Instituto Politécnico Nacional
Escuela Superior de Cómputo

Paradigmas de Programación

Actividad: Ejercicio de laboratorio 7

Profesor: García Floriano Andrés

Fecha: 13/05/2024

Alumno:

Pacheco Refugio Alan Ivan

Grupo: 3CV1

```python
import math
import random

class Punto3D:
    def __init__(self, x=0, y=0, z=0):
        self.__x = x
        self.__y = y
        self.__z = z

    @classmethod
    def from_point(cls, punto):
        return cls(punto.get_x(), punto.get_y(), punto.get_z())

    def get_x(self):
        return self.__x

    def get_y(self):
        return self.__y

    def get_z(self):
        return self.__z
```

Las primeras dos líneas de códigos importan las librerías necesarias para funciones matemáticas, así como para generar números aleatorios.

Después se define la clase Punto3D la cual tiene un constructor llamado __init__ el cual se llama automáticamente cuando se crea una nueva instancia

El constructor tiene parámetros x, y y z con valores por defecto de 0. Si no se proporcionan valores al crear un objeto Punto3D, estos valores predeterminados se utilizarán.

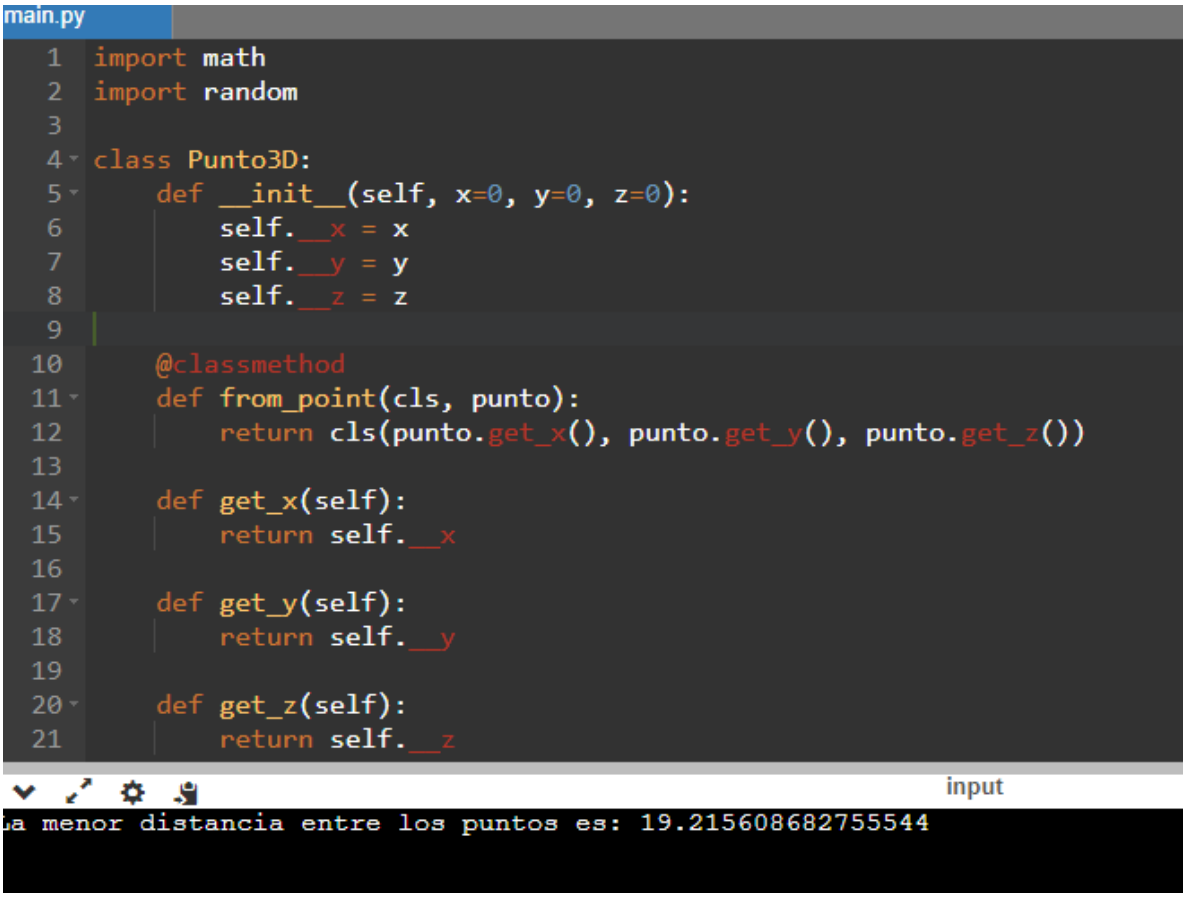
@classmethod es un decorador que indica que el método siguiente es un método de clase. Esto significa que el método from_point está ligado a la clase Punto3D y no a una instancia particular de la clase.

get_x devuelve el valor del atributo __x.

get_y devuelve el valor del atributo __y.

get_z devuelve el valor del atributo __z

Pruebas



Se pueden hacer con distintos valores e irá cambiando respecto a los datos que se proporcionan en el código

```
        return self.__z

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

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

    def set_z(self, z):
        self.__z = z

    def distancia(self, otro_punto):
        return math.sqrt(
            (self.__x - otro_punto.get_x())**2 +
            (self.__y - otro_punto.get_y())**2 +
            (self.__z - otro_punto.get_z())**2
        )

# Crear un arreglo de 10 objetos de tipo Punto3D con valores aleatorios
puntos = [Punto3D(random.uniform(0, 100), random.uniform(0, 100), random.uniform(0, 100)) for _ in range(10)]
```

```python
# Crear un arreglo de 10 objetos de tipo Punto3D con valores aleatorios
puntos = [Punto3D(random.uniform(0, 100), random.uniform(0, 100), random.uniform(0, 100)) for _ in range(10)]

# Función para calcular la menor distancia entre todos los puntos
def menor_distancia(puntos):
    min_distancia = float('inf')
    for i in range(len(puntos)):
        for j in range(i + 1, len(puntos)):
            distancia = puntos[i].distancia(puntos[j])
            if distancia < min_distancia:
                min_distancia = distancia
    return min_distancia

# Calcular y mostrar la menor distancia
min_dist = menor_distancia(puntos)
print(f"La menor distancia entre los puntos es: {min_dist}")
```