Herencia

La clase **punto3d** hereda de la clase punto todos sus propiedades y sus métodos. En la clase hija hemos añadido la propiedad y el setter para el nuevo atributo z, y hemos modificado el constructor (sobreescritura) el método mostrar y el método distancia.

```
class punto3d(punto):
    def __init__(self,x=0,y=0,z=0):
      super().__init__(x,y)
      self.z=z
    @property
    def z(self):
     print("Doy z")
     return self.__z

    @z.setter
    def z(self,z):
     print("Cambio z")
     self.__z=z

    def mostrar(self):
     return super().mostrar()+":"+str(self.__z)

    def distancia(self,otro):
     dx = self.__x - otro.__x
     dy = self.__y - otro.__y
     dz = self.__z - otro.__z
     return (dx*dx + dy*dy + dz*dz)**0.5
```

```
>>> p3d=punto3d(1,2,3)
>>> p3d.x
1
>>> p3d.z
3
>>> p3d.mostrar()
1:2:3
>>> p3d.y = 3
```

La función **super()** me proporciona una referencia a la clase base. Nos permite acceder a los métodos de la clase madre.

Delegación

Llamamos **delegación** a la situación en la que una clase contiene (como atributos) una o más instancias de otra clase, a las que delegará parte de sus funcionalidades.

```
class circulo():
    def __init__(self,centro,radio):
      self.centro=centro
      self.radio=radio

    def mostrar(self):
      return "Centro:{0}-Radio:{1}".format(self.centro.mostrar(),self.radio)
```

```
>>> c1=circulo(punto(2,3),5)
>>> print(c1.mostrar())
Centro:2:3-Radio:5
>>> c1.centro.x = 3
```