

Clase 3 Parte 2: Programación

Programación Orientada a Objetos.

Temario

- Conceptos de la Programación Orientada a Objetos
 - Definición de Objetos de software
 - Estado y comportamiento
 - Mensajes
 - Características

Programación Orientada a Objetos en Python

- Definición de clases
- Definición de atributos
- Definición e Invocación de métodos

PROGRAMACIÓN ORIENTADA A OBJETOS

Dado un problema debemos:



PROGRAMACIÓN ORIENTADA A OBJETOS

La **Programación Orientada a Objetos (POO)** se define como una manera de programar específica, donde se organiza el código en unidades denominadas **clases**, de las cuales se crean **objetos** que se relacionan entre sí para conseguir los objetivos de las aplicaciones.

Con la POO se resuelven los problemas escribiendo programas en términos de **clases**, **objetos**, **propiedades**, **métodos**, **diagrama de clases** entre otros conceptos. Ya no se escribe el código como se solía hacer en la Programación Estructurada.

PROGRAMACIÓN ORIENTADA A OBJETOS

El objetivo de la POO es **simplificar la complejidad del problema abstrayendo** su conocimiento y comportamiento, encapsulando en **OBJETOS**.

En POO se **'personifican' los objetos físicos, dándoles las características y la funcionalidad** que ellos tienen en el mundo real.

Un **programa** se describe en término de los **objetos involucrados**. Dichos objetos **interactúan entre sí** intercambiando **mensajes** para llevar a cabo una **tarea**.

PARADIGMA ORIENTADO A OBJETOS

Cuando debemos **realizar un sistema** lo primero que debemos hacer es **identificar los objetos** involucrados en el sistema.

Es importante **quedarnos** con aquellos **objetos** que son solamente **de nuestro interés**.

¿Que son los Objetos?

Cada **objeto** abstrae un dato del problema y lo que puede hacerse sobre él.

Los sustantivos son un buen punto de partida para **determinar los objetos** de un sistema.

Objeto: es una entidad que **tiene 2 características:**
estado y comportamiento.

Describiendo a los Objetos

Estado: está representado por los *atributos*, es decir las **propiedades de un objeto**. Los atributos de un objeto se almacenan en *variables de instancia*.

Comportamiento: está representado por una *serie de funciones o métodos* que modifican o no el estado del objeto.

A la **representación de un objeto de la vida real** en un programa se lo denomina **objeto de software**.

Describiendo Objetos

Ejemplo:

Un **perro** tiene
estado: nombre, raza, color
comportamiento: ladrar, jugar, comer, etc.



Composición de objetos

Un objeto puede componerse de dos o más objetos,
conformando así un **objeto compuesto**.



Un curso está formado por alumnos.



Mensajes. Métodos

Los objetos **se comunican** enviándose **mensajes**.

Supongamos que Pablo quiere hacer un pedido de helado para que se lo entreguen en su domicilio.

Pablo, llama a la heladería. Lo atiende la telefonista, Silvina.

Pablo solicita el pedido, indicándole la cantidad, gustos que desea y el domicilio donde debe enviarse el pedido.



Mensaje con el requerimiento

Telefonista
, agente
apropiado



Silvina, la telefonista tiene la
responsabilidad de satisfacer
el requerimiento.

Mensajes. Métodos

Los objetos **se comunican** enviándose **mensajes**.

- La resolución del problema, puede requerir la ayuda de otros individuos.



emisor

mensaje



receptor

mensaje



mensaje

mensaje



Cada **objeto** cumple un **ejecuta rol**, una **acción**, que es usada por otros miembros de la comunidad.

El **emisor** envía un mensaje al **receptor**, junto con los argumentos necesarios para llevar a cabo el requerimiento.

El **receptor** es el objeto a quien se le envía el mensaje.

El **receptor** en respuesta al mensaje **ejecutará** un **método** para satisfacer el requerimiento.

Clases e instancias

Una **clase** es un **molde o modelo** a partir del cual se crean **instancias** (objetos).

Los objetos creados a partir de la misma clase **comparten características y comportamiento**.

Una clase **agrupa todos los objetos** que tienen **las mismas propiedades y funcionalidades**.

Clases e instancias

- ¿Cuáles son los atributos/estado y el comportamiento que tienen en común todos los autos?

Clase AUTO



Modelo

Marca

Color

Velocidad

Acelerar

Desacelerar

Apagar

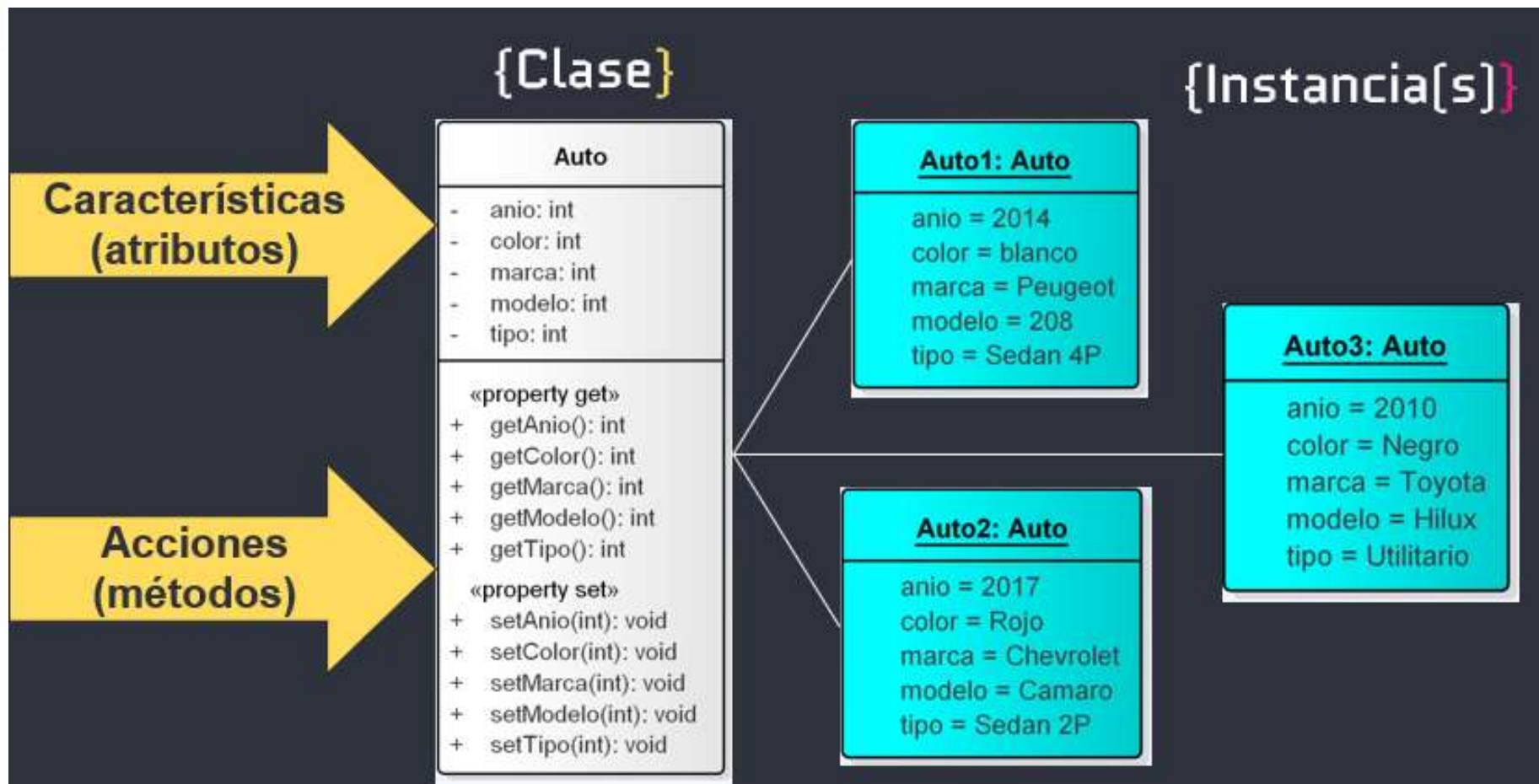
Arrancar

Atributos/Estado

Comportamiento

Clases e instancias

- ¿Cuáles son los atributos/estado y el comportamiento que tienen en común todos los autos?



Clases e instancias

Una **instancia** es un objeto en particular de una clase. Todas las instancias de una clase responden al mismo comportamiento.



Roco es una instancia de la clase Perro

Ana, Abril y Belén son instancias de la clase Alumno



Silvia es una instancia de la clase Cliente

El **auto** de Andrés es una instancia de la clase Auto



El **Banco** donde pago mis cuentas es una instancia de la clase Banco

Agustina es una instancia de la clase Empleado



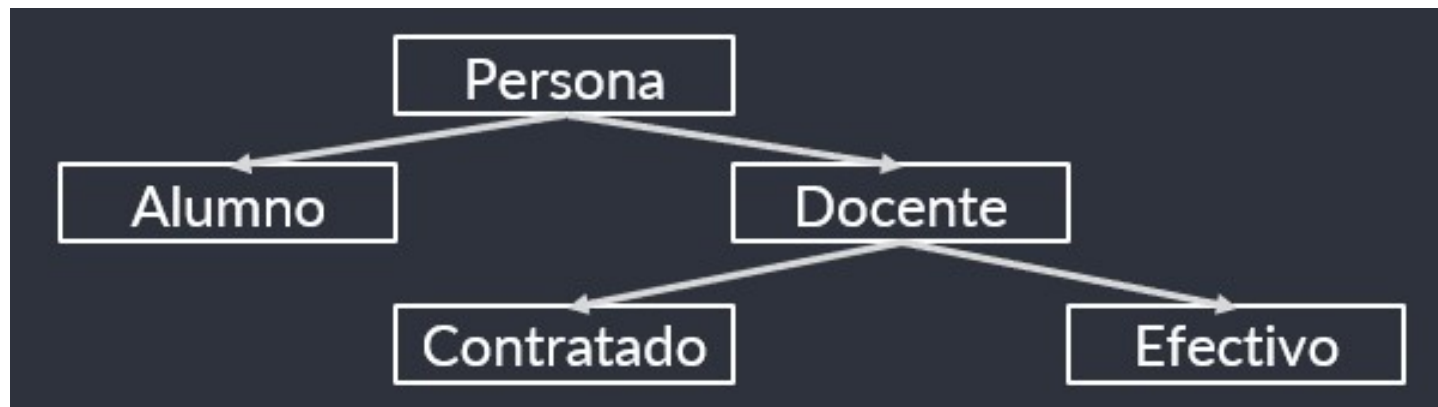
Los cuatro pilares de la POO.



Herencia

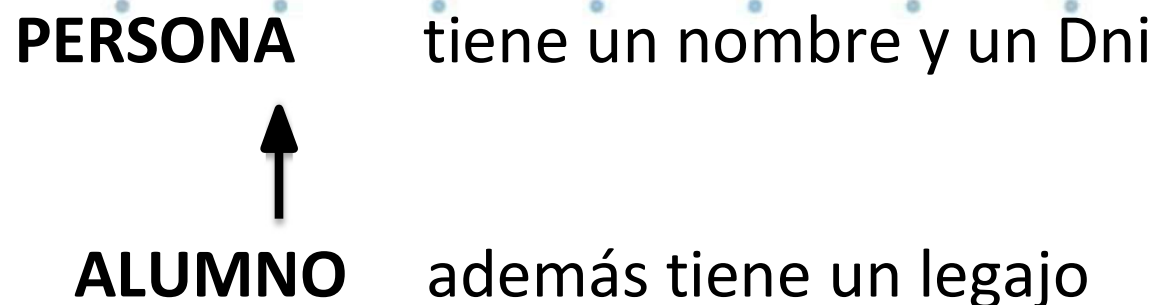
Las clases están organizadas en una **jerarquía de clases**, donde las **subclases heredan atributos y/o comportamiento de las clases padres**.

Las subclases pueden agregar nuevos atributos, o solo cambiar el comportamiento de la clase padre.



Jerarquía de clases

Así como todo **objeto** es una instancia de una clase, toda **clase** es **subclase** de otra clase más general. Cuando se crea una nueva clase debe “colgarse” de otra clase ya existente en el entorno. De esta forma se arma un **árbol jerárquico de clases**, donde **Object** es la clase raíz.

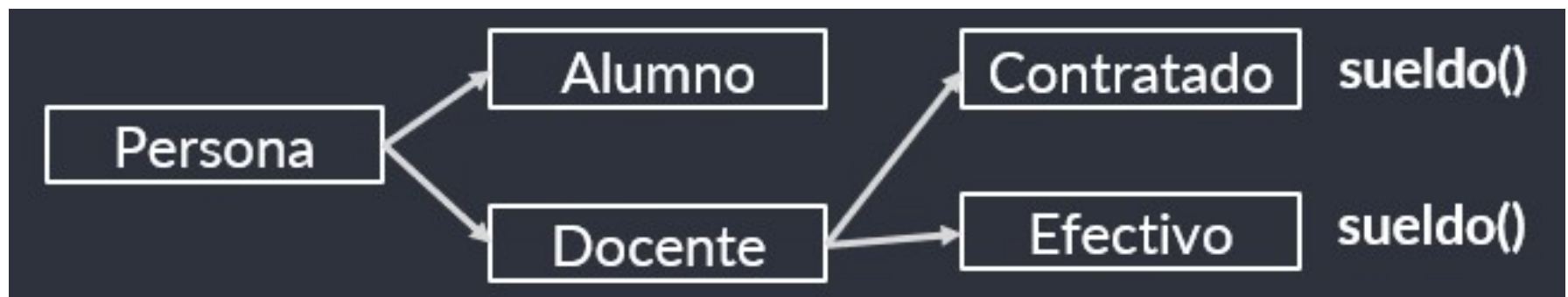


La clase Alumno es **subclase** de Persona. Es más detallada. Persona es la **superclase** de Alumno. Es más general.

Polimorfismo

Polimorfismo: Es la capacidad que tienen los objetos de **tener métodos con el mismo nombre y comportamientos diferentes**. La interpretación del método será según el objeto receptor.

Binding dinámico: significa que la ligadura entre el objeto receptor y el método que se va a ejecutar se realiza en tiempo de ejecución



Abstracción

La abstracción viene de la etapa donde analizamos los atributos y el comportamiento que las entidades deben tener.

Es la propiedad por la cual las entidades quedan representadas únicamente por los datos de interés para el problema, y no por todos los atributos que una entidad pueda tener en la vida real.

Por ejemplo: omitir color de ojos, altura, peso y talla de zapatillas de una persona para un sistema de gestión universitario.

Encapsulamiento y ocultamiento

Toda entidad queda comprendida dentro de un paquete compuesto de estado y comportamiento.

Este paquete debe poder protegerse por lo cual tiene propiedades de ocultamiento.

Tenemos datos públicos, protegidos o privados y métodos por medio de los cuales operarlos en base a la necesidad.

Encapsulamiento y ocultamiento

El **encapsulamiento** es un principio de la POO que **protege los datos internos de un objeto**.

- La clase **oculta sus atributos** y expone solo lo necesario mediante **métodos**.
- Los objetos pueden **usar y manipular datos de otros objetos** solo a través de esos métodos, sin conocer los detalles internos.

Encapsulamiento y ocultamiento.

Ocultamiento de la información:

Tanto **la representación interna de los objetos** como **la implementación de los métodos** queda oculta para el usuario.

Para **consultar o modificar el valor almacenado en las variables** de instancia de un objeto ***solo puede hacerse mediante los métodos*** definidos para tal fin.



Programación Orientada a Objetos con Python

PОО en Python

- Python está completamente preparado para soportar la programación orientada a objetos.
- En Python cuando se utiliza la programación orientada a objetos se puede:
 - Definir nuevas clases.
 - Utilizar mecanismos para definir relaciones de herencia entre clases.
 - Instanciar las clases ya definidas en el sistema o las que haya definido el desarrollador.

Clases en Python

- Una clase de Python empieza con la palabra reservada **class**, seguida de su nombre y finalizando con **:**
 - **class Auto:**
 - **class Persona:**
- Por convención los nombres de clases deben comenzar en mayúscula.

Clases en Python

- Dentro de la clase se especifican los métodos y los atributos comunes a todos los objetos de dicha clase.
- Todo lo que va dentro de una clase se escribe con sangría o indentación.
- Las clases pueden ser definidas en un módulo independiente o bien dentro del módulo de la aplicación que va a utilizar dicha clase. En este último caso, la primera sentencia que no esté con sangría, no pertenece a la clase.

Clase Libro en Python

- Definamos la clase **Libro**

```
class Libro:
    def __init__(self, unNombre, unAutor, unPrecio):
        self.__nombre = unNombre
        self.__autor = unAutor
        self.__precio = unPrecio

    def getNombre(self):
        return self.__nombre

    def setNombre(self, nombre_libro):
        self.__nombre = nombre_libro

    def __str__(self):
        return "Nombre: " + self.getNombre()
```

Nombre de la clase

Cuerpo de la
clase Libro

Variables de
instancia
(ATRIBUTOS!!!)

Métodos
COMPOR-
TAMIENTO

Instancias en Python

- Para crear un objeto hay que instanciarlo a partir de una clase. Para ello se utiliza la notación de funciones, o sea se escribe el nombre de la clase seguido por dos paréntesis, uno de apertura y otro de cierre.

```
libro1 = Libro("Los 7 Locos", "Roberto Arlt",1200)
libro2 = Libro("Los Lanzallamas", "Roberto Arlt",1300)
libro3 = Libro("El Juguete Rabioso", "Roberto Arlt",1500)
```

Creación de tres instancias de la clase Libro. Se invoca en forma automática al constructor. Se crean 3 objetos "Libro"

Constructores

- Los constructores son métodos especiales mediante los cuales los programadores pueden inicializar una instancia. Pueden recibir o no parámetros. Si tienen parámetros, dichos valores se usan para inicializar las variables de instancia del objeto creado.
- Los constructores se ejecutan automáticamente justo después de crear o instanciar un objeto.
- Los constructores en Python se definen codificando un método especial llamado **init**. (lleva doble guión)
- En el **__init__** se inicializan las variables de instancia del objeto, dejándolas disponibles para comenzar a operar con ellas a través de los métodos.

Constructores

- Ejemplo de la utilización de un constructor para inicializar una variable de instancia en los objetos Libro .

```
class Libro:  
    def __init__(self, unNombre, unAutor, unPrecio):  
        self.__nombre = unNombre  
        self.__autor = unAutor  
        self.__precio = unPrecio
```

Constructor de la clase Libro

**Inicializa el atributo
precio con el valor
contenido en el
parámetro unPrecio.**

Self

- La palabra reservada **self** sirve para referirse al objeto actual, que es el que recibe el mensaje. Es utilizada dentro del método para señalarse a sí mismo y sirve para poder acceder a los atributos y métodos del objeto actual.

En todos los métodos de una clase el primer parámetro es siempre self.

```
class Libro:

    def __init__(self, unNombre, unAutor, unPrecio):
        self.nombre = unNombre
        self.autor = unAutor
        self.precio = unPrecio
```

Nos permite especificar que nombre, autor y precio son las variables de instancia.

Atributos en Python

- Los ***atributos o variables de instancia*** permiten almacenar el estado de un objeto.
- Para asignar o acceder a la información que se almacena en ellos se utiliza la notación de punto.

`return self.__nombre`

Notación de Punto


Atributo que almacena el nombre del libro

- ❖ Descriptor de acceso **get**: se usa para consultar y retornar el valor de un campo privado → (getter)
- ❖ Descriptor de acceso **set**: se usa para asignar un nuevo valor a un campo privado → (setter)

Atributos en Python

```
class Libro:
```

Referencia al
objeto actual



```
def __init__(self, unNombre, unAutor, unPrecio):
```

```
    self.__nombre = unNombre
```


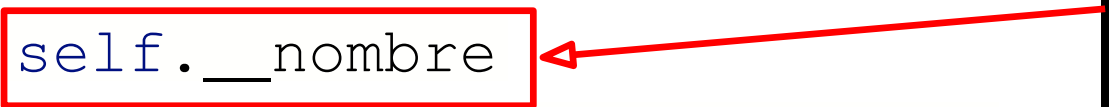
```
    self.__autor =
```

```
    self.__precio =
```

```
def getNombre(self):
```

```
    return self.__nombre
```

Referencia al
atributo o
variable de
instancia



```
def setNombre(self, nombre_libro):
```

```
    self.__nombre = nombre_libro
```

Declaración de Métodos

- Los métodos en Python son muy similares a las funciones.
- Las similitudes más significativas son:
 - Para definir un método:
def nombreMetodo(parámetros):
 - El cuerpo debe estar indentado.
 - Pueden o no retornar un resultado.
- Pero se diferencian de las funciones en que:
 - Los métodos se encuentran dentro de una definición de clase.
 - El primer parámetro siempre debe ser **self** aunque no se lo utilice luego en el método.

Declaración de Métodos

```
class Libro:
    def __init__(self, unNombre, unAutor, unPrecio):
        self.nombre = unNombre
        self.autor = unAutor
        self.precio = unPrecio

    def getNombre(self):
        return self.nombre

    def setNombre(self, nombre_libro):
        self.nombre = nombre_libro

    def precioEnDolares(self, valor_dolar):
        return self.precio / valor_dolar

    def __str__(self):
        return "Nombre: " + self.getNombre()
```

Invocación de Métodos

La forma en que los métodos son invocados difiere de la forma de invocar a las funciones. Se debe utilizar la notación de punto.

- Al invocar un método el parámetro **self** se omite ya que está implícito en el objeto.

```
libro1 = Libro("Los 7 Locos",  
"Roberto Arlt", 1200)
```

```
libro1.precioEnDolares(500
```

```
libro1.getNombre(  
)
```

Nombre del
método
invocado

