Conceptos avanzados de programación orientada a objetos I

Atributos de clase (estáticas)

En Python, las variables definidas dentro de una clase, pero no dentro de un método, son llamadas variables estáticas o de clase. Estas variables son compartidas por todos los objetos de la clase.

Para acceder a una variable de clase podemos hacerlo escribiendo el nombre de la clase o a través de self.

```
>>> class Alumno():
...     contador=0
...     def __init__(self,nombre=""):
...         self.nombre=nombre
...         Alumno.contador+=1
...
>>> a1=Alumno("jose")
>>> a1.contador
1
>>> Alumno.contador
1
```

Usamos las variables estáticas (o de clase) para los atributos que son comunes a todos los atributos de la clase. Los atributos de los objetos se definen en el constructor.

Atributos privados y ocultos

Las variables que comienzan por un guión bajo `_` son consideradas privadas. Su nombre indica a otros programadores que no son públicas: son un detalle de implementación del que no se puede depender — entre otras cosas porque podrían desaparecer en cualquier momento. **Pero nada nos impide acceder a esas variables.**

```
>>> class Alumno():
...     def __init__(self,nombre=""):
...         self.nombre=nombre
...         self._secreto="asdasd"
...
>>> a1=Alumno("jose")
>>> a1.nombre
'jose'
>>> a1._secreto
'asdasd'
```

Dos guiones bajos al comienzo del nombre `__` llevan el ocultamiento un paso más allá, "enmarañando" (name-mangling) la variable de forma que sea más difícil verla desde fuera.

```
>>> class Alumno():
...     def __init__(self,nombre=""):
...         self.nombre=nombre
...         self.__secreto="asdasd"
...
>>> a1=Alumno("jose")
>>> a1.__secreto
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Alumno' object has no attribute '__secreto'
```

Pero en realidad sigue siendo posible acceder a la variable.

```
>>> a1._Alumno__secreto
'asdasd'
```

Se suelen utilizar cuando una subclase define un atributo con el mismo nombre que la clase madre, para que no coincidan los nombres.

Métodos de clase (estáticos)

Los método estáticos (static methods) son aquellos que no necesitan acceso a ningún atributo de ningún objeto en concreto de la clase.

```
>>> class Calculadora():
...   def __init__(self,nombre):
...     self.nombre=nombre
...   def modelo(self):
...     return self.nombre
...   @staticmethod
...   def sumar(x,y):
...     return x+y
...
>>> a=Calculadora("basica")
>>> a.modelo()
'basica'
>>> a.sumar(3,4)
7
>>> Calculadora.sumar(3,4)
7
```

Nada nos impediría mover este método a una función fuera de la clase, ya que no hace uso de ningún atributo de ningún objeto, pero la dejamos dentro porque su lógica (hacer sumas) pertenece conceptualmente a Calculadora.

Lo podemos llamar desde el objeto o desde la clase.

Funciones getattr,setattr,delattr,hasattr

```
>>> a1=Alumno("jose")
>>> getattr(a1,"nombre")
'jose'
>>> getattr(a1,"edad","no tiene")
'no tiene'

>>> setattr(a1,"nombre","pepe")
>>> a1.nombre
'pepe'

>>> hasattr(a1,"nombre")
True

>>> delattr(a1,"nombre")
```