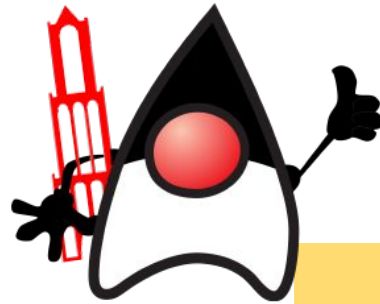




Curso FullStack Python

Codo a Codo 4.0



Python

Parte 5



Temas que veremos las próximas clases

P00

Objetos

Clases

Abstracción

Encapsulamiento

Polimorfismo

Herencia

Programación orientada a objetos (POO)

Metodologías de programación

Prácticamente todos los lenguajes desarrollados en los últimos 25 años implementan la posibilidad de trabajar con POO (Programación Orientada a Objetos).

Python permite programar con las siguientes metodologías:

- **Programación Lineal:** El código es desarrollado en una *secuencia lineal*, sin emplear funciones. Es apenas modificada por las bifurcaciones (condicionales) o repeticiones.
- **Programación Estructurada:** El código es desarrollado *modularmente a través de funciones* que agrupan actividades a desarrollar y luego son llamadas dentro del programa principal. Estas funciones pueden estar dentro del mismo archivo (módulo) o en una librería separada (funciones dentro de módulos que podemos importar).
- **Programación Orientada a Objetos:** Aplica la metodología de la programación orientada a objetos que tiene una sintaxis particular, donde se plantean *clases* y definen *objetos*.

Por ejemplo: Al hablar de un animal no estamos hablando de un objeto, no es un objeto hasta que no “exista”, tenga características o un comportamiento. A eso lo vamos a llamar clase, que es el “molde” para construir el objeto. También podemos hacer referencia a objetos intangibles (por ejemplo: una cuenta bancaria).

Programación orientada a objetos (POO)

El paradigma orientado a objetos

Un paradigma de programación es un estilo de desarrollo de programas, un modelo para resolver problemas computacionales o, mejor dicho, una forma distinta de pensar la programación. Los lenguajes de programación, necesariamente, se encuadran en uno o varios paradigmas a la vez a partir del tipo de órdenes que permiten implementar, algo que tiene una relación directa con su sintaxis.

- En el ***paradigma orientado a objetos*** el comportamiento del programa es llevado a cabo por objetos, entidades que representan elementos del problema a resolver y tienen atributos y comportamiento (pueden almacenar información y realizar acciones).
- Hace que el desarrollo de grandes proyectos de software sea más fácil y más intuitivo.
- Nos permite pensar sobre el software en términos de objetos del mundo real y sus relaciones. Nos acercamos más a la realidad, dado que trabajamos con elementos de la vida real.

En el caso de la programación el POO es de mediados de los 70 y su gran auge es a mediados de los 90. El lenguaje que vino a destacarse en este nuevo paradigma es JAVA (por excelencia orientado a objetos).

El concepto de objeto excede JAVA, Python, etc, aplica a varios lenguajes.

Objetos y clases

Clase

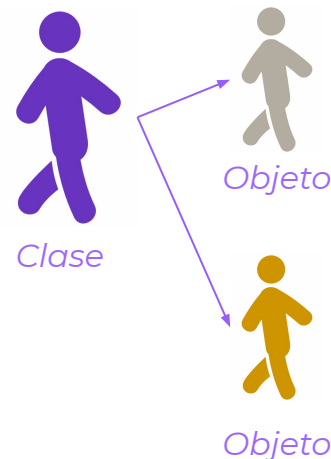
La programación orientada a objetos se basa en la definición de **clases**; a diferencia de la programación estructurada, que está centrada en las **funciones**.

Una clase es un **molde** del que luego se pueden crear múltiples objetos, con similares características. Esta plantilla o molde define los atributos (*variables*) y métodos (*funciones*) comunes a los objetos de ese tipo, pero luego, cada objeto tendrá sus propios valores y compartirán las mismas funciones.

Dado que las clases se usan para crear objetos, debemos declararlas antes de poder crear objetos (instancias) de esa clase. Al crear un objeto de una clase, se dice que se crea una **instancia de la clase** o un objeto propiamente dicho.

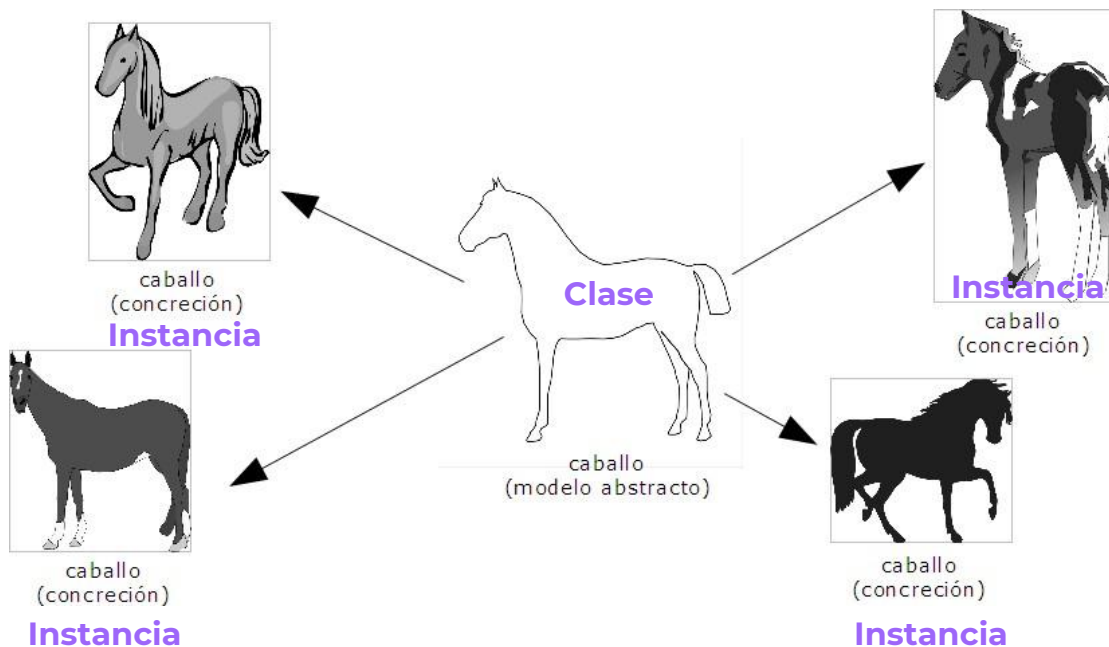
Una clase está formada por los **métodos** y las **variables** (atributos) que definen las características comunes a todos los objetos de esa clase. Precisamente la clave de la POO está en abstraer los métodos y los datos comunes a un conjunto de objetos y almacenarlos en una **clase**.

Una clase equivale a la **generalización de un tipo específico de objetos**. Una **instancia** es la concreción de una clase en un objeto. Las clases definen el tipo de datos (*type*).



Podemos crear muchos objetos desde una sola clase

Objetos y clases



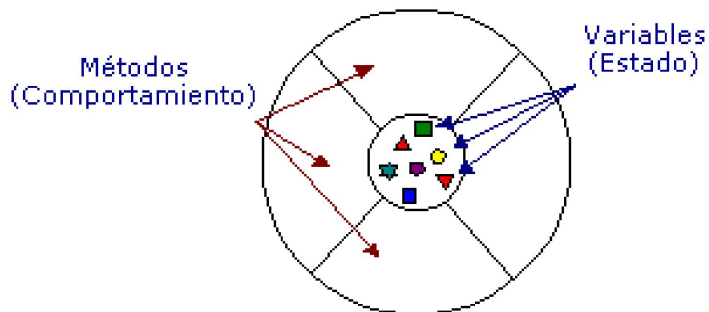
Cada uno de los objetos tiene su propia copia de las variables definidas en la clase de la cual son instanciados y comparten la misma implementación de los métodos.

No se puede crear un objeto sin previamente haber creado o definido la clase, porque la clase es el molde para ese objeto.

Objetos y clases

Objeto

- Es una **encapsulación genérica de datos** y de los procedimientos (*funciones*) para manipularlos. Debemos pensarlos como objetos del mundo real.
- Tienen un **estado** y un **comportamiento**: El **estado** de los objetos se determina a partir de una o más *variables (valores del atributo)* y el **comportamiento** con la implementación de *métodos*.



Representación común de los objetos de software

*Se habla de **encapsulación** porque esa información asociada al objeto podría ser vulnerable, son datos sensibles para el objeto. Ejemplo: si cambio el saldo de la cuenta del objeto "cliente", cambio información sensible.*

Objetos y clases

EJEMPLO DE CLASES Y OBJETOS

Clase:

Coche

Clase Coche

arrancar, ir, parar, girar

color, velocidad, carburante

nombre de la clase

métodos (funciones)

atributos (datos)

Objeto:

Ferrari

coche.ferrari

nombre del objeto

métodos

arrancar, ir, parar, girar

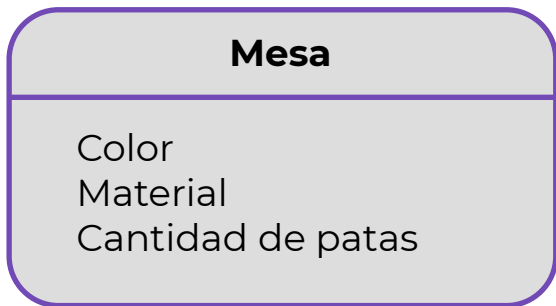
datos

rojo, 280 km/h, Lleno

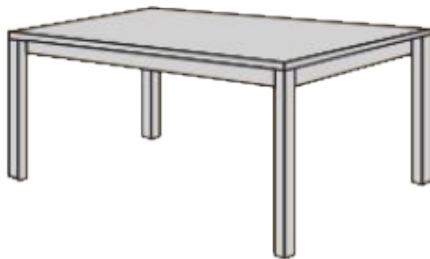


Objetos y clases

Clase



Objeto 1



Objeto 2



Un objeto es una entidad en un programa, usualmente un sustantivo.

Por ejemplo: una persona en particular.

- **Clase:** Persona;
- **Propiedades/Atributos:** Nombre; Edad; Dirección;
- **Comportamiento/Métodos:** Caminar; Hablar; Respirar;
- **Objeto:** José Pérez de 23 años que vive en la calle Cucha cucha 123;



Otro ejemplo: Crear una nueva variable denominada `mi_nombre` con el valor de “Matias”. Esta variable es en realidad una referencia a un objeto. El tipo de objeto es `str` porque para poder crearla, necesitamos instanciar desde la clase `str` `mi_nombre = “Matias”`

Objetos y clases

Otros conceptos relacionados con clases y objetos

- **Atributos:** Son los *datos* que caracterizan al objeto. Son **variables** que almacenan datos relacionados al estado de un objeto.
- **Métodos (o funciones de miembro):** Son los que caracterizan su **comportamiento**, es decir, son todas las **acciones** (denominadas *operaciones*) que el objeto puede realizar por sí mismo. Estas operaciones hacen posible que el objeto responda a las solicitudes externas (o que actúe sobre otros objetos). Además, las operaciones están estrechamente ligadas a los atributos, ya que sus acciones pueden depender de, o modificar, los valores de un atributo.
- **Identidad:** El objeto tiene una *identidad*, que lo distingue de otros objetos, sin considerar su estado. Por lo general, esta identidad se crea mediante un **identificador** que deriva naturalmente de un problema (por ejemplo: un producto puede estar representado por un código, un automóvil por un número de modelo, etc.).

Mensajes y métodos

El modelado de objetos no sólo tiene en consideración los objetos de un sistema, sino también sus interrelaciones, es decir la interacción entre ellos.

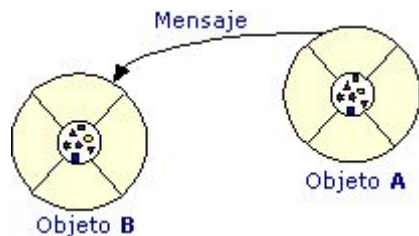
Mensaje

Los objetos interactúan enviándose mensajes unos a otros. Tras la recepción de un mensaje el objeto actuará. La acción puede ser el envío de otros mensajes, el cambio de su estado, o la ejecución de cualquier otra tarea que se requiera que haga el objeto.

Método

Un método se implementa en una clase, y determina cómo tiene que actuar el objeto cuando recibe un mensaje. El método es una acción que va a llevar adelante el objeto a través de la llamada de otro objeto.

Cuando un objeto A necesita que el objeto B ejecute alguno de sus métodos, el objeto A le manda un mensaje al objeto B.



Al recibir el mensaje del objeto A, el objeto B ejecutará el método adecuado para el mensaje recibido.

Objetos y clases

¿Cómo declaramos una clase y creamos objetos?

Para definir una clase conviene buscar un nombre de clase lo más próximo a lo que representa. La definimos con la palabra clave ***class***, seguidamente del nombre de la clase y dos puntos.

Problema 1:

Implementar una clase llamada Persona que tendrá como atributo (variable) su nombre y dos métodos (funciones), uno de dichos métodos inicializará el atributo nombre y el siguiente método mostrará en la pantalla el contenido del mismo. Definir dos objetos de la clase Persona e incorporar una variable de clase (piernas).

Datos:

Clase: Persona

Variable: Nombre

Métodos: Inicializar e imprimir

Variable de clase: Piernas (2)

Objetos de la clase: 2



```
class Persona: #Creamos la clase
```

Esta clase tendrá como atributo (variable) su nombre y dos métodos (funciones), uno de dichos métodos inicializará el atributo nombre y el siguiente método mostrará en la pantalla el contenido del mismo. Además, definiremos un atributo de la clase llamado “piernas”.

```
piernas=2 #Variable de clase
```

Los métodos de una clase se definen utilizando la misma sintaxis que para la definición de funciones. Dentro del método diferenciamos los atributos del objeto antecediendo el identificador **self**.

```
def inicializar(self,nombre): #Constructor
    self.nombre=nombre

def imprimir(self): #Método para mostrar datos
    print("Nombre: {}".format(self.nombre))
```

*Todo método tiene como primer parámetro el identificador **self** que tiene la referencia del objeto que llamó al método.*

Con la asignación previa almacenamos en el atributo **nombre** el parámetro nom, los atributos siguen existiendo cuando finaliza la ejecución del método. Por ello cuando se ejecuta el método imprimir podemos mostrar el nombre que cargamos en el primer método.

Recordemos que una clase es un molde que nos permite definir objetos. Crearemos dos objetos de la clase Persona:

```
persona1=Persona()  
persona2=Persona()
```

Definimos un objeto llamado **persona1** y lo creamos asignándole el nombre de la clase con paréntesis abierto y cerrado al final (como cuando llamamos a una función).

Luego llamaremos a los métodos (funciones), para ello debemos disponer luego del nombre del objeto el operador . (punto) y por último el nombre del método (función).

En el caso que tenga parámetros se los enviamos (salvo el primer parámetro (*self*) que el mismo Python se encarga de enviar la referencia del objeto que se creó):

```
persona1.inicializar("Pedro") #Llamamos al constructor con un nombre  
persona1.imprimir() #Mostramos los datos
```

También podemos llamar a los métodos para el otro objeto creado:

```
persona2.inicializar("Carla")  
persona2.imprimir()
```



Ejercicio_1_POO.py

La declaración de clases es una de las ventajas fundamentales de la POO, ya que la reutilización de código (gracias a que está encapsulada en clases) es muy sencilla.

Objetos y clases

¿Cómo declaramos una clase y creamos objetos?

Problema 2:

*Implementar una clase llamada Alumno que tenga como atributos su nombre y su nota. Definir los métodos para inicializar sus atributos, imprimirlos y mostrar un mensaje si su estado es “regular” (nota mayor o igual a 4).
Definir dos objetos de la clase Alumno.*

Datos:

Clase: Alumno

Variables: Nombre y nota

Métodos: Inicializar, imprimir y mostrar_estado

Objetos de la clase: 2

Declaramos la clase Alumno y definimos sus tres métodos, en el método inicializar llegan como parámetros aparte del self el nombre y nota del alumno:

```
class Alumno: #Creamos la clase

    def inicializar(self,nombre,nota): #Constructor
        self.nombre=nombre
        self.nota=nota

    def imprimir(self): #Método para mostrar los datos
        print("Nombre: {}".format(self.nombre))
        print("Nota: {}".format(self.nota))

    def mostrar_estado(self): #Método para ver si está aprobado
        if self.nota>=4:
            print("Regular")
        else:
            print("Desaprobado")
```

No hay problema que los atributos se llamen iguales a los parámetros ya que siempre hay que anteceder la palabra "self" al nombre del atributo: **self.nombre=nombre**

Tener en cuenta que cuando se crean los atributos en el método inicializar luego podemos acceder a los mismos en los otros métodos de la clase, por ejemplo, en el método **mostrar_estado** verificamos el valor almacenado en el atributo nota, creado dentro del método constructor inicializar.

Habíamos dicho que una clase es un **molde** que nos permite crear luego objetos de dicha clase, en este problema el molde **Alumno** lo utilizamos para crear dos objetos de dicha clase:

Obj 1

```
alumno1=Alumno() #Creamos el objeto
alumno1.inicializar("Diego",2) #Le damos 2 atributos (nombre y nota)
alumno1.imprimir() #Imprimimos los datos
alumno1.mostrar_estado() #Vemos si está aprobado
```

Obj 2

```
alumno2=Alumno()
alumno2.inicializar("Ana",10)
alumno2.imprimir()
alumno2.mostrar_estado()
```

Es fundamental la definición de objetos de una clase para que tenga sentido la declaración de dicha clase.

Nombre: Diego

Nota: 2

Desaprobado

Nombre: Ana

Nota: 10

Regular

terminal



Ejercicio_2_POO.py

Objetos y clases

Atributos de instancia

Dijimos que cuando creamos una clase estamos generando una *plantilla* o *molde* para un nuevo tipo de objeto que puede almacenar información y realizar acciones. Por ejemplo podemos crear la clase “Perro” con **class Perro** (se sugiere la notación CamelCase para los nombres de las clases):

- Agreguemos propiedades que todos los perros deberían tener (nombre, edad).
- Las propiedades de todos los objetos de tipo Perro deben definirse en un método denominado **__init__()**
 - **__init__()** define el estado inicial del objeto asignando valores a las propiedades del objeto.
 - Esto significa que **__init__()** inicializa *a cada nueva instancia* de la clase.
 - Le podemos pasar la cantidad de parámetros deseada, siempre y cuando el primero sea **self**.
- Los métodos se declaran de manera similar a las funciones con **def** como primer elemento.

```
class Perro:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad
```

Objetos y clases

Atributos de clase

Los atributos de clase son atributos que tienen el mismo valor **para todas las instancias** (en este caso, para todos los objetos de tipo *perro*).

Estos atributos de clase pueden definirse por fuera del método `__init__()` y siempre se encontrarán directamente debajo de la definición del nombre de la clase.

Importante: cualquier modificación al valor de una variable de clase es **arrastrada** hacia las instancias.

Ejemplo: en la misma clase Perro podemos asignar un atributo “Género” que sea siempre el mismo para todos.

```
class Perro:
    # Atributo de Clase
    genero= "Canis"
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad
```

Objetos y clases

Instancia

Para instanciar una clase en el interprete de Python escribimos el nombre de la clase con paréntesis. Esto hace que la instancia de Perro se coloque en una posición de memoria. Para poder almacenarla en una variable hacemos lo siguiente:

```
miMascota = Perro()  
otroPerro = Perro()
```

Para instanciar el perro con atributos de nombre y edad hacemos lo siguiente:

```
miMascota = Perro("Popey", 8)  
otroPerro = Perro("Bart", 5)
```

*Obviamos el parámetro **self** ya que es transparente al usuario en Python.*

Podemos acceder a cada atributo de cada instancia mediante el nombre de la variable y el nombre del atributo asociado:

```
miMascota.edad  
otroPerro.nombre
```

Si queremos cambiar un atributo podríamos hacer lo siguiente:

```
otroPerro.edad= 10  
otroPerro.genero= "Felis"
```

Métodos de instancias

- Los métodos de instancia son “funciones” definidas dentro de una clase que solo pueden ser llamadas desde la instancia de la clase.
- Como el método `__init__()`, en un método de instancia siempre el primer parámetro será ***self***.

```
class Perro:
    # Atributo de Clase
    genero= "Canis"
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad
    # Método de instancia
    def imprimir(self):
        return f'{self.nombre} tiene {self.edad} años.'
    # Otro método de instancia
    def ladrar(self, sonido):
        return f'{self.nombre} dice {sonido}.'
```

Llamada a un método

Para llamar a cada método simplemente utilizamos el operador unario (el punto) y entre paréntesis pasamos los parámetros (que puede o no tener):

```
miMascota = Perro("Paka", 11)
miMascota.imprimir()
miMascota.ladRAR("Guau guau!")
```

Utilizando **print** podemos ver las propiedades de cada objeto:

```
print("Género:", miMascota.genero)
print(miMascota.imprimir())
print(miMascota.ladRAR("Guau, guau!"))
```

Género: Canis
Paka tiene 11 años.
Paka dice Guau, guau!.

terminal



Ejemplo_perro.py