Laboratorio M3-15

Realizar los siguientes ejercicios en tu Editor de Código preferida, adjuntar tu archivo py.

Enviar laboratorio a: jpruiz@itgcorp.co

Recuerda adjuntar el número de grupo al que perteneces en el asunto del email

Ejercicio No.15 – Puntos en un plano

Tiempo Estimado

30 – 60 minutos

Nivel de Dificultad

Fácil / Medio

Objetivos

- Mejorar las habilidades del estudiante para definir clases desde cero.
- Definir y usar variables de instancia.
- Definir y usar métodos.

Escenario

Visitemos un lugar muy especial: un plano con el sistema de coordenadas cartesianas (puedes obtener más información sobre este concepto aquí: https://en.wikipedia.org/wiki/Cartesian_coordinate_system

Cada punto ubicado en el plano puede describirse como un par de coordenadas habitualmente llamadas **x** y **y**. Queremos que escribas una clase en Python que almacene ambas coordenadas como números flotantes. Además, queremos que los objetos de esta clase evalúen las distancias entre cualquiera de los dos puntos situados en el plano.

La tarea es bastante fácil si empleas la función denominada **hypot()** (disponible a través del módulo math) que evalúa la longitud de la hipotenusa de un triángulo rectángulo (más detalles aquí: https://en.wikipedia.org/wiki/Hypotenuse) y aquí: https://docs.python.org/3.7/library/math.html#trigonometric-functions

Así es como imaginamos la clase:

- Se llama **Point**.
- Su constructor acepta dos argumentos (**x** y **y** respectivamente), ambos por defecto se igualan a cero.
- Todas las propiedades deben ser privadas.
- La clase contiene dos métodos sin parámetros llamados **getx()** y **gety()**, que devuelven cada una de las dos coordenadas (las coordenadas se almacenan de forma privada, por lo que no se puede acceder a ellas directamente desde el objeto).
- La clase proporciona un método llamado **distance_from_xy(x,y)**, que calcula y devuelve la distancia entre el punto almacenado dentro del objeto y el otro punto dado en un par de números flotantes.
- La clase proporciona un método llamado **distance_from_point(point)**, que calcula la distancia (como el método anterior), pero la ubicación del otro punto se da como otro objeto de clase Point.

Completa la plantilla que te proporcionamos en el editor, ejecuta tu código y verifica si tu salida se ve igual que la nuestra.

Salida Esperada

1.4142135623730951
1.4142135623730951

Código base

```python
import math


class Point:
    def __init__(self, x=0.0, y=0.0):
        #
        # Escribir el código aquí.
        #

    def getx(self):
        #
        # Escribir el código aquí.
        #

    def gety(self):
        #
        # Escribir el código aquí.
        #

    def distance_from_xy(self, x, y):
        #
        # Escribir el código aquí.
        #

    def distance_from_point(self, point):
        #
        # Escribir el código aquí.
        #


point1 = Point(0, 0)
point2 = Point(1, 1)
print(point1.distance_from_point(point2))
print(point2.distance_from_xy(2, 0))
```