U6P01-Programas_01. Orientación a objetos

1. Descripción

Desarrolla y prueba en Python los programas que permitan resolver los requisitos planteados.

Dentro de la carpeta **PEPXX** crea las subcarpetas **U6/01_orientación_a_objetos .** Para cada programa crea un archivo que contenga: al inicio comentando el enunciado y a continuación la solución.

Puedes crear un repositorio en GitHub con todo el código.

2. Formato de entrega

Será propuesto en clase por el profesor.

3. Trabajo a realizar

Programa01: Clase básica con constructor y métodos de instancia

Escribe un programa en Python que :

- Cree una clase Animal con:
  - Atributos de instancia: `nombre, especie y edad`
  - Constructor `__init__(self, nombre, especie, edad)`
  - Método de instancia `saluda` que muestre algo como:

    ```
    Soy un tigre llamado Kuma y tengo 5 años.
    ```

  - Método de instancia `cumplir_anios` que aumente en 1 la edad.
- Crea una función `main` que:
  - Cree dos animales distintos.
  - Muestre su `nombre, edad` y `especie`.
  - Llame a `saluda() y` luego a `cumplir_anios()` para cada uno.
  - Cambie el nombre del un animal y que se presente de nuevo.

Programa02: Encapsulación con atributo “privado” y getters/setters

Escribe un programa en Python que :

- Amplíe la clase Animal para manejar un atributo no público:
  - Añade al constructor un parámetro `id_chip` y guarda el valor en un atributo privado.

    - Implementa los métodos:
        - `get_id_chip(self)` → devuelve el id_chip.
        - `set_id_chip(self, nuevo_id)`
            - Solo acepta valores de tipo str no vacíos.
            - Si el valor no es del tipo adecuado lanza una excepción de tipo TypeError.
            - Si el valor no es un id_chip adecuado lanza una excepción del tipo ValueError.
- Crea una función `main` que:
    - Cree un Animal con un `id_chip` inicial.
    - Muestra el chip con `get_id_chip()`.
    - Intenta cambiarlo por:
        - un valor válido (por ejemplo "ABC123"),
        - un valor no válido (por ejemplo 123 o ""), y comprueba el comportamiento.

Recuerda: el atributo sigue siendo accesible con `_Animal__id_chip`, pero no se debe usar en el código normal.

Programa03: Propiedades con @property

Escribe un programa en Python que :

- Modifique la clase Animal
    - Añade al constructor un parámetro `peso` y guarda el valor en un atributo privado.
    - Modifica los get y set del parámetro `id_chip` para propiedades (usando decoradores `@property`).
    - Define propiedades (get, set y delete) para el atributo `peso` (usando decoradores `@property`)
- Crea una función `main` que:
    - Cree un `Animal` con peso inicial.
    - Muestre el peso
    - Cambie el peso.
    - Elimina el peso y comprueba qué ocurre si se intenta mostrar el peso.

Programa04: Atributos de clase, @classmethod y @staticmethod

Escribe un programa en Python que:

- Amplíe la clase Animal
    - Crea un atributo de clase: `numero_animales = 0`
    - En el `__init__`, cada vez que se cree un `Animal`, incrementa `Animal.numero_animales`.
        - Crea un **método de clase**:

```
@classmethod
def contar_animales(cls):
     return cls.numero_animales
```

- Crea un **método estático**:

```
@staticmethod
def es_mayor_de_edad(edad):
      """Devuelve True si el animal se considera adulto (≥ 2
          años)."""
      return edad >= 2
```

- Crea una función main que:
  - Cree varios animales.
  - Muestre cuántos animales hay creados usando:
    `Animal.contar_animales()`
  - Compruebe el método estático.