











Clases y objetos

>>> c1 = Coche('rojo', 20)

>>> print(c1.color)

rojo

>>> print(c1.ruedas)

4

>>> c2 = Coche('azul', 30)

>>> print(c2.color)

azul

>>> print(c2.ruedas)

4

Constructor

obj = MiClase()

def \_\_init\_\_(self, color, aceleracion):

self.color = color

self.aceleracion = aceleracion

self.velocidad = 0

c1 = Coche('rojo', 20)

Atributos y métodos

>>> c1 = Coche('rojo', 20)

>>> print(c1.color)

rojo

>>> print(c1.velocidad)

0

>>> c1.acelera()

>>> print(c1.velocidad)

20

Instancia

>>> c1 = Coche('rojo', 20)

>>> c2 = Coche('azul', 20)

>>> print(c1.color)

rojo

>>> print(c2.color)

azul

>>> print(c1.ruedas) # Atributo de clase

4

>>> print(c2.ruedas) # Atributo de clase

4

>>> Coche.ruedas = 6 # Atributo de clase

>>> print(c1.ruedas) # Atributo de clase

6

>>> print(c2.ruedas) # Atributo de clase

6

>>> c1 = Coche('rojo', 20)

>>> c2 = Coche('azul', 20)

>>> print(c1.color)

rojo

>>> print(c2.color)

azul

>>> c1.ruedas = 6 # Crea el atributo de instancia ruedas

>>> print(c1.ruedas)

6

>>> print(c2.ruedas)

4

>>> print(Coche.ruedas)

4

Herencia

class CocheVolador(Coche):

ruedas = 6

def \_\_init\_\_(self, color, aceleracion, esta\_volando=False):

super().\_\_init\_\_(color, aceleracion)

self.esta\_volando = esta\_volando

def vuela(self):

self.esta\_volando = True

def aterriza(self):

self.esta\_volando = False

>>> c = Coche('azul', 10)

>>> cv1 = CocheVolador('rojo', 60)

>>> print(cv1.color)

rojo

>>> print(cv1.esta\_volando)

False

>>> cv1.acelera()

>>> print(cv1.velocidad)

60

>>> print(CocheVolador.ruedas)

6

>>> print(c.esta\_volando)

Traceback (most recent call last):

File "<input>", line 1, in <module>

AttributeError: 'Coche' object has no attribute 'esta\_volando'

Encapsulamiento

class A:

def \_\_init\_\_(self):

self.\_contador = 0 # Este atributo es privado

def incrementa(self):

self.\_contador += 1

def cuenta(self):

return self.\_contador

class B(object):

def \_\_init\_\_(self):

self.\_\_contador = 0 # Este atributo es privado

def incrementa(self):

self.\_\_contador += 1

def cuenta(self):

return self.\_\_contador

>>> a = A()

>>> a.incrementa()

>>> a.incrementa()

>>> a.incrementa()

>>> print(a.cuenta())

3

>>> print(a.\_contador)

3

>>> b = B()

>>> b.incrementa()

>>> b.incrementa()

>>> print(b.cuenta())

2

>>> print(b.\_\_contador)

Traceback (most recent call last):

File "<input>", line 1, in <module>

AttributeError: 'B' object has no attribute '\_\_contador'

>>> print(b.\_B\_\_contador)

2

Polimorfismo

class Perro:

def sonido(self):

print('Guauuuuu!!!')

class Gato:

def sonido(self):

print('Miaaauuuu!!!')

class Vaca:

def sonido(self):

print('Múuuuuuuu!!!')

def a\_cantar(animales):

for animal in animales:

animal.sonido()

if \_\_name\_\_ == '\_\_main\_\_':

perro = Perro()

gato = Gato()

gato\_2 = Gato()

vaca = Vaca()

perro\_2 = Perro()

granja = [perro, gato, vaca, gato\_2, perro\_2]

a\_cantar(granja)