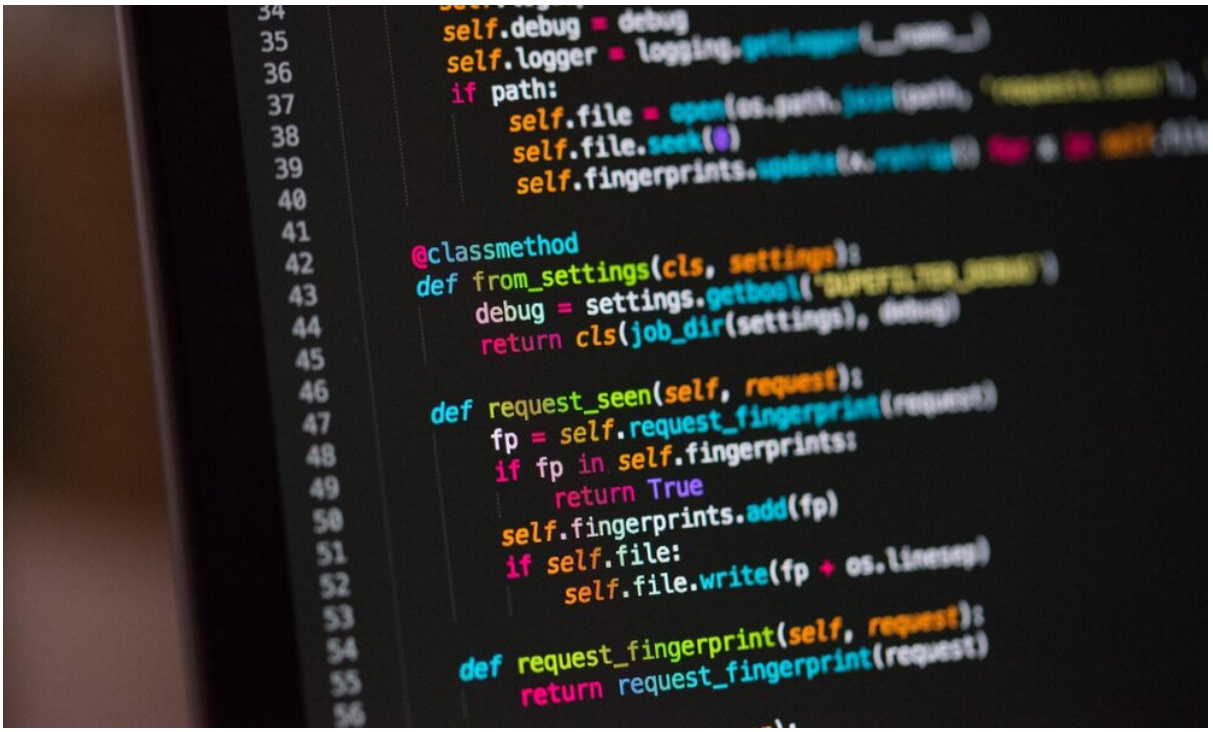
HITO GRUPAL PROGRAMACIÓN

Lara López-Pozuelo Cebadera
Javier Ahijado Luna
Chedey Gallego Gabaldon

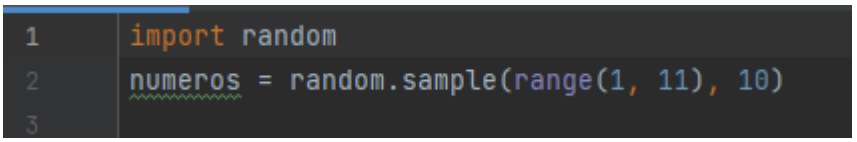


CÓDIGO (LARA)

- **En Pseint o en Python (Jupyter, PyCharm, VSCode) y haciendo uso de funciones, para cada uno de los algoritmos de ordenación asignados, se tiene que:**

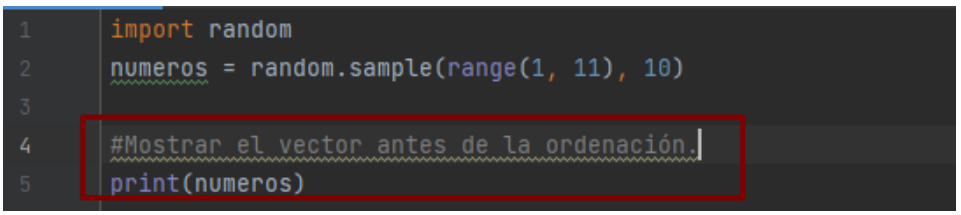
  Para realizar el código he decidido hacerlo en Pycharm

1. **Generar de forma aleatoria un vector de 10 elementos enteros, los números aleatorios serán del 1 al 10 y se tiene que controlar que no se repitan.**



   Para ello he tenido que importar la librería random, lo que hace es sacar números aleatorios dentro de un rango
   Para que no se repitan he utilizado random.sample cuya función es que no se repitan los números aleatorios dentro del rango
   - (1, 11) significa que el rango de los números aleatorios, se comprenden entre el 1 y el 11
   - , 10) significa el número de números que van a salir aleatoriamente del rango

2. **Mostrar el vector antes de la ordenación.**



   Para este paso solo habría que poner que se muestren los números aleatorios asignados a la variable “números”, con un “print”

3. **Aplicar el algoritmo de ordenación: se debe mostrar información por pantalla para indicar los distintos pasos que realiza el algoritmo .**

```
1   import random
2   numeros = random.sample(range(1, 11), 10)
3
4   #Mostrar el vector antes de la ordenación.
5   print(numeros)
6
7   #Aplicar el algoritmo de ordenación
8   numeros.sort()
9   print(numeros)
```

Para ordenar los número de menor a mayor he utilizado el nombre de la variable más ".sort" que lo que hace es ordenarlos . Y luego les dices que se muestren.

4. **Mostrar el vector después de la ordenación.**

```
[1, 10, 3, 2, 8, 4, 5, 7, 9, 6]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Process finished with exit code 0
```

- **Buscar información sobre la búsqueda dicotómica o binaria, explicarlo con un ejemplo e implementarlo (se puede implementar la versión recursiva o iterativa) en PSeint. Analiza su eficiencia comparándolo si hacemos la búsqueda sin que el vector esté ordenado. (JAVI)**

Búsqueda Dicotómica o Binaria:

Se utiliza cuando el vector en el que se determina la existencia de un elemento está ordenado. Este tipo de algoritmo reduce el tiempo de búsqueda, ya que disminuye el número de iteraciones necesarias.

- **Recursiva**: Es cuando utilizamos un método que se llama a sí mismo.
- **Iteración:** Es cuando permiten repetir una sentencia o conjunto de ellas.

- El código con recursividad es más consistente que el código con iteración. Solo hay una sentencia en el método de recursividad, y existe más de una sentencia en el método con iteraciones.

EJEMPLO:

hito.psc

```
1  Funcion saludar(a)
2      escribir a;
3      si a>0 Entonces
4          saludar(a-1)
5      FinSi
6  FinFuncion
7  Algoritmo recursividad
8      definir num Como Entero;
9      num=0;
10     Escribir "Dime un número";
11     Leer num;
12     saludar(num);
13
14 FinAlgoritmo
15
```

Lista de Variables | Operadores y Funciones

PSeInt - Ejecutando proceso RECURSIVIDAD

```
*** Ejecución Iniciada. ***
Dime un número
> 4
4
3
2
1
0
*** Ejecución Finalizada. ***
```

- **Cada grupo tiene asignados los siguientes algoritmos de ordenación: (CHEDEY)**

  MODELO A

  - Ordenación de burbuja (Bubble Sort) o intercambio directo: La ordenación en burbuja es un sencillo algoritmo de ordenamiento. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. El proceso se repite hasta que la estructura esté ordenada. Se irá comparando desde la casilla 0 número tras número hasta encontrar uno mayor. Si este es realmente el mayor de todo el vector, se llevará hasta la última casilla. Si no es así, será reemplazado por uno mayor que él.

  - Ordenación por inserción: El algoritmo de ordenamiento por inserción es un algoritmo de fácil aplicación que permite el ordenamiento de una lista. Consiste en ir insertando un elemento de la lista asumiendo que el primer elemento es la parte ordenada. El algoritmo irá comparando un elemento de la parte desordenada de la lista con los elementos de la parte ordenada, insertando el elemento en la posición correcta dentro de la parte ordenada, y así sucesivamente hasta obtener la lista ordenada.

  MODELO B

  - Ordenación por selección: Este algoritmo mejora ligeramente el algoritmo de la burbuja. Consiste en encontrar el menor de todos los elementos del arreglo e intercambiarlo con el que está en la primera posición. Luego el segundo más pequeño, y así sucesivamente hasta ordenarlo todo.

  - Ordenación Shell o por incremento decreciente: Consiste en dividir el arreglo en bloques de varios elementos para organizarlos después. El proceso se repite, pero con intervalos cada vez más pequeños, de tal manera que al final, el ordenamiento se haga en un intervalo de una sola posición. Es más utilizado para listas grandes y suele ser más rápido.

WEBGRAFÍA:

- https://cdklhph.wordpress.com/2015/08/10/ordenamiento-insercion/
- https://uniwebsidad.com/libros/algoritmos-python/capitulo-19/ordenamiento-por-insercion
- https://pythondiario.com/2017/09/ordenamiento-de-burbuja-algoritmos-de.html
- http://lwh.free.fr/pages/algo/tri/tri_selection_es.html#:~:text=Consiste%20en%20encontrar%20el%20menor,as%C3%AD%20sucesivamente%20hasta%20ordenarlo%20too
- http://132.248.48.64/repositorio/moodle/pluginfile.php/1472/mod_resource/content/1/contenido/index.html .