






Actividad 05 // Clases y Objetos

Sámano Juárez Juan Jesús

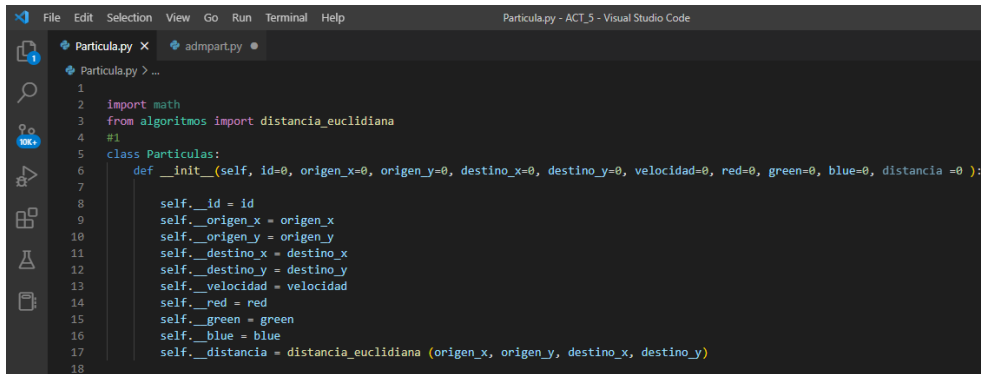
Seminario de solución de problemas de algoritmia.

Lineamientos de evaluación.

-  El reporte está en formato Google Docs o PDF.
-  El reporte sigue las pautas del Formato de Actividades .
-  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().

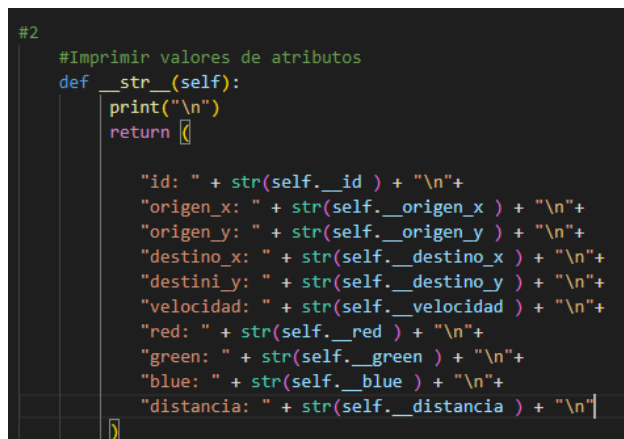
Desarrollo.

1.- Crea la clase **Particula** con los siguientes atributos privados:



```
1
2 import math
3 from algoritmos import distancia_euclidiana
4 #1
5 class Particulas:
6     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, distancia =0 ):
7
8         self.__id = id
9         self.__origen_x = origen_x
10        self.__origen_y = origen_y
11        self.__destino_x = destino_x
12        self.__destino_y = destino_y
13        self.__velocidad = velocidad
14        self.__red = red
15        self.__green = green
16        self.__blue = blue
17        self.__distancia = distancia_euclidiana (origen_x, origen_y, destino_x, destino_y)
18
```

2.- Implementa el método **__str__(self)** con el objetivo de poder imprimir un objeto de tipo **Particula**.

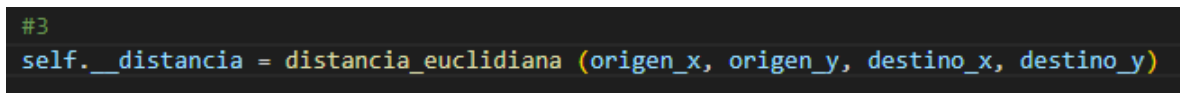


```
#2
#Imprimir valores de atributos
def __str__(self):
    print("\n")
    return [

        "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"]

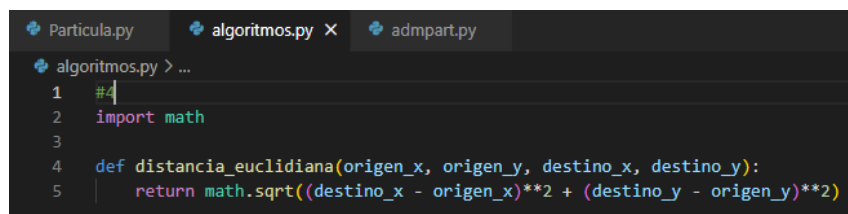
```

3.- El atributo **distancia** deberá ser calculado cada vez que se crea un objeto de la clase **Particula**, por lo que se deberá implementar el método **distancia_euclidiana()** usando la siguiente fórmula:



```
#3
self.__distancia = distancia_euclidiana (origen_x, origen_y, destino_x, destino_y)
```

4.- Implementa la función **distancia_euclidiana()** en un archivo de nombre **algoritmos.py**. Este archivo puede tener la siguiente estructura:



```
Particula.py  algoritmos.py  admpart.py
algoritmos.py > ...
1  #4
2  import math
3
4  def distancia_euclidiana(origen_x, origen_y, destino_x, destino_y):
5      return math.sqrt((destino_x - origen_x)**2 + (destino_y - origen_y)**2)
```

5.- Importa la función `distancia_euclidiana()`, que está en el archivo `algoritmos.py` que creaste en el punto 4., usando `from archivo import función` en el archivo donde definiste la clase `Particula`. El archivo puede llevar el nombre de `particula.py`.

```

Particula.py > ...
1
2 import math
3 #5
4 from algoritmos import distancia_euclidiana

```

6. Crea la clase para administrar objetos de la clase `Particula` con lo siguiente:
 1. Una **lista** privada para almacenar objetos de la clase `Particula`.
 2. Implementa, tal cómo se hicieron para la clase `Libreria`, los siguientes métodos:
 - `agregar_inicio()` : Recibe un objeto de la clase `Particula` y con `insert` agrega el objeto en la posición 0 de la lista.
 - `agregar_final()` : Recibe un objeto de la clase `Particula` y con `append` agrega el objeto en la última posición de la lista.
 - `mostrar()` : Muestra la información con `print` de cada objeto almacenado en la lista.

```

admpart.py > ...
1 from Particula import Particulas
2
3
4 class admpart:
5     def __init__(self):
6         self._almpart = []
7
8     def agregar_final(self, Particula:Particulas):
9         self._almpart.append(Particula)
10
11     def agregar_inicio(self, Particula:Particulas):
12         self._almpart.insert(0, Particula)
13
14     def mostrar(self):
15         for Particula in self._almpart:
16             print(Particula)
17
18 particula01 = Particulas(id=1, origen_x=2, origen_y=3, destino_x=4, destino_y=5, velocidad=6, red=7, green=8, blue=9, distancia=10)
19 print(particula01)
20 particula02 = Particulas(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
21 print(particula02)
22 admpart = admpart()
23 admpart.agregar_final(particula01)
24 admpart.agregar_inicio(particula02)
25 admpart.mostrar

```

7.- Toma captura de pantalla en donde se muestre el uso de los métodos `agregar_inicio()`, `agregar_final()`, `mostrar()` y reporta con código en *Google Docs* siguiendo las pautas (guidelines).

```

id: 1
origen_x: 2
origen_y: 3
destino_x: 4
destini_y: 5
velocidad: 6
red: 7
green: 8
blue: 9
distancia: 2.8284271247461903

id: 10
origen_x: 20
origen_y: 30
destino_x: 40
destini_y: 50
velocidad: 60
red: 70
green: 80
blue: 90
distancia: 28.284271247461902

```

captura de pantalla de los datos antes de usar el método `agregar_inicio()`

```
particula01 = Particulas(id=1, origen_x=2, origen_y=3, destino_x=4, destino_y=5, velocidad=6, red=7, green=8, blue=9, distancia = distancia_eucl  
particula02 = Particulas([10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
```

captura de pantalla del método `mostrar()` después de haber utilizado el método `agregar_inicio()`.

```
origen_x: 2  
origen_y: 3  
destino_x: 4  
destino_y: 5  
velocidad: 6  
red: 7  
green: 8  
blue: 9  
distancia: 2.8284271247461903  
  
id: 10  
origen_x: 20  
origen_y: 30  
destino_x: 40  
destino_y: 50  
velocidad: 60  
red: 70  
green: 80  
blue: 90  
distancia: 28.284271247461902
```

Se muestra la captura de pantalla de los datos antes de usar el método `agregar_final()`

```
particula01 = Particulas(id=1, origen_x=2, origen_y=3, destino_x=4, destino_y=5, velocidad=6, red=7, green=8, blue=9, distancia = distancia_eucl  
particula02 = Particulas([10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
```

captura de pantalla del método `mostrar()` después de haber utilizado el método `agregar_final()`.

```
id: 10  
origen_x: 20  
origen_y: 30  
destino_x: 40  
destino_y: 50  
velocidad: 60  
red: 70  
green: 80  
blue: 90  
distancia: 28.284271247461902  
  
id: 1  
origen_x: 2  
origen_y: 3  
destino_x: 4  
destino_y: 5  
velocidad: 6  
red: 7  
green: 8  
blue: 9  
distancia: 2.8284271247461903
```

Conclusiones.

Desde el punto 1 hasta el punto 3 todo lo realicé sin ningún problema, fue solo el punto 4 donde se me dificultó y a partir de ahí se realizó sin ningún inconveniente, esto gracias a el video que se nos proporcionó en la plataforma de actividades.

Referencias.

Michel Davalos Boites.[MICHEL DAVALOS BOITES](18/10/2022) PySide2 - Clases y Objetos (Qt for Python)(II)[Archivo de video].
<https://www.youtube.com/watch?v=KfQDtrrL2OU>.

Código.

Particula.py

```
import math
#5
from algoritmos import distancia_euclidiana
#1
class Particulas:
    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, distancia =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
        #3
        self.__distancia = distancia_euclidiana (origen_x, origen_y,
destino_x, destino_y)

#2
    #Imprimir valores de atributos
    def __str__(self):
        print("\n")
        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"+
        "destini_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"
    )
#Asignar valores a los atributos
#particula01 = Particulas(id=1, origen_x=2, origen_y=3, destino_x=4,
destino_y=5, velocidad=6, red=7, green=8, blue=9, distancia =
distancia_euclidiana)
#print(particula01)
#particula02 = Particulas(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
#print(particula02)

```

algoritmos.py

```

#4
import math

def distancia_euclidiana(origen_x, origen_y, destino_x, destino_y):
    return math.sqrt((destino_x - origen_x)**2 + (destino_y -
origen_y)**2)

```

admpart.py

```

from Particula import Particulas

class admpart:
    def __init__(self):
        self.__almpart = []

    def agregar_final(self, Particula:Particulas):
        self.__almpart.append(Particula)

    def agregar_inicio(self, Particula:Particulas):
        self.__almpart.insert(0, Particula)

```

```
def mostrar(self):
    for Particula in self.__almpart:
        print(Particula)

particula01 = Particulas(id=1, origen_x=2, origen_y=3, destino_x=4,
destino_y=5, velocidad=6, red=7, green=8, blue=9, distancia=10)
print(particula01)
particula02 = Particulas(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
print(particula02)
admpart = admpart()
admpart.agregar_inicio(particula02)
admpart.agregar_final(particula01)
admpart.mostrar
```