.1

Tiempo de ejecución: 0.00999927520752



**Tiempo de ejecución 2 (Inserta el tiempo en C)**

**Método de selección**

El método de ordenación por selección consiste en encontrar el elemento más pequeño del arreglo y colocarlo en la primera posición, entre los restantes se busca el segundo elemento más pequeño y se coloca en la segunda posición y así sucesivamente hasta ordenar el arreglo.

**Algoritmo**

*-1*

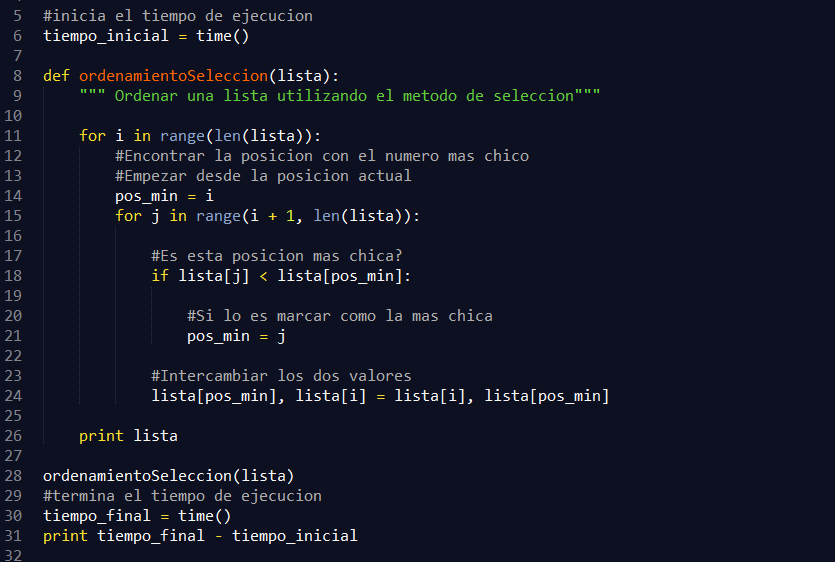
*mínimo ← j 𝒔𝒊𝒂𝒓[u codigo)*

**Diagrama de flujo**

****

**Código en C (Inserta tu código)**

**Código en Python**



**Tiempo de ejecución 1**

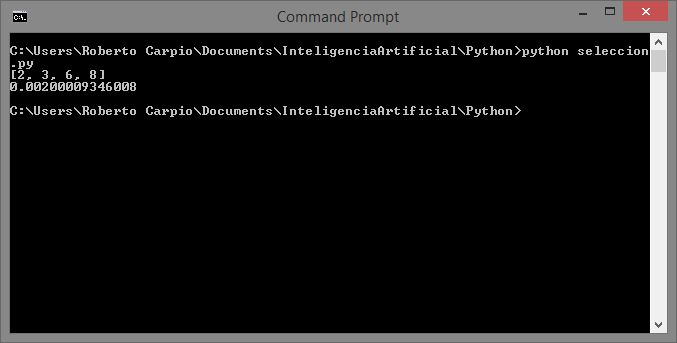
Procesador: Intel® Core™ i3-4030U CPU 1.90GHz 1.90Hz

Memoria Ram: 6.00GB

Tipo de Sistema: 64-bit, x64

Sistema Operativo: Windows 8.1

Tiempo de ejecución: 0.0020000936008

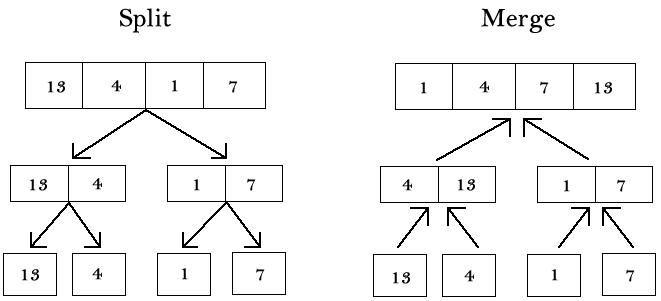


**Tiempo de ejecución 2 (Inserta el tiempo en C)**

**Método MergeSort**

Este algoritmo consiste básicamente en dividir en partes iguales la lista de números y luego mezclarlos comprándolos, y dejándolos ordenado. Si se piensa este algoritmo recursivamente, podemos ver que dividirá la lista hasta tener un elemento en cada lista, luego lo compara con el elemento de al lado y según su corresponda lo sitúa donde corresponde.

**Algoritmo**



Paso 1: Si es solamente un elemento de la lista ya está ordenado

Pasó 2: dividir la lista en partes iguales hasta que ya no pueda ser más dividida

Paso 3: Comparar las listas más pequeñas y unificarlas en una nueva con el orden ya establecido

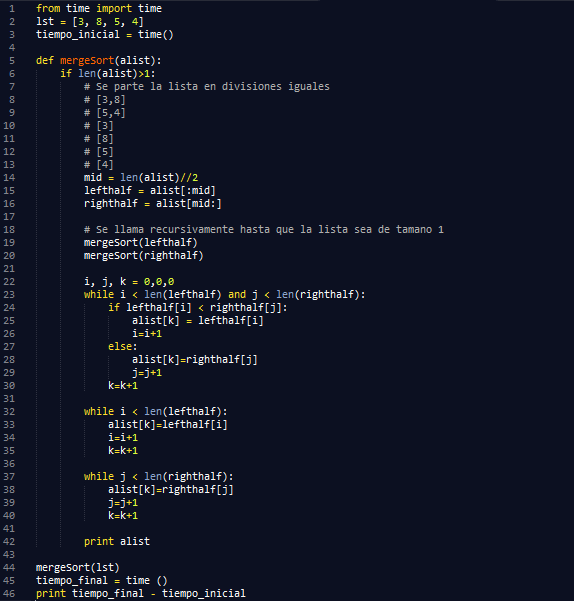
****

Una vez ordenadas las listas más pequeñas lo siguiente es comparar 4 > 1 si entonces 1 pasa a la posición 1 del vector, después compara 4 > 7 no entonces 4 para a la posición 2 del arreglo para finalizar compara 13 > 7 si entonces 7 ocupa la posición 13 del arreglo y por ultimo 13 queda en la posición final y nuestro arreglo ha sido ordenado.

**Diagrama de flujo (Hazlo perro)**

**Código en C (Inserta tu código)**

**Código en Python**



**Tiempo de ejecución 1**

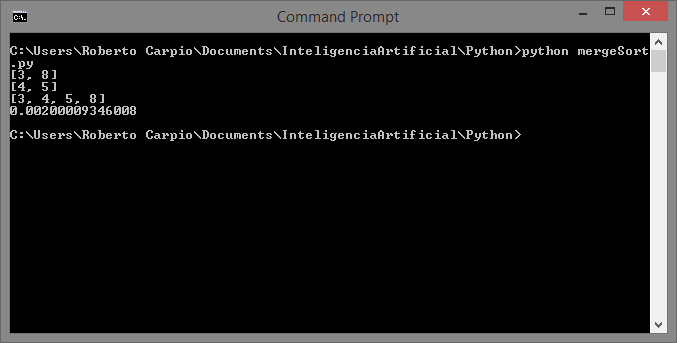
Procesador: Intel® Core™ i3-4030U CPU 1.90GHz 1.90Hz

Memoria Ram: 6.00GB

Tipo de Sistema: 64-bit, x64

Sistema Operativo: Windows 8.1

Tiempo de ejecución: 0.0020000346008



**Tiempo de ejecución 2 (Inserta el tiempo en C)**