**VARIABLES DE CLASE**

Las variables de clase son variables que son iguales para cada objeto, a diferencia de las variables de instancia, las cuales poseen un valor != para cada instancia.

Para declarar una variable de clase, se define un atributo **fuera del método \_\_init\_\_**

Codigo de ejemplo:

class Banda:

    def \_\_init\_\_(self, cant, gen) -> None:

        self.\_miembros = cant

        self.\_genero = gen

    def \_\_str\_\_(self) -> str:

        return f'[Cantidad de miembros: {self.\_miembros} , Genero: {self.\_genero}]'

    variable\_clase = 'I\'m in the band'

Si se quisiese se podría modificar el valor de **variable\_clase** para cada instancia. Aún así, el valor para **Banda.variable\_clase** mientras no se modifique directamente, se mantendrá inmutado.

**METODOS ESTATICOS EN PYTHON**

Los métodos estáticos son el equivalente a las variables de clase en métodos. Estos se relacionan directamente con las clases y por tanto, no acceden a la variable **self** ya que esta hace referencia a una instancia de la clase.

Estos métodos se usan “antes” de la creación de una instancia por tanto tiene sentido que no accedan a **self**

Contexto estatico 🡪 métodos y atributos de clase

Contexto dinámico 🡪 métodos y atributos de instancia, una vez que se crea el objeto en memoria

Si quisiésemos acceder a atributos de clase, deberíamos hacerlo de esta manera:

    @staticmethod

    def descripcion():

        return f'Esto es un metodo estatico, relacionado con la class Banda. Una variable de clase es "{Banda.variable\_clase}"'

**METODOS DE CLASE**

La principal diferencia entre métodos de clase y métodos estáticos, es que los segundos no reciben parámetros (como puede ser la variable **self**) con información referida a la clase.

Los métodos de clase, si lo hacen y por tanto pueden acceder a información de la misma.

Con esta definición los métodos estáticos son como si fuesen definidos fuera de la clase

Este es un ejemplo de un método de clase

    @classmethod

    def metodo\_clase(cls):

        print(cls.variable\_clase)

como se ve, recibe la referencia **cls** como parámetro que contiene la referencia a la clase.

**CONTEXTO ESTÁTICO Y CONTEXTO DINÁMICO**

El contexto dinámico puede acceder al contexto estático,pero no viceversa.

Cuando se declara una instancia, esta puede acceder a los métodos de clase porque la Clase ya está definida en memoria. Por el contrario, la clase no puede acceder al contexto dinámico porque “aun no se ha creado una instancia”.

**CONSTANTES**

El concepto de constante no existe como tal en Python.

Las constantes suelen definirse en un modulo a parte.

Por convención, las constantes se declaran en MAYUSCULAS y aunque en el lenguaje sea posible cambiar su valor, debe respetarse la convención.

En este código de ejemplo se emula el funcionamiento de la clase Math.

class Matematicas:

    def \_\_init\_\_(self) -> None:

        pass

    CONST\_PI = 3.1416

    @classmethod

    def PI(cls):

        print(cls.CONST\_PI)

y desde otro archivo se importa la clase Matematicas para operar:

from Constantes import \*

Matematicas.PI()

**EJERCICIO CREAR UN CONTADOR DE OBJETOS CREADOS**

Para este ejercicio se crea una variable de clase, al inicio de la misma para que sea compartida para todas las instancias.

contador\_bandas = 0

Luego, en el método init (el que se invoca cada vez que se instancia la clase) “aumentamos“ este contador.

Además, este contador se agrega como atributo a cada instancia, para manejar un attr **id\_banda**

    def \_\_init\_\_(self, cant, gen) -> None:

        self.\_miembros = cant

        self.\_genero = gen

        Banda.contador\_bandas += 1

        self.\_id\_banda = Banda.contador\_bandas

Ahora, el atributo de clase se comparte a todos los elementos, pero para todos tiene un valor distinto.