# Actividad 05 – Clases y objetos

## Roberto Haro González

Seminario de solución de problemas de algoritmia

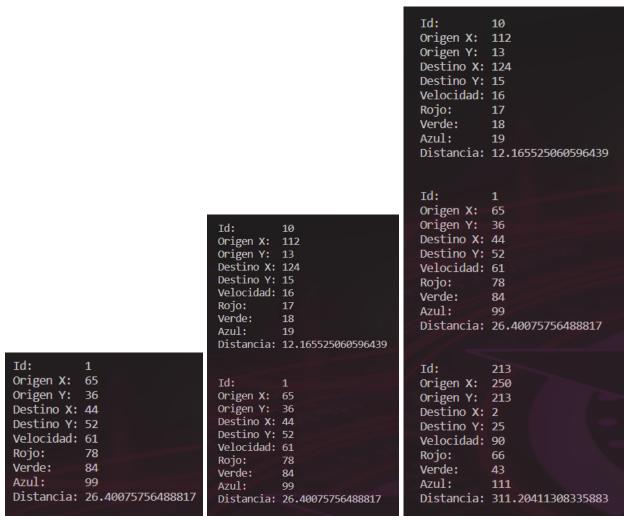
#### Lineamientos de evaluación

- ☑ El reporte está en formato Google Docs o PDF.
- ☑ El reporte sigue las pautas del Formato de Actividades.
- ☑ Se muestra la captura de pantalla de los datos antes de usar el método agregar\_inicio() y la captura de pantalla del método mostrar() después de haber utilizado el método agregar\_inicio().

☑Se muestra la captura de pantalla de los datos antes de usar el método agregar\_final() y la captura de pantalla del método mostrar() después de haber utilizado el método agregar final().Desarrollo

#### Desarrollo

Usando de referencia la clase partícula de la actividad pasada agregue el calculo de la distancia euclidiana retornando directamente el resultado de la operación. Para imprimir se hace uso del método str para la lista y las propias partículas.



## **Conclusiones**

Si bien por error entregué tarde en classroom por no presionar el botón de entregado y no darme cuenta, haber hecho la practica 4 me sirvió mucho ya que en ella ya había trabajado los métodos de impresión para la lista, pude aprovechar lo que aprendí allá y sumarle el calculo de la distancia.

## Referencias

[MICHEL DAVALOS BOITES]. (2020b, octubre 8). PySide2 - Introducción (Qt for Python)(II).

YouTube. Recuperado 10 de octubre de 2022, de

1ZvgVgnoYXeZo49Boz6CDGTf&index=5

## Código

#### Algoritmos

```
import math

def distancia_euclidiana(x_1, y_1, x_2, y_2):
    return math.sqrt(math.pow((x_1-x_2),2)+math.pow((y_2-y_1),2))
```

#### Clase partícula

```
from algoritmos import distancia_euclidiana
class Particula:
   def __init__(self, id=0, origen_x=0<=500, origen_y=0<=500, destino_x=0<=500,
destino_y=0<=500, velocidad=0, red=0<=255, green=0<=255, blue=0<=255):
        self.__id = id
       self.__origen_x = origen_x
       self.__origen_y = origen_y
       self.__destino_x = destino_x
       self.__destino_y = destino_y
       self.__velocidad = velocidad
       self.__red = red
       self.__green = green
       self.__blue = blue
       self.__distancia =
distancia_euclidiana(origen_x,origen_y,destino_x,destino_y)
   def __str__(self):
        return(
                        ' + str(self.__id) + '\n' +
            '\nId:
            'Origen X: ' + str(self.__origen_x) + '\n' +
            'Origen Y: ' + str(self.__origen_y) + '\n' +
            'Destino X: ' + str(self. destino x) + '\n' +
```

```
'Destino Y: ' + str(self.__destino_y) + '\n' +
   'Velocidad: ' + str(self.__velocidad) + '\n' +
   'Rojo: ' + str(self.__red) + '\n' +
   'Verde: ' + str(self.__green) + '\n' +
   'Azul: ' + str(self.__blue) + '\n' +
   'Distancia: ' + str(self.__distancia) + '\n'
)
```

#### Clase lista de partículas

```
from particula import Particula
class ListaParticulas:
    def __init__(self):
        self.__particulas = []
    def agregar final(self, Particula:Particula):
        self.__particulas.append(Particula)
    def agregar_inicio(self, Particula:Particula):
        self.__particulas.insert(0,Particula)
    def mostrar(self):
        for Particula in self.__particulas:
            print(Particula)
    def __str__(self):
        return "".join(
            str(Particula) for Particula in self._particulas
p1 = Particula(1,65,36,44,52,61,78,84,99)
p2 = Particula(10,112,13,124,15,16,17,18,19)
p3 = Particula(213,250,213,2,25,90,66,43,111)
lista = ListaParticulas()
lista.agregar_inicio(p1)
lista.mostrar()
lista.agregar inicio(p2)
lista.mostrar()
lista.agregar_final(p3)
lista.mostrar()
```