Método de Ordenamiento de la Burbuja (BubbleSort)

(Análisis del código 1.)

Antes de entender de entender cualquier código que use este método debemos definir de que se trata el **Ordenamiento de la Burbuja** en Python.

El *ordenamiento de burbuja* es un algoritmo que te permite ordenar valores de un arreglo. Funciona revisando cada elemento con su adyacente. Si ambos elementos no están ordenados, se procede a intercambiarlos, si por el contrario los elementos ya estaban ordenados se dejan tal como estaban. Este proceso sigue para cada elemento del arreglo hasta que quede completamente ordenado.

Mecanismo del proceso:

Supongamos que vamos a ordenar la lista “a” de longitud n=5:

```
a = [8, 3, 1, 19, 14]
```

El algoritmo burbuja se compone de 2 bucles, uno dentro del otro. Llamaremos “bucle indentado” al que se encuentra dentro del otro bucle, es decir del “bucle principal”.

El “bucle principal” realizará el número de iteraciones necesarias para ordenar la lista (las iteraciones necesarias son n-1 veces) y el “bucle indentado” se encargará de comparar cada elemento con su adyacente y ordenarlos.

Si deseamos ordenar la lista “a”, el principal y el indentado comenzarán recorriendo n-1 veces (es decir, 4 veces), tomando en cuenta, que cuando el principal realice una iteración, el número de iteraciones del indentado se irá reduciendo: comienza con n-1-0 iteraciones, luego n-1-1 iteraciones, luego n-1-2 iteraciones, y así sucesivamente, hasta que termine el ordenamiento.

Ahora bien, una vez entendido el concepto y el funcionamiento del método de ordenamiento de la burbuja, es necesario escribir nuevas definiciones de ciertas funciones que se usan en el código planteado, para un mejor entendimiento del mismo.

- **Módulo random:** el módulo random de la librería estándar de Python incluye un conjunto de funciones que permiten obtener de distintos modos números aleatorios o, para ser rigurosos, pseudoaleatorios.

- **Sample:** es una función incorporada del módulo ocasional en Python que devuelve una lista de longitud particular de elementos elegidos de la secuencia, es decir, lista, tupla, cadena o conjunto. Se utiliza para mostrar esporádicamente sin reemplazo.

- **Def**: las funciones en Python se crean usando la palabra clave def, seguida de un nombre de función y parámetros de función entre paréntesis.

- **Operador de asignación += :** es un operador de asignación, y el resultado se almacena en la variable a, por ejemplo: *a + = 10*. Es lo mismo que a = a + 10.

Conociendo estas definiciones importantes para entender el código, procedemos a analizarlo. El algoritmo planteado es el siguiente:

```
from random import sample
# Importamos un Método de la biblioteca random para generar listas
aleatorias

lista = list(range(100)) # Creamos la lista base con números del 1 al
100

# Creamos una lista aleatoria con sample
#(8 elementos aleatorios de la lista base)
vectorbs = sample(lista,8)


def bubblesort(vectorbs):
```

```
    """Esta función ordenara el vector que le pases como argumento con
el Método de Bubble Sort"""

    # Imprimimos la lista obtenida al principio (Desordenada)
    print("El vector a ordenar es:",vectorbs)
    n = 0 # Establecemos un contador del largo del vector

    for _ in vectorbs:
        n += 1 #Contamos la cantidad de caracteres dentro del vector

    for i in range(n-1):
    # Le damos un rango n para que complete el proceso.
        for j in range(0, n-i-1):
            # Revisa la matriz de 0 hasta n-i-1
            if vectorbs[j] > vectorbs[j+1] :
                vectorbs[j], vectorbs[j+1] = vectorbs[j+1],
vectorbs[j]
            # Se intercambian si el elemento encontrado es mayor
            # Luego pasa al siguiente
    print ("El vector ordenado es: ",vectorbs)

bubblesort(vectorbs)
```

Análisis del código:

1. En principio se le indica a Python mediante el comando *`from random import sample`* que se quiere importar un método de la biblioteca *random* para generar listas aleatorias.

2. Se crea una variable llamada “lista” para almacenar en ella la lista que se ordenará más adelante. Esta lista se crea mediante la función *list(),* definida en un rango que va del 0 al 100, con la función *range(100)*.

3. Seguido de esto, se hace uso de la función *sample().* Se crea una nueva variable llamada: `vectorbs=sample(lista,8).` Básicamente, aquí le pedimos a python que, a partir de la lista anterior, definida como “lista”, cree otra con números aleatorios y con una cantidad de 8 términos.

4. Mediante `def bubblesort(vectorbs):` le indicamos a Python que debe ordenar el vector que se le pase como argumento con el método bubblesort.

5. Antes de seguir con el bucle for, se establece un contador que determinará el largo del vector, de tal manera: n=0.

6. Para saber la cantidad de términos que posee el vector se hace uso de un primer blucle for:

```
for _ in vectorbs:
      n += 1
```

7. Posteriormente, se emplea un segundo blucle for. Aquí es donde entra el proceso de ordenamiento para la lista. Como se explicó al principio, este bucle lo que hará es realizar el número de iteraciones necesarias para ordenar la lista (las iteraciones necesarias son n-1 veces).

8. Mediante el segundo for indentado, se harán las iteraciones necesarias en el rango de (0, n-i-1). Aquí inicia la comparación de cada valor de la lista, en donde, si el número adyacente es menor, se intercambian los elementos. Luego pasa al siguiente y así sucesivamente hasta que termine el ciclo de ordenamiento.