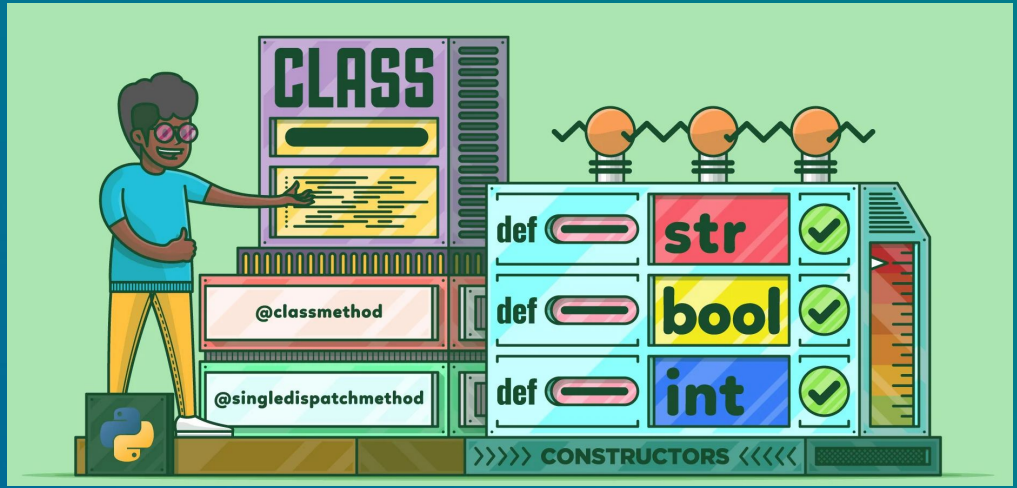
Programación orientada a objetos



Paradigma orientado a objetos

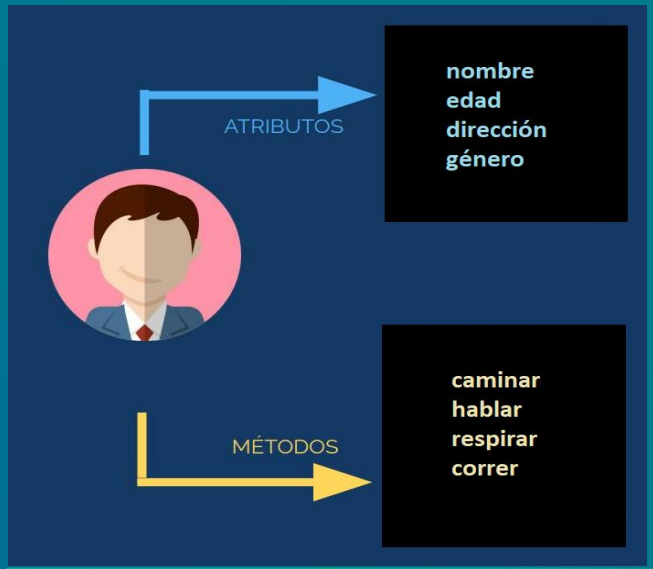
Este paradigma responde a una forma de encarar el desarrollo de software. El mismo se basa en la definición de objetos que forman parte de un dominio en cuestión, interactuando con otros objetos que se comunican por medio de mensajes. Por lo tanto, la programación orientada a objetos define la forma de producir código bajo este paradigma.

Pilares de POO

Abstracción	Proceso de definir los atributos y métodos de una clase.
Encapsulamiento	Protege la información de manipulaciones no autorizadas
Polimorfismo	Da la misma orden a varios objetos para que respondan de diferentes maneras
Herencia	Las clases hijas heredan atributos y métodos de las clases padre.

¿Qué son los objetos?

Un objeto es la representación de una abstracción (una clase) en memoria. En un programa podría ser una persona con características (atributos) como nombre, edad y dirección, y comportamientos (métodos) como caminar, hablar, respirar y correr.

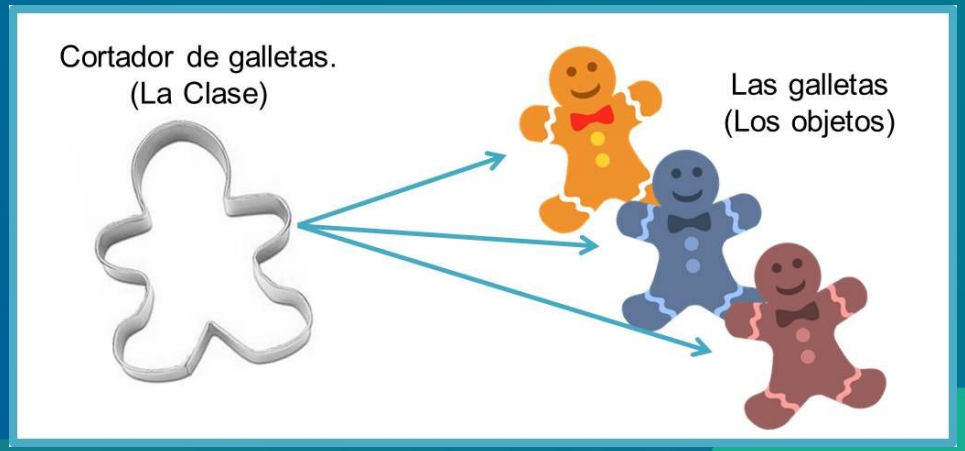


Comunicación entre objetos

Los objetos no trabajan de forma aislada, sino que se comunican entre sí mediante el envío de mensajes. Estos se dan cuando un objeto invoca un método de otro para solicitarle una acción o información. Este intercambio permite que los objetos trabajen juntos para resolver problemas más complejos, manteniendo independencia y encapsulamiento.

¿Qué es una clase?

Podemos definir a una clase como un molde que definirá las características y el comportamiento que tendrá un objeto



Cómo definir una clase

Todas las definiciones de clase comienzan con la palabra clave class, seguida del nombre de la clase y dos puntos.

```
class Personaje:
    pass
```

Los nombres de las clases de python se escriben en CapitalizedWords (UpperCamelCase)

¿Cómo definir atributos?

A las características de un objeto las vamos a llamar atributos. Estos atributos los tendrán todos los objetos de tipo Personaje. Los mismos se definirán en un método llamado __init__ (Constructor de la clase)

```
class Personaje:
    def __init__(self,id,nombre,nano,vuela) -> None:
        self.id = id # crea un atributo id y le asigna el valor id.
        self.nombre = nombre
        self.usa_nanotecnologia = nano
        self.puede_volar = vuela
```

Crear una instancia

La creación de un nuevo objeto a partir de una clase se denomina instanciación de un objeto: se crea el objeto en memoria (por eso se dice que un objeto es una instancia de una clase)
Se puede instanciar un nuevo Personaje escribiendo el nombre de la clase, seguido de paréntesis de apertura y cierre:

```
personaje_A = Personaje(1,'IronMan','No', 'Si')
personaje_B = Personaje(2,'IronSpider','Si', 'No')
```

Acceder a los atributos

Se puede acceder a sus atributos de instancia mediante la notación de puntos:

```
personaje_A = Personaje(1,'IronMan','No', 'Si')
personaje_B = Personaje(2,'IronSpider','Si', 'No')

print(personaje_A.nombre) # IronMan
print(personaje_B.nombre) # IronSpider
```

Cómo definir métodos

Los métodos de instancia representan las acciones que pueden realizar los objetos. Son funciones que se definen dentro de una clase y solo se pueden llamar desde una instancia de esa clase. Al igual que .__init__(), el primer parámetro de un método de instancia siempre es self.

```
class Personaje:
    def __init__(self,id,nombre,nano,vuela) -> None:
        self.id = id # crea un atributo id y le asigna el valor id.
        self.nombre = nombre
        self.usa_nanotecnologia = nano
        self.puede_volar = vuela

    def retornar_descripcion(self) -> str:
        return '{0}-{1}'.format(self.id,self.nombre)
```

Llamar a un método

En la clase Personaje, el método retornar_descripcion devuelve una cadena que contiene el id y el nombre del personaje.

```
personaje_A = Personaje(1,'IronMan','No', 'Si')
personaje_B = Personaje(2,'IronSpider','Si', 'No')

print(personaje_A.retornar_descripcion())# 1-IronMan
print(personaje_B.retornar_descripcion())# 2-IronSpider
```