#### Actividad 10 - Clases // Sort

Michel Dávalos Boites // Alan Gabriel Bautista Chavira

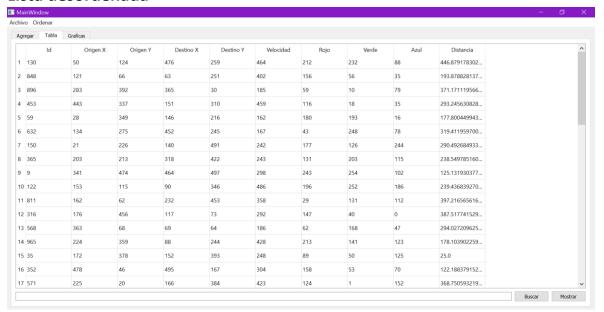
Programación // Seminario de solución de problemas de algoritmia

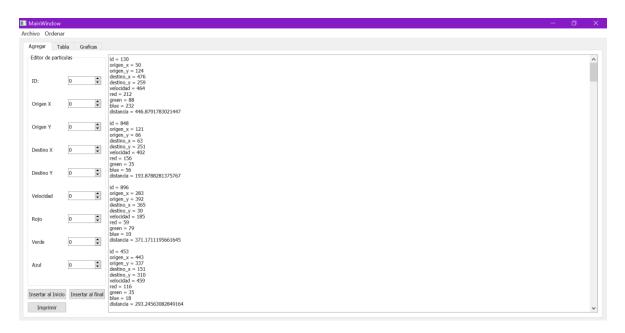
#### Lineamientos de evaluación:

- El reporte esta en formato Docs o PDF
- El reporte sigue las pautas del formato de actividades
- El reporte tiene desarrolladas todas las pautas del formato de actividades
- Capturas del ordenamiento ascendente y descendente por ID en el QTable y QPlainText
- Capturas del ordenamiento ascendente y descendente por Velocidad en el QTable y QPlainText
- Capturas del ordenamiento ascendente y descendente por Distancia en el QTable y QPlainText

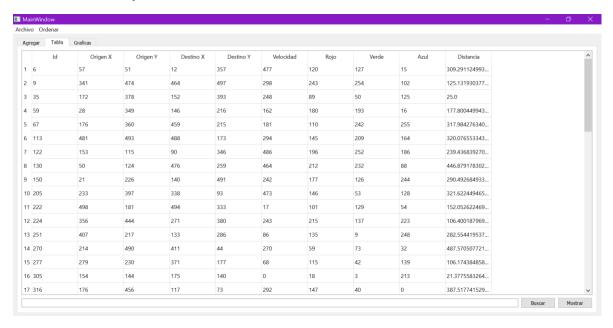
### Desarrollo

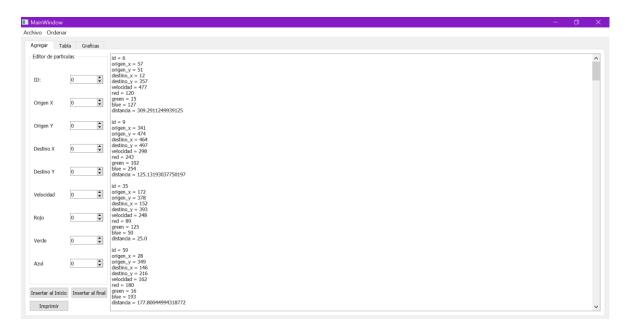
## Lista desordenada



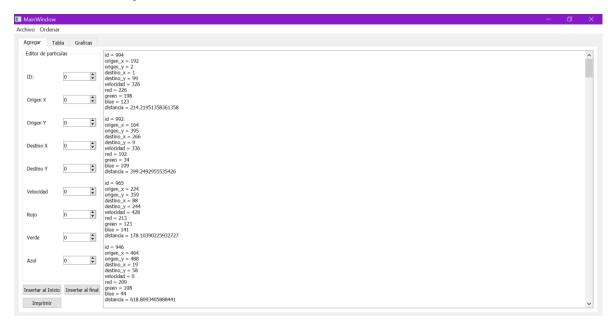


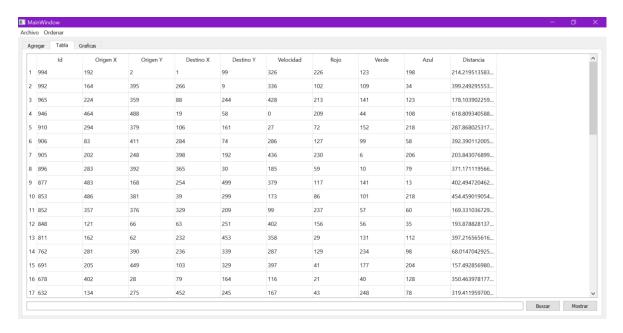
# Ordenamiento por ID ascendente



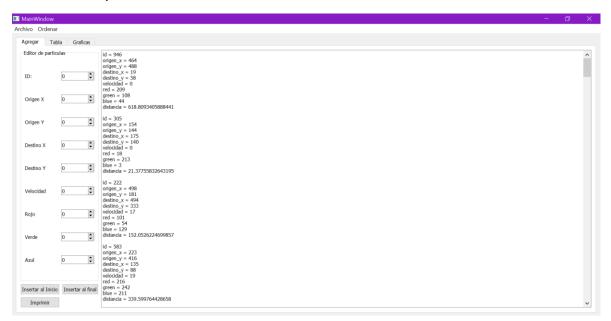


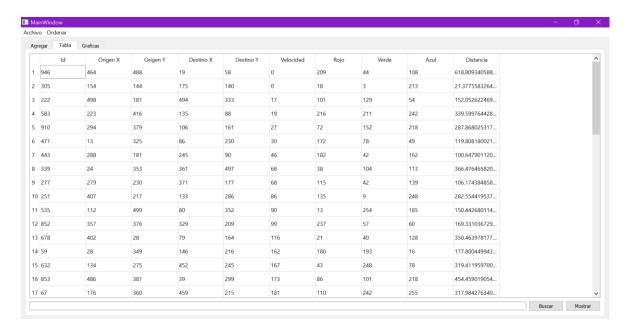
# Ordenamiento por ID descendente



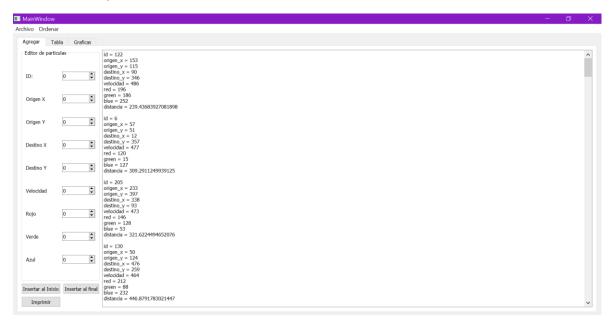


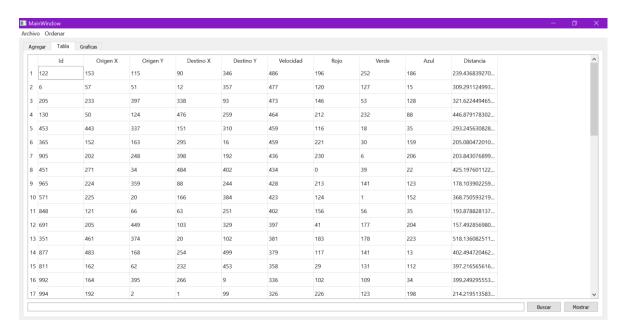
## Ordenamiento por Velocidad ascendente



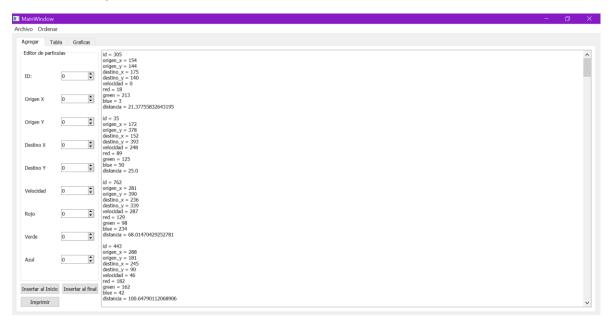


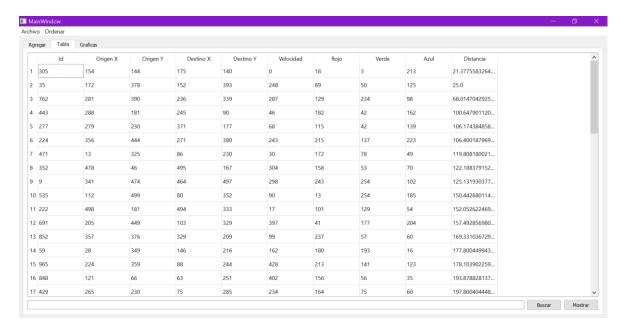
# Ordenamiento por Velocidad descendente



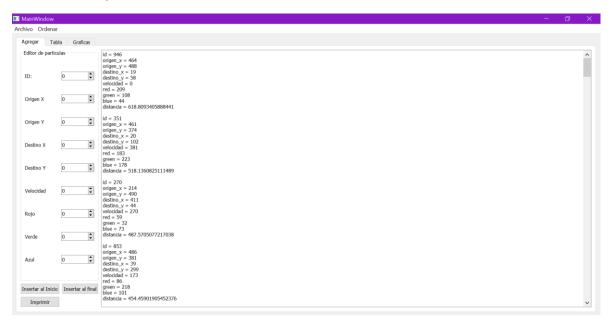


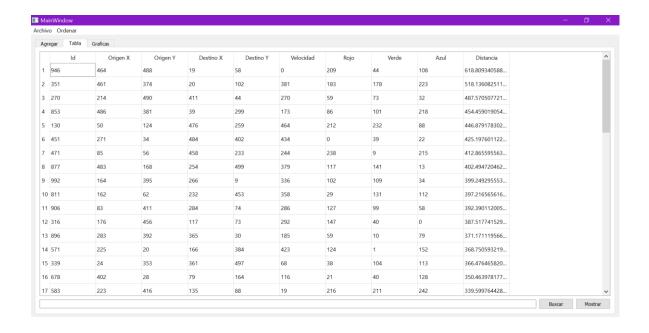
# Ordenamiento por Distancia ascendente





# Ordenamiento por Distancia descendente





#### **Conclusiones**

La función sort de Python cumple de una manera extremadamente sencilla y práctica para el programador, algo que en lenguajes de bajo nivel puede ser un proceso engorroso.

Solo tuvimos que especificarle a la clase Particula cuál sería el atributo que tomaríamos por defecto para que la función sort pudiera comparar las distintas partículas de la lista, así como métodos "getter" para obtener el miembro velocidad y distancia según nos convenga.

Creo que me tardé más en poner los botones o "actions" y vinculando esas acciones a funciones específicas del mainwindow.py, que implementado los ordenamientos en sí.

### Referencias

https://www.youtube.com/watch?v=0NZajLly5qQ&t=622s&ab channel=MICHELDAVALOSBOITES