Modulo 3 (intermedio): Los conceptos básicos del enfoque orientado a objetos

Demos un paso fuera de la programación y las computadoras, y analicemos temas de programación orientada a objetos.

Casi todos los programas y técnicas que has utilizado hasta ahora pertenecen al estilo de programación procedimental. Es cierto que has utilizado algunos objetos incorporados, pero cuando nos referimos a ellos, se mencionan lo mínimo posible.

La programación procedimental fue el enfoque dominante para el desarrollo de software durante décadas de TI, y todavía se usa en la actualidad. Además, no va a desaparecer en el futuro, ya que funciona muy bien para proyectos específicos (en general, no muy complejos y no grandes, pero existen muchas excepciones a esa regla).

El enfoque orientado a objetos es bastante joven (mucho más joven que el enfoque procedimental) y es particularmente útil cuando se aplica a proyectos grandes y complejos llevados a cabo por grandes equipos formados por muchos desarrolladores.

Este tipo de programación en un proyecto facilita muchas tareas importantes, por ejemplo, dividir el proyecto en partes pequeñas e independientes y el desarrollo independiente de diferentes elementos del proyecto.

Python es una herramienta universal para la programación procedimental y orientada a objetos. Se puede utilizar con éxito en ambos enfoques.

Además, puedes crear muchas aplicaciones útiles, incluso si no se sabe nada sobre clases y objetos, pero debes tener en cuenta que algunos de los problemas (por ejemplo, el manejo de la interfaz gráfica de usuario) puede requerir un enfoque estricto de objetos.

Afortunadamente, la programación orientada a objetos es relativamente simple.

Enfoque procedimental frente al enfoque orientado a objetos

En el **enfoque procedimental**, es posible distinguir dos mundos diferentes y completamente separados: **el mundo de los datos y el mundo del código**. El mundo de los datos está poblado con variables de diferentes tipos, mientras que el mundo del código está habitado por códigos agrupados en módulos y funciones.

Las funciones pueden usar datos, pero no al revés. Además, las funciones pueden abusar de los datos, es decir, usar el valor de manera no autorizada (por ejemplo, cuando la función seno recibe el saldo de una cuenta bancaria como parámetro).

Los datos no pueden usar funciones. ¿Pero es esto completamente cierto? ¿Hay algunos tipos especiales de datos que puedan usar funciones?

Sí, los hay, los llamados métodos. Estas son funciones que se invocan desde dentro de los datos, no junto con ellos. Si puedes ver esta distinción, has dado el primer paso en la programación de objetos.

El **enfoque orientado a objetos** sugiere una forma de pensar completamente diferente. Los datos y el código están encapsulados juntos en el mismo mundo, divididos en clases.

Cada **clase es como una receta que se puede usar cuando quieres crear un objeto útil**. Puedes

producir tantos objetos como necesites para resolver tu problema.

Cada objeto tiene un conjunto de rasgos (se denominan propiedades o atributos; usaremos ambas palabras como sinónimos) y es capaz de realizar un conjunto de actividades (que se denominan métodos).

Las recetas pueden modificarse si son inadecuadas para fines específicos y, en efecto, pueden crearse nuevas clases. Estas nuevas clases heredan propiedades y métodos de los originales, y generalmente agregan algunos nuevos, creando nuevas herramientas más específicas.

Los objetos son encarnaciones de las ideas expresadas en clases, como un pastel de queso en tu plato, es una encarnación de la idea expresada en una receta impresa en un viejo libro de cocina.

Los objetos interactúan entre sí, intercambian datos o activan sus métodos. Una clase construida adecuadamente (y, por lo tanto, sus objetos) puede proteger los datos sensibles y ocultarlos de modificaciones no autorizadas.

No existe un límite claro entre los datos y el código: viven como uno solo dentro de los objetos.

Todos estos conceptos no son tan abstractos como pudieras pensar al principio. Por el contrario, todos están tomados de experiencias de la vida real y, por lo tanto, son extremadamente útiles en la programación de computadoras: no crean vida artificial **reflejan hechos reales, relaciones y circunstancias**.

Jerarquías de clase

La palabra *clases* tiene muchos significados, pero no todos son compatibles con las ideas que queremos discutir aquí. La clase que nos concierne es como una categoría, como resultado de similitudes definidas con precisión.

Intentaremos señalar algunas clases que son buenos ejemplos de este concepto.



Veamos por un momento los vehículos. Todos los vehículos existentes (y los que aún no existen) están **relacionados por una sola característica importante**: la capacidad de moverse. Puedes argumentar que un perro también se mueve; ¿Es un perro un vehículo? No lo es. Tenemos que mejorar la definición, es decir, enriquecerla con otros criterios, distinguir los vehículos de otros seres y crear una conexión más fuerte. Consideremos las siguientes circunstancias: los vehículos son entidades creadas artificialmente que se utilizan para el transporte, movidos por fuerzas de la naturaleza y dirigidos (conducidos) por humanos.

Según esta definición, un perro no es un vehículo.

La clase Vehículos es muy amplia. Tenemos que definir clases especializadas. Las clases especializadas son las subclases. La clase Vehículos será una superclase para todas ellas.

Nota: **la jerarquía crece de arriba hacia abajo, como raíces de árboles, no ramas**. La clase más general y más amplia siempre está en la parte superior (la superclase) mientras que sus descendientes se encuentran abajo (las subclases).

A estas alturas, probablemente puedas señalar algunas subclases potenciales para la superclase *Vehículos*. Hay muchas clasificaciones posibles. Elegimos subclases basadas en el medio ambiente y decimos que hay (al menos) cuatro subclases:

- Vehículos Terrestres.
- Vehículos Acuáticos.
- Vehículos Aéreos.

- Vehículos Espaciales.

En este ejemplo, discutiremos solo la primera subclase: Vehículos Terrestres. Si lo deseas, puedes continuar con las clases restantes.

Los vehículos terrestres pueden dividirse aún más, según el método con el que impactan el suelo. Entonces, podemos enumerar:

- Vehículos con ruedas.
- Vehículos oruga.
- Aerodeslizadores.

La figura ilustra la jerarquía que hemos creado.

Ten en cuenta la dirección de las flechas: siempre apuntan a la superclase. La clase de nivel superior es una excepción: no tiene su propia superclase.

Otro ejemplo es la jerarquía del reino taxonómico de los animales.

Podemos decir que todos los Animales (nuestra clase de nivel superior) se puede dividir en cinco subclases:

- Mamíferos.
- Reptiles.
- Aves.
- Peces.
- Anfibios.

Tomaremos el primero para un análisis más detallado.

Hemos identificado las siguientes subclases:

- Mamíferos Salvajes.
- Mamíferos Domesticados.

Intenta extender la jerarquía de la forma que quieras y encuentra el lugar adecuado para los humanos.

¿Qué es un objeto?

Una clase (entre otras definiciones) es un **conjunto de objetos**. Un objeto es **un ser perteneciente a una clase**.

Un objeto es **una encarnación de los requisitos, rasgos y cualidades asignados a una clase específica**. Esto puede sonar simple, pero ten en cuenta las siguientes circunstancias importantes. Las clases forman una jerarquía.

Esto puede significar que un objeto que pertenece a una clase específica pertenece a todas las superclases al mismo tiempo. También puede significar que cualquier objeto perteneciente a una superclase puede no pertenecer a ninguna de sus subclases.

Por ejemplo: cualquier automóvil personal es un objeto que pertenece a la clase Vehículos Terrestres. También significa que el mismo automóvil pertenece a todas las superclases de su clase local; por lo tanto, también es miembro de la clase Vehículos.

Tu perro (o tu gato) es un objeto incluido en la clase Mamíferos Domesticados, lo que significa explícitamente

que también está incluido en la clase Animales.

Cada **subclase es más especializada** (o más específica) que su superclase. Por el contrario, cada **superclase es más general** (más abstracta) que cualquiera de sus subclases.

Ten en cuenta que hemos supuesto que una clase solo puede tener una superclase; esto no siempre es cierto, pero discutiremos este tema más adelante.

Herencia

Definamos uno de los conceptos fundamentales de la programación de objetos, llamado **herencia**. Cualquier objeto vinculado a un nivel específico de una jerarquía de clases **hereda todos los rasgos (así como los requisitos y cualidades) definidos dentro de cualquiera de las superclases**.

La clase de inicio del objeto puede definir nuevos rasgos (así como requisitos y cualidades) que serán heredados por cualquiera de sus superclases.



No deberías tener ningún problema para hacer coincidir esta regla con ejemplos específicos, ya sea que se aplique a animales o vehículos.

¿Qué contiene un objeto?

La programación orientada a objetos supone que **cada objeto existente puede estar equipado con tres grupos de atributos**:

- Un objeto tiene un **nombre** que lo identifica de forma exclusiva dentro de su namespace (aunque también puede haber algunos objetos anónimos).
- Un objeto tiene un **conjunto de propiedades individuales** que lo hacen original, único o sobresaliente (aunque es posible que algunos objetos no tengan propiedades).
- Un objeto tiene un **conjunto de habilidades para realizar actividades específicas**, capaz de cambiar el objeto en sí, o algunos de los otros objetos.

Existe una pista (aunque esto no siempre funciona) que te puede ayudar a identificar cualquiera de las tres esferas anteriores. Cada vez que se describe un objeto y se usa:

- Un sustantivo: probablemente se está definiendo el nombre del objeto.
- Un adjetivo: probablemente se está definiendo una propiedad del objeto.
- Un verbo: probablemente se está definiendo una actividad del objeto.

Dos ejemplos deberían servir como un buen ejemplo:

- Un Cadillac rosa pasó rápidamente.
    - Nombre del objeto = Cadillac
    - Clase = Vehículos con ruedas
    - Propiedad = Color (rosa)
    - Actividad = Pasar (rápidamente)
- Max es un gato grande que duerme todo el día.
    - Nombre del objeto = Max
    - Clase = Gato
    - Propiedad = Tamaño (Grande)

- Actividad = Dormir (Todo el día)



Tu primera clase

La programación orientada a objetos es **el arte de definir y expandir clases**. Una clase es un modelo de una parte muy específica de la realidad, que refleja las propiedades y actividades que se encuentran en el mundo real.

Las clases definidas al principio son demasiado generales e imprecisas para cubrir el mayor número posible de casos reales.

No hay obstáculo para definir nuevas subclases más precisas. Heredarán todo de su superclase, por lo que el trabajo que se utilizó para su creación no se desperdicia.

La nueva clase puede agregar nuevas propiedades y nuevas actividades y, por lo tanto, puede ser más útil en aplicaciones específicas. Obviamente, se puede usar como una superclase para cualquier número de subclases recién creadas.

El proceso no necesita tener un final. Puedes crear tantas clases como necesites.

La clase que se define no tiene nada que ver con el objeto: **la existencia de una clase no significa que ninguno de los objetos compatibles se creará automáticamente**. La clase en sí misma no puede crear un objeto: debes crearlo tu mismo y Python te permite hacerlo.

Es hora de definir la clase más simple y crear un objeto. Analiza el siguiente ejemplo:

```
<code python>class TheSimplestClass:
    pass
```

</code>

Hemos definido una clase. La clase es bastante pobre: no contiene propiedades ni actividades. Esta **vacía**, pero eso no importa por ahora. Cuanto más simple sea la clase, mejor para nuestros propósitos.

La definición comienza con la palabra clave reservada *class*. La palabra clave reservada es seguida por **un identificador que le dará nombre a la clase** (nota: no lo confundas con el nombre del objeto: estas son dos cosas diferentes).

A continuación, se agregan **dos puntos** (:), como clases, como funciones, forman su propio bloque anidado. El contenido dentro del bloque define todas las propiedades y actividades de la clase.

La palabra clave reservada pass llena la clase con nada. No contiene ningún método ni propiedades.

Tu primer objeto

La clase recién definida se convierte en una herramienta que puede crear nuevos objetos. La herramienta debe usarse explícitamente, bajo demanda.

Imagina que deseas crear un objeto (exactamente uno) de la clase TheSimplestClass.

Para hacer esto, debes asignar una variable para almacenar el objeto recién creado de esa clase y crear un objeto al mismo tiempo.

Se hace de la siguiente manera:

```
my_first_object = TheSimplestClass()
```

Nota:

- El nombre de la clase intenta fingir que es una función, ¿puedes ver esto? Lo discutiremos pronto.
- El objeto recién creado está equipado con todo lo que trae la clase. Como esta clase está completamente vacía, el objeto también está vacío.

El acto de crear un objeto de la clase seleccionada también se llama **instanciación** (ya que el objeto se convierte en una **instancia de la clase**).

Dejemos las clases en paz por un breve momento, ya que ahora diremos algunas palabras sobre *pilas*. Sabemos que el concepto de clases y objetos puede no estar completamente claro todavía. No te preocupes, te explicaremos todo muy pronto.

Puntos Clave

1. Una clase es una idea (más o menos abstracta) que se puede utilizar para crear varias encarnaciones; una encarnación de este tipo se denomina objeto.

2. Cuando una clase se deriva de otra clase, su relación se denomina herencia. La clase que deriva de la otra clase se denomina subclase. El segundo lado de esta relación se denomina superclase. Una forma de presentar dicha relación es en un diagrama de herencia, donde:

- Las superclases siempre se presentan encima de sus subclases.
- Las relaciones entre clases se muestran como flechas dirigidas desde la subclase hacia su superclase.

3. Los objetos están equipados con:

- Un nombre que los identifica y nos permite distinguirlos.
- Un conjunto de propiedades (el conjunto puede estar vacío).
- Un conjunto de métodos (también puede estar vacío).

4. Para definir una clase de Python,se necesita usar la palabra clave reservada class. Por ejemplo:

```
class This_Is_A_Class:
    pass
```

5. Para crear un objeto de la clase previamente definida, se necesita usar la clase como si fuera una función. Por ejemplo:

```
this_is_an_object = This_Is_A_Class()
```

From:
https://matebcn.eu/ - **miguel angel torres egea**

Permanent link:
https://matebcn.eu/info:cursos:netacad:python:pe2m3:fundamentosoop

Last update: **05/07/2022 21:20**