

Actividad 05 (Clases y Objetos)



Rafael Arturo Gutiérrez Cruz

Seminario de Solucion 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](#) .
- [] 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

agregar_inicio()

```
from particulas import Particula

class administrador:
    def __init__(self):
        self.__particles = []

    def agregar_final(self, particle:Particula):
        self.__particles.append(particle)

    def agregar_inicio(self, particle:Particula):
        self.__particles.insert(0,particle)

    def mostrar(self):
        for particle in self.__particles:
            print(particle)

    def __str__(self):
        return "".join(
            str(particle) for particle in self.__particles
        )

P01 = Particula(1,35,40,20,10,15,20,15,10)
P02 = Particula(2,84,115,35,3,15,20,15,10)
administrador = administrador()

# administrador.agregar_inicio(P02)
# administrador.agregar_final(P01)

# administrador.mostrar()
```

Captura de pantalla antes de agregar partículas.

```
particle_administrator.py > ...
11 self.__particles.append((0,particle))
12
13 def mostrar(self):
14     for particle in self.__particles:
15         print(particle)
16
17 def __str__(self):
18     return "".join(
19         str(particle) for particle in self.__particles
20     )
21
22
23 P01 = Particula(1,35,40,20,10,15,20,15,10)
24 P02 = Particula(2,84,115,35,3,15,20,15,10)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Origen_x:84
Origen_y:115
Destino_x:35
Destino_y:3
Velocidad:15
Red:20
Green:15
Blue:10
Distancia:12.68857754044952

cinso@TsuruTuneado MINGW64 ~/Desktop/Actividad 5
\$

Captura de pantalla despues del método agregar al inicio.

Teoricamente tambien seria la captura antes de agregar al final.

```
particle_administrator.py > ...
25 administrador = administrador()
26
27 administrador.agregar_inicio(P02)
28 administrador.agregar_final(P01)
29
30 administrador.mostrar()
31
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Distancia:6.708203932499369

cinso@TsuruTuneado MINGW64 ~/Desktop/Actividad 5
\$ C:/Users/cinos/AppData/Local/Programs/Python/Python310/python.exe ...
Id:2
Origen_x:84
Origen_y:115
Destino_x:35
Destino_y:3
Velocidad:15
Red:20
Green:15
Blue:10
Distancia:12.68857754044952

Id:1
Origen_x:35
Origen_y:40
Destino_x:20
Destino_y:10
Velocidad:15
Red:20
Green:15

Captura despues de agregar al final

Conclusiones

Siguiendo los videos de referencia no es difícil, pero tuve algunos problemas a la hora de calcular la distancia, pero eso fue porque puse los valores al revés.

Referencias

Boites, M. D. (2020, 8 octubre). *PySide2 - Clases y Objetos (Qt for Python)(II)*

[Vídeo]. YouTube. Recuperado 15 de octubre de 2022, de <https://www.youtube.com/watch?v=KfQDtrrL2OU&feature=youtu.be>

Código

particle_administrator.py

```
from particulas import Particula

class administrador:
    def __init__(self):
        self.__particles = []

    def agregar_final(self, particle:Particula):
        self.__particles.append(particle)

    def agregar_inicio(self, particle:Particula):
        self.__particles.insert(0,particle)

    def mostrar(self):
        for particle in self.__particles:
            print(particle)

    def __str__(self):
        return "".join(
            str(particle) for particle in self.__particles
```

```

    )

P01 = Particula(1,35,40,20,10,15,20,15,10)
P02 = Particula(2,84,115,35,3,15,20,15,10)
administrador = administrador()

administrador.agregar_inicio(P02)
administrador.agregar_final(P01)

administrador.mostrar()

```

particulas.py

```

from algoritmos import distancia_euclidiana

class Particula:

    def __init__(self, id=0, origen_x=0,
                  origen_y=0, destino_x=0,
                  destino_y=0, velocidad=0,
                  red=0, green=0, blue=0):

        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, destino_x ,
origen_y, destino_y )

def __str__(self):

    return(

        'Id:' + str(self.__id) + '\n'

        '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' +

        'Red:' + str(self.__red) + '\n' +

        'Green:' + str(self.__green) + '\n' +

        'Blue:' + str(self.__blue) + '\n' +

        'Distancia:' + str(self.__distancia) + '\n'

    )

```

algoritmos.py

```

import math

def distancia_euclidiana(x_1, y_1, x_2, y_2):

    valor1 = x_1 - y_1

```

```
valor1**2
```

```
valor2 = x_2 - y_2
```

```
valor2**2
```

```
return math.sqrt(valor1+valor2)
```