**HERENCIA**

* **Dentro de un mismo tipo de objetos (por ejemplo, las bicicletas), existe una variedad de modelos (normales, de carreras, de montaña, etc) que dificulta la definición de la clase genérica. Por ejemplo, si defino los métodos subir\_marcha() y bajar\_marcha(), este método sólo se podrá aplicar a aquellas bicicletas que tengan cambios.**
* **Para resolver este tipo de problemas, la programación orientada a objetos hace uso de la** **herencia. La herencia consiste en definir una clase a partir de otra ya existente. De esta forma podemos definir una bicicleta genérica y mediante herencia crear otras clases más especializadas que respondan a las necesidades de cada tipo de bicicleta. Se puede observar el ahorro que supone a la hora de escribir código. Por ejemplo:**

|  |
| --- |
| Image14 |

**La sintaxis para realizar el proceso de herencia es indicar entre paréntesis el nombre de la clase de la que queremos heredar.**

**class NombreClaseDerivada(NombreClaseBase):**

**Propiedades y métodos de la clase**

En el siguiente ejemplo vamos a crear una clase genérica bicicleta que tendrá el comportamiento lógico de todos los modelos de bicicleta.

A continuación, crearemos la clase BicicletaCarreras que heredará de la clase Bicicleta e implementará dos métodos nuevos: subirMarcha() y bajarMarcha().

**ClaseBicicleta.py**

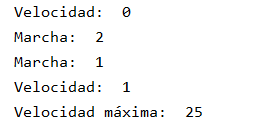
**class** Bicicleta:  
 color = **""** tamanio = **""** Vel\_max=15

velocidad=-1  
 **def** \_\_init\_\_(self):  
 self.velocidad = 0  
 **def** subirvelocidad (self):  
 self.velocidad = self.velocidad+1  
 **def** bajarvelocidad(self):  
 self.velocidad = self.velocidad - 1  
 **def** cambiarVelMax(self,maxVel):  
 self.Vel\_max = maxVel  
  
**class** BicicletaCarreras(Bicicleta):  
 marcha=0  
 **def** subirmarcha(self):  
 self.marcha = self.marcha + 1  
 **def** bajarmarcha(self):  
 self.marcha = self.marcha - 1

Ciclista creará una BicicletaCarreras y veremos que puede acceder a los métodos subirMarcha, bajarMarcha y a los que está heredando.

**Ciclista.py**

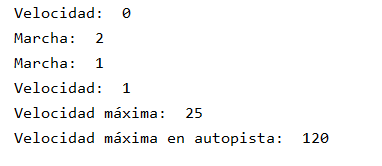
**from** ClaseBicicleta **import** BicicletaCarreras  
  
mibici = BicicletaCarreras()  
print (**"Velocidad: "**,mibici.velocidad)  
mibici.subirmarcha()  
mibici.subirmarcha()  
print (**"Marcha: "**,mibici.marcha)  
mibici.bajarmarcha()  
print (**"Marcha: "**,mibici.marcha)  
mibici.subirvelocidad()  
print (**"Velocidad: "**,mibici.velocidad)  
mibici.cambiarVelMax(25)  
print (**"Velocidad máxima: "**,mibici.Vel\_max)



**SUPER()**

Permite acceder desde la clase hija a un método de la superclase.

En el ejemplo accedemos desde el método velocidadMaximaAutopista de BicicletaCarreras al método cambiarVelMax de Bicicleta.



**ClaseBicicleta.py**

**class** Bicicleta:  
 color = **""** tamanio = **""** Vel\_max=15  
 velocidad=-1  
 **def** \_\_init\_\_(self):  
 self.velocidad = 0  
 **def** subirvelocidad (self):  
 self.velocidad = self.velocidad+1  
 **def** bajarvelocidad(self):  
 self.velocidad = self.velocidad - 1  
 **def** cambiarVelMax(self,maxVel):  
 self.Vel\_max = maxVel  
  
**class** BicicletaCarreras(Bicicleta):  
 marcha=0  
 **def** subirmarcha(self):  
 self.marcha = self.marcha + 1  
 **def** bajarmarcha(self):  
 self.marcha = self.marcha - 1  
 **def** velocidadMaximaAutopista(self):  
 super().cambiarVelMax(120)

**Ciclista.py**

**from** ClaseBicicleta **import** BicicletaCarreras

mibici = BicicletaCarreras()  
print (**"Velocidad: "**,mibici.velocidad)  
mibici.subirmarcha()  
mibici.subirmarcha()  
print (**"Marcha: "**,mibici.marcha)  
mibici.bajarmarcha()  
print (**"Marcha: "**,mibici.marcha)  
mibici.subirvelocidad()  
print (**"Velocidad: "**,mibici.velocidad)  
mibici.cambiarVelMax(25)  
print (**"Velocidad máxima: "**,mibici.Vel\_max)  
mibici.velocidadMaximaAutopista()  
print (**"Velocidad máxima en autopista: "**,mibici.Vel\_max)