Actividad 05 - Clases y Objetos

Arturo Sánchez Sánchez

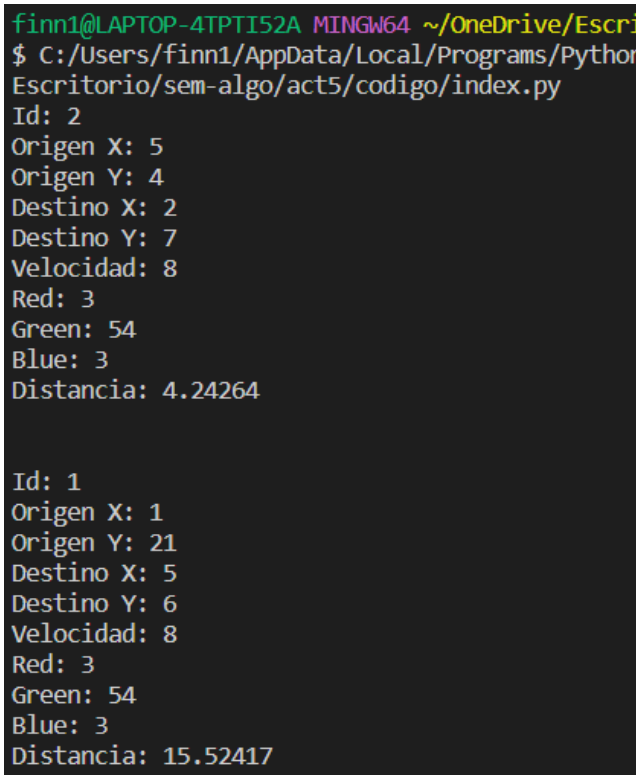
Seminario de Algoritmia

Lineamientos de evaluación

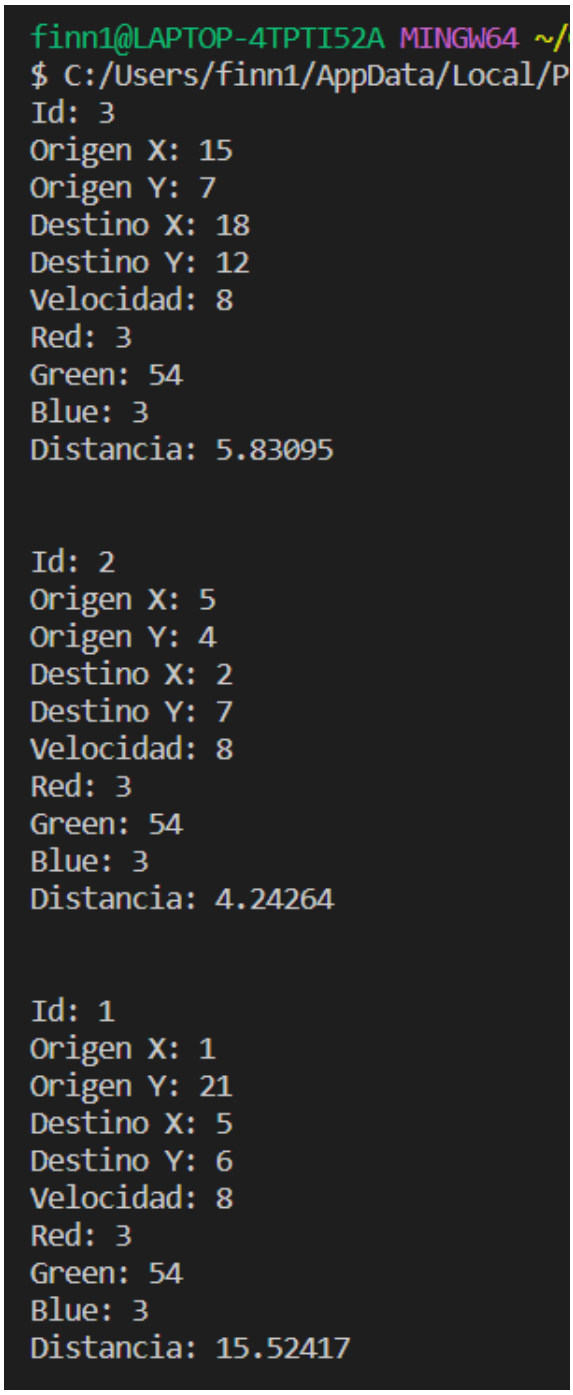
- El reporte está en formato Google Docs o PDF.
- El reporte sigue las pautas del Formato de Actividades .
- El reporte tiene desarrollada todas las pautas del Formato de Actividades.
- Se muestra la captura de pantalla de los datos antes de usar el método agregar_inicio() y la captura de pantalla del 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

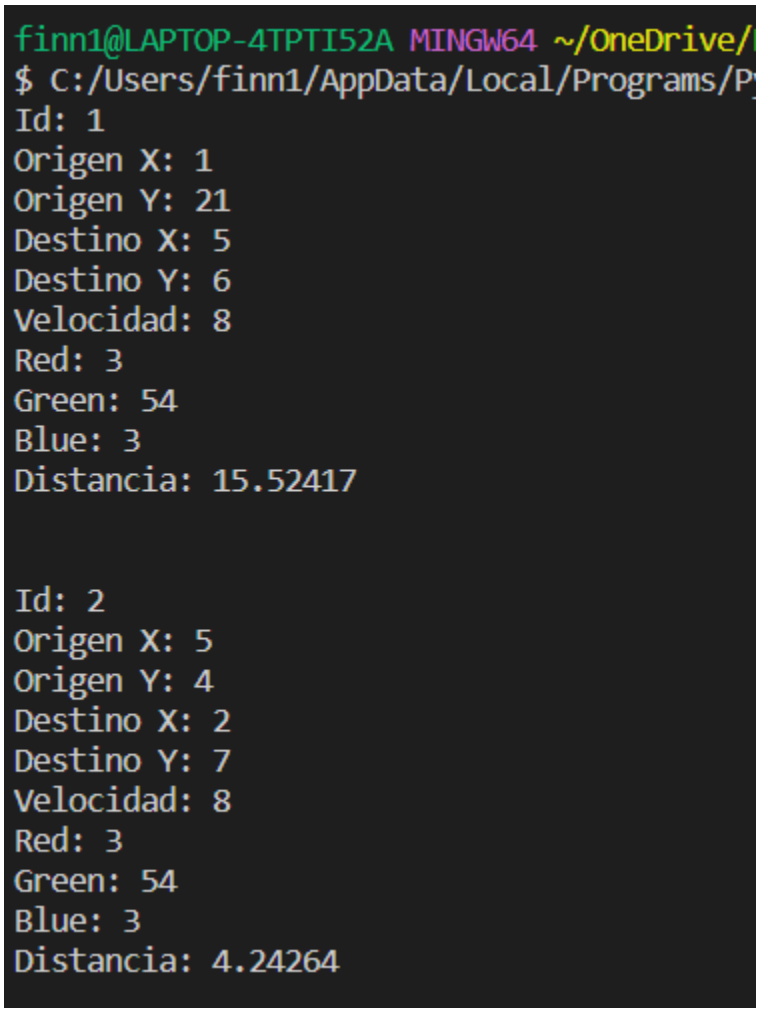
Primero creé la clase de partícula con sus atributos y los métodos requeridos, por consiguiente en otro archivo creé la función para calcular la distancia entre las coordenadas de una partícula. Una vez que una partícula podría ser impresa correctamente procedí a crear la lista doblemente ligada la cual recibe nodos con información que estaban conectados a otros. Por último creé los métodos agregar_final, agregar_inicio y mostrar.



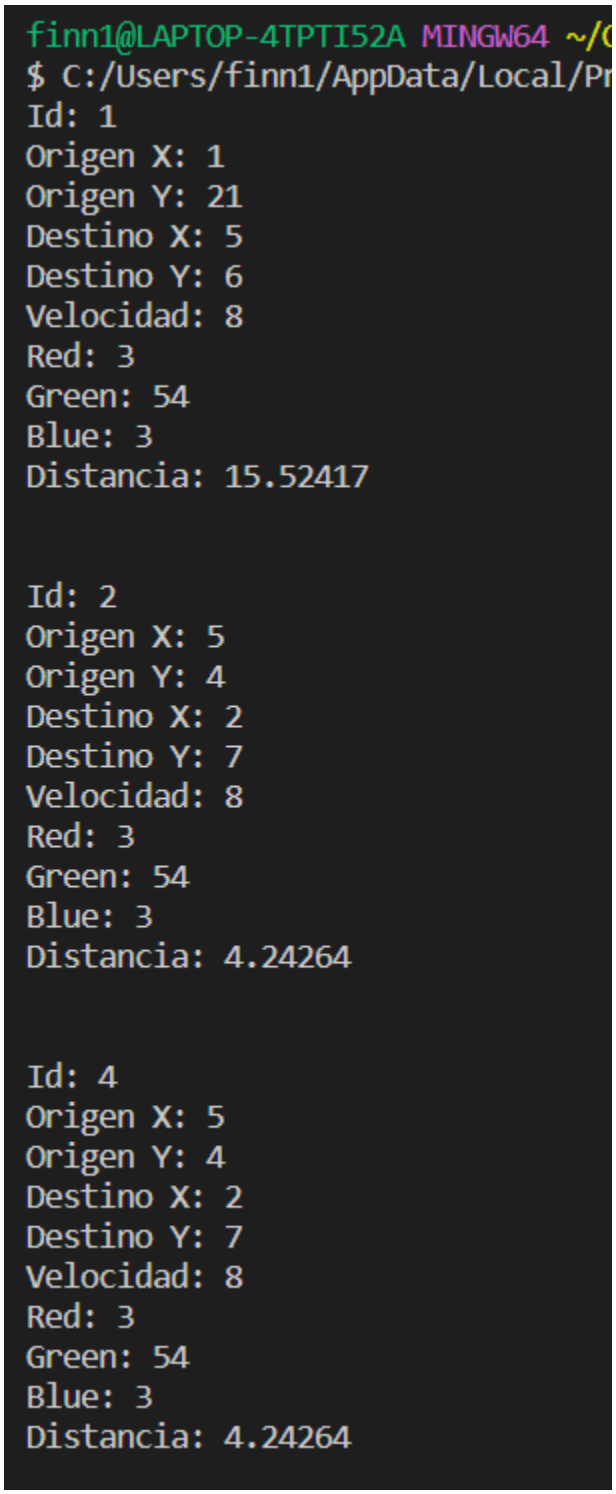
Captura de pantalla antes de usar el método agregar_inicio()



Captura de pantalla después de usar el método agregar_inicio()



Captura de pantalla antes de usar el método agregar_final()



Captura de pantalla despues de usar el método agregar_final()

Conclusiones

Una actividad muy interesante en su totalidad, me gustó el uso de clases y objetos y su manejo, creo que nos adentramos a un mundo de posibilidades una vez que sabemos usar las herramientas que nos da python para el uso de la programación orientada a objetos, espero que en un futuro podamos seguir trabajando con esto, creando cada vez más cosas teniendo como límite únicamente nuestra imaginación.

Referencias

undefined [MICHEL DAVALOS BOITES]. (2020, October 7). PySide2 - Introducción (Qt for Python)(I) [Video]. YouTube. Retrieved September 29, 2022, from https://www.youtube.com/watch?v=T0qJdF1fMqo&t=424s

Código

index.py

```
from particula import Particula



class Nodo():
    dato = None
    siguiente = None
    anterior = None

    def __init__(self, dato):
        self.dato = dato
        self.siguiente = None



class Lista_ligada():
    nodo_inicial = None
    nodo_final = None
    no_elements = 0

    def __init__(self):
        self.nodo_inicial = None
        self.nodo_final = None

    def agregar_inicio(self, nodo):
        if(self.no_elements == 0):
            self.nodo_inicial = nodo
```

```
            self.nodo_final = nodo
            self.no_elements = self.no_elements + 1
        else:
            temporal = self.nodo_inicial
            temporal.anterior = nodo
            nodo.siguiente = temporal
            self.nodo_inicial = nodo
            self.no_elements = self.no_elements + 1

    def agregar_final(self, nodo):
        if(self.no_elements == 0):
            self.nodo_inicial = nodo
            self.nodo_final = nodo
            self.no_elements = self.no_elements + 1
        else:
            temporal = self.nodo_final
            temporal.siguiente = nodo
            nodo.anterior = temporal
            self.nodo_final = nodo
            self.no_elements = self.no_elements+1

    def mostrar(self):
        temp = self.nodo_inicial
        while(temp):
            print(temp.dato)
            temp = temp.siguiente
```

```
lista_ligada = Lista_ligada()


particula1 = Particula(1, 1, 21, 5, 6, 8, 3, 54, 3)
particula2 = Particula(2, 5, 4, 2, 7, 8, 3, 54, 3)
particula3 = Particula(3, 15, 7, 18, 12, 8, 3, 54, 3)
particula4 = Particula(4, 5, 4, 2, 7, 8, 3, 54, 3)


nodo1 = Nodo(particula1)
nodo2 = Nodo(particula2)
nodo3 = Nodo(particula3)
nodo4 = Nodo(particula4)



lista_ligada.agregar_final(nodo1)
lista_ligada.agregar_final(nodo2)
lista_ligada.agregar_inicio(nodo3)
lista_ligada.agregar_final(nodo4)


lista_ligada.mostrar()
```

particula.py

```
from algoritmos import distancia_euclidiana



class Particula:
    def __init__(self, id, origen_x, origen_y, destino_x, destino_y,
velocidad, red, green, blue):
```

```
        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 f"Id: {self.id}\nOrigen X: {self.origen_x}\nOrigen Y:
{self.origen_y}\nDestino X: {self.destino_x}\nDestino Y:
{self.destino_y}\nVelocidad: {self.velocidad}\nRed: {self.red}\nGreen:
{self.green}\nBlue: {self.blue}\nDistancia: {self.distancia}\n\n"
```

algoritmos.py

```
import math


def distancia_euclidiana(x_1, y_1, x_2, y_2):
    return "{:.5f}".format(math.sqrt(math.pow((x_2-x_1),
2)+math.pow(y_2-y_1, 2)))
```