EJERCICIOS:

Estás a cargo de tu propia empresa **"El Mejor Programa S.A**." dedicada al desarrollo de software, y estás compitiendo con otras empresas similares para ganar una licitación. Dentro de las bases de la licitación, se solicita resolver diversos problemas computacionales, a través de un diseño orientado a objetos.

1. Escriba un programa en donde defina la clase Punto, la cual representa un punto bidimensional con coordenadas x e y. Construya el método constructor y los métodos getter y setter correspondientes, así como también el método___str_.

`p = Punto()`
`p.set_x(2)`
`p.set_y(3)`
`print(p)`

`(2,3)`

SOLUCION

```
class Punto:
    def __init__(self):
        self.x = 0
        self.y = 0

    def set_x(self, x):
        self.x = x

    def set_y(self, y):
        self.y = y

    def get_x(self):
        return self.x

    def get_y(self):
        return self.y

    def __str__(self):
        return f"({self.x},{self.y})"
```

2. Con la clase Punto definida en el ejercicio anterior, construya el método **cuadrante**, que indique el cuadrante[1] en el cual se encuentra el punto (cuadrante 1, 2, 3 o 4).

```
p = Punto()
p.set_x(2)
p.set_y(3)
print(p)
print(p.cuadrante())

q = Punto()
q.set_x(-3)
q.set_y(-1)
print(q)
print(q.cuadrante())
```

```
(2,3)
1
(-3,-1)
3
```

[1] https://es.wikipedia.org/wiki/Cuadrante_(geometr%C3%ADa)

SOLUCION

```
from math import sqrt

class Punto:
    def __init__(self):
        self.x = 0
        self.y = 0

    def set_x(self, x):
        self.x = x

    def set_y(self, y):
        self.y = y

    def get_x(self):
        return self.x

    def get_y(self):
        return self.y

    def __str__(self):
        return f"({self.x},{self.y})"

    def cuadrante(self):
        x,y = self.x,self.y
        if x >= 0 and y >= 0:
            return 1
        elif x < 0 and y >= 0:
            return 2
        elif x < 0 and y < 0:
            return 3
        else:
            return 4
```

3. Considere la misma clase Punto definida en los ejercicios anteriores. Ahora construya el método **distancia** que reciba como parámetro otro objeto de tipo Punto y retorne la distancia Euclediana[2] entre ellos.

```
p = Punto()
p.set_x(2)
p.set_y(3)
print(p)

q = Punto()
q.set_x(-3)
q.set_y(-1)
print(q)

d = p.distancia(q)
print(d)
```

```
(2,3)
(-3,-1)
6.4031242374328485
```

[2] https://es.wikipedia.org/wiki/Distancia_euclidiana

SOLUCION

```python
from math import sqrt

class Punto:
    def __init__(self):
        self.x = 0
        self.y = 0

    def set_x(self, x):
        self.x = x

    def set_y(self, y):
        self.y = y

    def get_x(self):
        return self.x

    def get_y(self):
        return self.y

    def __str__(self):
        return f"({self.x},{self.y})"

    def cuadrante(self):
        x,y = self.x,self.y
        if x >= 0 and y >= 0:
            return 1
        elif x < 0 and y >= 0:
            return 2
        elif x < 0 and y < 0:
            return 3
        else:
            return 4

    def distancia(self, pto):
        x,y = self.x,self.y
        xp,yp = pto.get_x(), pto.get_y()
        d = sqrt((x - xp) ** 2 + (y - yp) ** 2)
        return d
```

4. Escriba un programa en donde defina la clase Circunferencia, la cual representa una circunferencia con centro en un punto x,y bidimensional y un radio r. Construya el método constructor y los métodos getter y setter correspondientes así como también el método_str_.

```
c = Circunferencia()
c.set_centro(1,1)
c.set_radio(3)
print(c)
```

```
centro:(1,1) , radio:3
```

SOLUCION

```
from punto import Punto

class Circunferencia:
    def __init__(self):
        self.centro = Punto()
        self.r = 0

    def set_centro(self, x, y):
        self.centro.set_x(x)
        self.centro.set_y(y)

    def set_radio(self, r):
        self.r = r

    def get_centro(self):
        return self.centro

    def get_radio(self):
        return self.r

    def __str__(self):
        return f"centro:({self.centro.x},{self.centro.y}) , radio:{self.r}"
```

5. Con la clase Circunferencia definida en el ejercicio anterior, construya los métodos **area** y **perimetro**, que indique el área y el perímetro de la circunferencia respectivamente.

```
c = Circunferencia()
c.set_centro(1,1)
c.set_radio(3)
print(c)
print(f"area: {c.area()}")
print(f"perimetro: {c.perimetro()}")
```

```
centro:(1,1) , radio:3
area: 28.274333882308138
perimetro: 18.84955592153876
```

SOLUCION

```
from punto import Punto
from math import pi

class Circunferencia:
    def __init__(self):
        self.centro = Punto()
        self.r = 0

    def set_centro(self, x, y):
        self.centro.set_x(x)
        self.centro.set_y(y)

    def set_radio(self, r):
        self.r = r

    def get_centro(self):
        return self.centro

    def get_radio(self):
        return self.r

    def __str__(self):
        return f"centro:({self.centro.x},{self.centro.y}) , radio:{self.r}"

    def area(self):
        return pi * self.r ** 2

    def perimetro(self):
        return 2 * pi * self.r
```

6. Considere la misma clase Circunferencia definida en los ejercicios anteriores. Ahora construya el método **interior** que reciba como parámetro otro objeto de tipo Punto e indique si el punto se encuentra al interior o no de la circunferencia.

```
c = Circunferencia()
c.set_centro(1,1)
c.set_radio(3)
print(c)

puntos = [[1,2],[-1,2],[-1,-2],[3,3],[-2,4]]
objPuntos  =  []
for pto in puntos:
    x,y = pto
    p = Punto()
    p.set_x(x)
    p.set_y(y)
    objPuntos.append(p)

for p in objPuntos:
    print(c.interior(p))
```

```
centro:(1,1) , radio:3
True
True
False
True
False
```

SOLUCION

```
from punto import Punto
from math import pi

class Circunferencia:
    def __init__(self):
        self.centro = Punto()
        self.r = 0

    def set_centro(self, x, y):
        self.centro.set_x(x)
        self.centro.set_y(y)

    def set_radio(self, r):
        self.r = r

    def get_centro(self):
        return self.centro

    def get_radio(self):
        return self.r

    def __str__(self):
        return f"centro:({self.centro.x},{self.centro.y}) , radio:{self.r}"

    def area(self):
        return pi * self.r ** 2

    def perimetro(self):
        return 2 * pi * self.r

    def interior(self, pto):
        d = self.centro.distancia(pto)
        if d <= self.r:
            return True
        else:
            return False
```