EXERCISES QUE TRABAJAREMOS EN EL CUE

- EXERCISE 1: IMPLEMENTACIÓN EN PYTHON DE LA PROGRAMACIÓN ORIENTADA A OBJETOS QUE MODELE LA CLASE VEHÍCULO CON MÉTODOS O COMPORTAMIENTOS

EXERCISES 1: IMPLEMENTACIÓN EN PYTHON DE LA PROGRAMACIÓN ORIENTADA A OBJETOS QUE MODELE LA CLASE VEHÍCULO CON MÉTODOS O COMPORTAMIENTOS

El objetivo del presente ejercicio es plasmar en el lenguaje de programación Python el concepto de clase, pilar sobre el que está construido todo el paradigma de la programación orientada a objetos.

Para comenzar, debemos identificar qué tipos de objetos se observan. Podemos tener vehículos, personas, semáforos, animales, entre otros. Debemos tomar en cuenta qué tipo y clase son términos sinónimos, y en este sentido modificar por clases de objetos que observamos, tales como los mencionados anteriormente.

Partiendo de ello, tenemos la clase de objeto llamada `Vehiculo`, la cual contiene ciertas características que nos pueden diferenciar una de otros, por ejemplo: marca, modelo, color, cilindrada, entre otros. Estas características son los atributos de la clase vehículo.

Adicionalmente, podemos tener un comportamiento de la clase vehículo como: arrancar, detenerse, girar a la izquierda, girar a la derecha, acelerar, frenar. Este comportamiento de una clase se conoce como métodos.

En este sentido, una clase esta compuestas por atributos y métodos. Lo que se realizará a continuación, es modelar en el lenguaje de programación Python la clase `Vehiculo`. Modelar no es más que la abstracción que represente algún modo una determinada realidad, en nuestro caso el vehículo.

REPRESENTACIÓN DE LA CLASE VEHICULO EN PYTHON

Para crear en Python la clase `Vehiculo`, comenzamos usando la palabra reservada **class**, seguida del nombre de la clase y del símbolo de dos puntos:

```
class Vehiculo:
    instrucción 1
    instrucción 2
    ...
    instrucción n
```

La clase más simple y que no ejecuta nada es:

```
class Vehiculo:
    pass
```

La palabra reservada **pass** pasa completamente, no hace absolutamente nada. Es útil en contextos en los que se requiere la presencia de al menos una instrucción, hasta que la inspiración acuda y coloquemos código sustituto.

Típicamente, indentados en la definición, introduciremos los atributos y métodos de que consta la clase `Vehiculo`.

El nombre de la clase `Vehiculo` lo escribimos con la primera letra en mayúsculas, pues es práctica común entre la POO que los nombres de clases empiecen así.

Introduzcamos algunos atributos:

```
class Vehiculo:
    modelo = ''
    marca = ''
    color = ''
    velocidad = 0
    nro_puertas = 0
    kilometraje = 0
    arrancando = False
```

Definimos los atributos iniciales de la clase `Vehiculo` con cadena vacía tipo String y cero para los numéricos, y False de tipo booleano para el estado de arrancando, pero podrían haber sido otros cualesquiera. En el momento en que comencemos a fabricar objetos de esta clase, todos estos atributos podrán recibir su valor concreto y particular.

Definamos ahora los métodos de la clase `Vehiculo`:

```
class Vehiculo:
    modelo = ''
    marca = ''
    color = ''
    velocidad = 0
    nro_puertas = 0
    kilometraje = 0
    arrancando = False

    def encender(self):
        pass

    def parar(self):
        pass

    def acelerar(self):
        pass

    def frenar(self):
        pass

     def get_velocidad(self):
        pass

    def get_kilometraje(self):
          pass

    def set_kilometraje(self, kilometraje):
        pass
```

Podemos notar que los métodos se implementan exactamente igual que las funciones con la palabra reservada **def,** seguida por el nombre del método, y unos paréntesis con la declaración de parámetros formales, y finalizando con dos puntos e indentando el cuerpo del método a continuación. Al igual que las funciones ordinarias, los métodos podrán o no devolver un valor.

La palabra reservada **self** es un parámetro que debe figurar siempre al comienzo de todo método dentro de una clase. El parámetro **self** hace referencia al objeto concreto sobre el que se está invocando el método.

Construyendo cada uno de los comportamientos de la clase `Vehiculo`, tenemos:

```
class Vehiculo:
    modelo = ''
    marca = ''
    color = ''
    velocidad = 0
    nro_puertas = 0
    kilometraje = 0
    encendido = False

    def encender(self):
        if not self.encendido:
            print('Encendiendo el vehiculo')
            self.encendido = True
        else:
            # Dale al encendido estando el coche arrancado y
escucha...
            print('vehiculo encendido suena el arranque')

    def apagar(self):
        self.encendido = False

    def acelerar(self):
        if self.encendido:
            self.velocidad = self.velocidad + 1

    def frenar(self):
        if self.velocidad > 0:
            self.velocidad = self.velocidad - 1

    def get_velocidad(self):
        return self.velocidad

    def get_kilometraje(self):
        return self.kilometraje

    def set_kilometraje(self, kilometraje):
        self.kilometraje = kilometraje
```

Al definir la clase **Vehiculo** con sus atributos y métodos podemos crear objetos de dicha clase, construyendo cada uno de los comportamientos de la clase **Vehiculo** tenemos:

```
obj_carro1 = Vehiculo()
obj_carro2 = Vehiculo()
print (obj_carro1.marca)
print (obj_carro2.modelo)
obj_carro1.marca = 'Fiat'
obj_carro1.modelo = 'Uno'
obj_carro1.kilometraje = 100
print (obj_carro1.marca)
print (obj_carro1.kilometraje)
obj_carro1.set_kilometraje(500)
print (obj_carro1.kilometraje)
print(obj_carro1.get_kilometraje())
```

Podemos también crear un método constructor por defecto de la clase de la siguiente manera:

```
    def __init__(self, modelo, marca, color, nro_puertas,
kilometraje):
        self.modelo = modelo
        self.marca = marca
        self.color = color
        self.velocidad = 0
        self.nro_puertas = nro_puertas
        self.kilometraje = kilometraje
        self.encendido = False
```

Aquí observamos que pasamos 6 parámetros al constructor, la velocidad y encendido tienen valores iniciales por defecto, el valor de la velocidad es 0, y el estado del encendido es falso.

Ahora construimos los objetos de la siguiente manera:

```
obj_carro1 = Vehiculo('Duro','Toyota', 'Blanco', 2, 2000)
obj_carro2 = Vehiculo('Uno','Fiat', 'Gris', 5, 100000)
```