Programación orientada a objetos

Ricardo Pérez López

IES Doñana, curso 2019/2020

Índice general

1. Introducción

1.1. Recapitulación

Recordemos lo que hemos aprendido hasta ahora:

- La abstracción de datos nos permite definir datos complejos mediante las operaciones que los manipulan y de una forma independiente de su implementación.
- Las funciones pueden tener estado interno usando funciones de orden superior y variables no locales.
- Una función puede representar un dato.
- Los datos pueden tener estado interno usando el estado interno de la función que lo representa.
- El paso de mensajes agrupa las operaciones que actúan sobre ese dato dentro de una función que responde a diferentes mensajes despachando a otras funciones dependiendo del mensaje recibido.
- La fución que representa al dato encapsula su estado interno junto con las operaciones que lo manipulan en una sola unidad sintáctica que oculta sus detalles de implementación.

1.2. La metáfora del objeto

Al principio, distinguíamos entre funciones y datos: las funciones realizan operaciones sobre los datos y éstos esperan pasivamente a que se opere con ellos.

Cuando empezamos a representar a los datos con funciones, vimos que los datos también pueden encapsular **comportamiento**.

Esos datos ahora representan información, pero también **se comportan** como las cosas que representan.

Por tanto, los datos ahora saben cómo reaccionar ante los mensajes que reciben cuando las demás partes del programa les envían mensajes.

Esta forma de ver a los datos como objetos activos que se relacionan entre sí y que son capaces de reaccionar y cambiar su estado interno en función de los mensajes que reciben, da lugar a todo un nuevo paradigma de programación llamado **programación orientada a objetos**.

La programación orientada a objetos (OOP) es un método para organizar programas que reúne muchas de las ideas vistas hasta ahora.

Al igual que las funciones en la abstracción de datos, los objetos imponen barreras de abstracción entre el uso y la implementación de datos.

Al igual que los diccionarios de despacho, los objetos responden a solicitudes de comportamiento.

Los objetos tienen un estado interno local al que no se puede acceder directamente desde el entorno global.

El sistema de objetos Python proporciona una sintaxis conveniente para promover el uso de estas técnicas para organizar programas. Gran parte de esta sintaxis se comparte entre otros lenguajes de programación orientados a objetos.

El sistema de objetos ofrece más que solo conveniencia. Permite una nueva metáfora para diseñar programas en los que varios agentes independientes interactúan dentro de la computadora. Cada objeto agrupa el estado local y el comportamiento de una manera que abstrae la complejidad de ambos. Los objetos se comunican entre sí, y los resultados útiles se calculan como consecuencia de su interacción. Los objetos no solo transmiten mensajes, sino que también comparten el comportamiento entre otros objetos del mismo tipo y heredan características de tipos relacionados.

El paradigma de la programación orientada a objetos tiene su propio vocabulario que respalda la metáfora del objeto. Hemos visto que un objeto es un valor de datos que tiene métodos y atributos, accesibles mediante notación de puntos. Cada objeto también tiene un tipo, llamado su clase . Para crear nuevos tipos de datos, implementamos nuevas clases.

Los datos tienen estado interno.

El almacenamiento y acceso al estado interno de una función mediante variables no locales.

La representación de datos como funciones.

La encapsulación de los datos junto con las operaciones que los manipulan en una sola unidad sintáctica.

1.3. Perspectiva histórica

1.4. Lenguajes orientados a objetos

2. Conceptos básicos

2.1. Clase

2.2. Objeto