POO y algoritmos

Decoradores: Los decoradores son funciones de orden mayor, ósea, funciones que toman una función como parámetro y/o retornan una función, añade capacidades a una función sin modificarla.

En Python las funciones son ciudadanos de primera clase, ósea, pueden ser pasadas y utilizadas como argumentos.

Un decorador toma una función le añade una funcionalidad y la retorna.

```
def funcion_decoradora(funcion):
    def wrapper():
        print("Este es el último mensaje...")
        funcion()
        print("Este es el primer mensaje ;)")
    return wrapper


def zumbido():
    print("Buzzzzzz")


zumbido = funcion_decoradora(zumbido)
```

En el ejemplo anterior se le agrega una función a zumbido y se retorna la función, haciendo que el nuevo zumbido posea la funcionalidad agregada.

Para una mejora de sintaxis

```
@funcion_decoradora
def zumbido():
    print("Buzzzzzz")
```

```
    @property
    def region(self):
        return self.__region

    @region.setter
    def region(self, region):
        if region in self.__pais:
            self.__region = region
        else:
            raise ValueError(f'La region {region} no es valida en
{self.__pais}')
```

@property es para implementar un getter

@property_name.setter es un decorador, el cual es para implementar un setter.

Para encapsular algo (prevenir el acceso no deseado) en Python:

__nombre_atributo

El Big O notation se centra en hallar la complejidad en el termino de mayor tamaño, lo que le interesa es el peor de los casos.

Reducir un problema de solución desconocida, a uno el cual ya sabemos su solución.

Optimización: Función en la cual se quiere maximizar o minimizar, encontrar la entrada que nos da la salida mas alta o mas baja.