# Actividad 05 – (Clases y objetos)


## Rubio Valenzuela Miguel Angel - 216567795


### Seminario de Algoritmia – D02.

## 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.

## Creación de la clase Particula.

El desarrollo de la actividad se realiza creando principalmente la clase particula, con los atributos, id, origen_x, origen_y, destino_x, destino_y, velocidad, red, green, blue y distancia.

```python
class Particula:
    def __init__(self, id='', ox= 0.0, oy=0.0, dx=0.0, dy=0.0, vel=0.0, r=0, g=0, b=0, distancia=0.0):
        self.__id = id
        self.__origen_x = ox
        self.__origen_y = oy
        self.__destino_x = dx
        self.__destino_y = dy
        self.__velocidad = vel
        self.__red = r
        self.__green = g
        self.__blue = b
        self.__distancia = distancia
```

Aquí únicamente vemos la estructura de cómo es que funciona nuestra clase, estamos tomando valores de 0 y de vacío para que se preestablezca.

## Creación de la función distancia_euclidiana.

Esta es la manera en la que funciona la función distancia_euclidiana.

```python
import math

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

Ya lo único que resta para que se vinculen los datos, es poner la línea de código, from algoritmos import distancia_euclidiana, y agregar en la parte de distancia la función distancia_eucludiana(ox, oy, dx, dy).

# Creación de la clase lista.

Lo siguiente es la creación de la lista donde se va a guardar la información de los objetos partícula, para esto creo una clase llamada lista donde ingresaremos toda la información de estos objetos.

De igual forma que como se muestra en el video, deje la clase lista de la siguiente forma:

```
2
3   from particula import Particula
4
5   class lista():
6       def __init__(self):
7           self.__particulas = []
8
9       def agregar_final(self, particula:Particula):
10          self.__particulas.append(particula)
11
12      def agregar_inicio(self, particula:Particula):
13          self.__particulas.insert(0, particula)
14
15      def mostrar(self):
16          for particula in self.__particulas:
17              print(particula)
18
```

Para que todos los datos de la clase partícula se pudiesen mostrar en la otra clase lista, tuve que hacer que los datos obtenidos se convirtieran en cadenas de caracteres, para lograr esto tuve que ingresar la función de repr() la cual hace que los datos contenidos en enteros o flotantes se convirtieran a string.

# Uso de la función def __str__(self).

Usamos la función def __str__(self): aquí nada mas ingresamos los datos en string que se van a representar en pantalla cada que un valor sea llamado a la función.

Aquí se ve como es que la función que realice termina funcionando.

```
def __str__(self):
    return(
        'ID: ' + self.__id + '\n' +
        'Origen X: ' + self.__origen_x + '\n' +
        'Origen Y: ' + self.__origen_y + '\n' +
        'Destino X: ' + self.__destino_x + '\n' +
        'Destino Y: ' + self.__destino_y + '\n' +
        'Velocidad: ' + self.__velocidad + '\n' +
        'Rojo: ' + self.__red + '\n' +
        'Verde: ' + self.__green + '\n' +
        'Azul: ' + self.__blue + '\n' +
        'Distancia: ' + self.__distancia + '\n\n'
        )
```

De esta forma ya podemos mostrar en pantalla los valores de una partícula, sin la necesidad de hacer referencia a que todos estos se encuentran dentro de la partícula, por ejemplo, partícula P.self.__id, e ir diciendo todos de uno en uno, sino que únicamente tenemos que marcar como partícula y funcionará.

Ahora para realizar la captura de los datos tenemos que utilizar las siguientes líneas de comandos.

```
l0 = Particula(id=5, ox=6.0, oy=7.0, dx=5.0, dy=4.0)
l2 = Particula( 102 , 7, 8 ,6 ,4 ,1,6 ,4,1)

lista = Lista()

lista.agregar_inicio(l2)
lista.agregar_final(l0)

lista.mostrar()
```

De esta forma podemos ingresar datos de tipo partícula, en la lista que almacena objetos de tipo partícula, y los mostramos con la función mostrar, pues este estos son los resultados de implementar esta información.

```
PS C:\Users\cober\Desktop\Sem de Algoritmia\A5>  c:; cd '
rs\cober\AppData\Local\Programs\Python\Python310\python.e
-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\d
em de Algoritmia\A5\libreria.py'
ID: 102
Origen X: 7
Origen Y: 8
Destino X: 6
Destino Y: 4
Velocidad: 1
Rojo: 6
Verde: 4
Azul: 1
Distancia: 4.123105625617661

ID: 5
Origen X: 6.0
Origen Y: 7.0
Destino X: 5.0
Destino Y: 4.0
Velocidad: 0.0
Rojo: 0
Verde: 0
Azul: 0
Distancia: 3.1622776601683795
```

Estos son los datos que ingresamos en la clase lista, donde le ingresamos y mostramos como string toda la información.

Y pues este es el resultado de todo.

# Conclusiones

Al momento de realizar el programa tuve problemas cuando quería saber como es que se usaba la función sqrt(), para esto tuve que buscar en internet como es que podemos cambiar esto y podemos hacer que funcione, para que funcione tuve que hacer el siguiente algoritmo: math.sqrt( pow ( x_1 – x_2, 2) + pow( y_1 – y_2, 2)) y además, importamos la librería math.

Otro punto con lo que se me complico el programa fue con el de imprimir la información con la función def __str__(self):, esto porque la función solamente funciona si los datos son mandados en forma de string, y como se tiene que realizar una operación es requerido tener valores enteros y flotantes, y para esto simplemente le asigne la función repr() donde transformamos el valor de entero a string y esto hace que este correcta la función.

De ahí en mas considero que no tuve mas problemas con la resolución de esta actividad.

Me pareció una buena actividad, ya que aprendí a como vincular de mejor manera una clase con otra, en este caso, partícula con la clase lista, y como convertir el uso de un constructor con paso de parámetros a pasarlos a la lista y estos siendo mostrados de una forma más sencilla, sin la necesidad de mostrar uno por uno sino que podemos simplemente llamar a la clase y esta no mostrara la dirección de memoria, sino que mostrará la información que nosotros queramos.

# Referencias


**https://how.okpedia.org/es/python/como-calcular-la-potencia-de-un-numero-en-python - Okpedia**

**https://www.delftstack.com/es/howto/python/python-float-to-string/#:~:text=Convertir%20un%20float%20a%20una%20cadena%20en%20Python,para%20convertir%20un%20float%20en%20cadena%20en%20Python. – DelfStack –**

**https://www.youtube.com/watch?v=KfQDtrrL2OU – pyside2 – clases y objetos (Qt for python)(II) – Michel Davalos Boites.**

# Código.

**Tiene 3 archivos.**

**Librería.py**

```python
from particula import Particula

class Lista:
    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)



l0 = Particula(id=5, ox=6.0, oy=7.0, dx=5.0, dy=4.0)
l2 = Particula( 102 , 7, 8, 6, 4, 1, 6, 4, 1)

lista = Lista()

lista.agregar_inicio(l2)
lista.agregar_final(l0)

lista.mostrar()
```

**Particula.py**

```
from algoritmos import distancia_euclidiana


class Particula:
    def __init__(self, id=0.0, ox=0.0, oy=0.0, dx=0.0, dy=0.0, vel=0.0, r=0,
g=0, b=0):
        self.__id = repr(id)
        self.__origen_x = repr(ox)
        self.__origen_y = repr(oy)
        self.__destino_x = repr(dx)
        self.__destino_y = repr(dy)
        self.__velocidad = repr(vel)
        self.__red = repr(r)
        self.__green = repr(g)
        self.__blue = repr(b)
        self.__distancia = repr(distancia_euclidiana(ox, oy, dx, dy))

    def __str__(self):
        return(
            'ID: ' + self.__id + '\n' +
            'Origen X: ' + self.__origen_x + '\n' +
            'Origen Y: ' + self.__origen_y + '\n' +
            'Destino X: ' + self.__destino_x + '\n' +
            'Destino Y: ' + self.__destino_y + '\n' +
            'Velocidad: ' + self.__velocidad + '\n' +
            'Rojo: ' + self.__red + '\n' +
            'Verde: ' + self.__green + '\n' +
            'Azul: ' + self.__blue + '\n' +
            'Distancia: ' + self.__distancia + '\n'
            )
```

**Algoritmos.py**

```
import math

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