Insaforp
FORMANDO UN MEJOR EL SALVADOR

FORMÁS
CENTRO DE FORMACIÓN AVANZADO PARA PROFESIONALES

FEDISAL
Fundación para la Educación Integral Salvadoreña

MODULO 1. INTRODUCCIÓN AL DESARROLLO DE SOFTWARE CON PYTHON

2. PROGRAMACIÓN ORIENTADA A OBJETOS

CONTENIDOS A DESARROLLAR

1. Herencia
   1. Clases abstractas
   2. Herencia múltiple
2. Variables de clase (estáticas)
3. @classmethod vs @staticmethod
4. Constantes
5. Diseño de clases (Diagramar)

HERENCIA

- La herencia es un proceso mediante el cual se puede crear una clase hija que hereda de una clase padre.
- Compartiendo sus métodos y atributos.
- Además de ello, una clase hija puede sobrescribir los métodos o atributos, o incluso definir unos nuevos.
- El principio DRY (Don't Repeat Yourself) consiste en no repetir código de manera innecesaria.
- la función super() nos permite acceder a los métodos de la clase padre desde una de sus hijas
- Todas las clases heredan de la clase Object
- Elaborar ejemplos según indicaciones del instructor

CLASES ABSTRACTAS

- Las clases abstractas no pueden ser instanciadas
- Solo usadas con herencia
- Obliga implementar los métodos abstractos en las clases derivadas

HERENCIA MÚLTIPLE

- La herencia múltiple es similar, pero una clase hereda de varias clases padre en vez de una sola.
- La prioridad de la herencia múltiple a de izquierda a derecha, pero la clase derivada tiene la prioridad 1

- Elaborar ejemplos según indicaciones del instructor

VARIABLES DE CLASE

- Atributos de clase
- Asociados a la clase, no a cada instancia de objetos
- Prácticamente variables estáticas
- La diferencia con otros lenguajes es que si modificamos un atributo de clase desde el objeto, este crea una nueva versión del atributo disponible solo por ese objeto.

- Elaborar ejemplos según indicaciones del instructor

@CLASSMETHOD

- Un método de clase recibe la clase como primer argumento implícito, al igual que un método de instancia recibe la instancia.
- Un método de clase es un método que está vinculado a la clase y no al objeto de la clase.
- Tienen acceso al estado de la clase, ya que toma un parámetro de clase que apunta a la clase y no a la instancia del objeto.
- Puede modificar un estado de clase que se aplicaría en todas las instancias de la clase. Por ejemplo, puede modificar una variable de clase que será aplicable a todas las instancias.

@STATICMETHOD

- Un método estático no recibe un primer argumento implícito
- Un método estático también es un método que está vinculado a la clase y no al objeto de la clase.
- Un método estático no puede acceder ni modificar el estado de la clase.
- Está presente en una clase porque tiene sentido que el método esté presente en la clase.

@CLASSMETHOD VS @STATICMETHOD

- Un método de clase toma cls como primer parámetro, mientras que un método estático no necesita parámetros específicos.
- Un método de clase puede acceder o modificar el estado de la clase, mientras que un método estático no puede acceder o modificarlo.
- En general, los métodos estáticos no saben nada sobre el estado de la clase. Son métodos de tipo de utilidad que toman algunos parámetros y funcionan sobre esos parámetros. Por otro lado, los métodos de clase deben tener la clase como parámetro.
- Usamos el decorador @classmethod en Python para crear un método de clase y usamos el decorador @staticmethod para crear un método estático en Python.
- ¿Cuándo usar qué?
- Generalmente usamos el método de clase para crear métodos de fábrica. Los métodos de fábrica devuelven un objeto de clase (similar a un constructor) para diferentes casos de uso.
- Generalmente usamos métodos estáticos para crear funciones de utilidad.
- Elaborar ejemplos según indicaciones del instructor

CONSTANTES

- No hay constantes en Python, al menos no una definición proporcionada por el lenguaje
- Se sugiere crearlas en un modulo separado con mayúsculas ya sea en una clase o no. Y luego importarlo
- También es posible crear un método que retorne este valor para que no pueda ser modificado por ejemplo

```
def GRAVEDAD():
        return 9.8
```