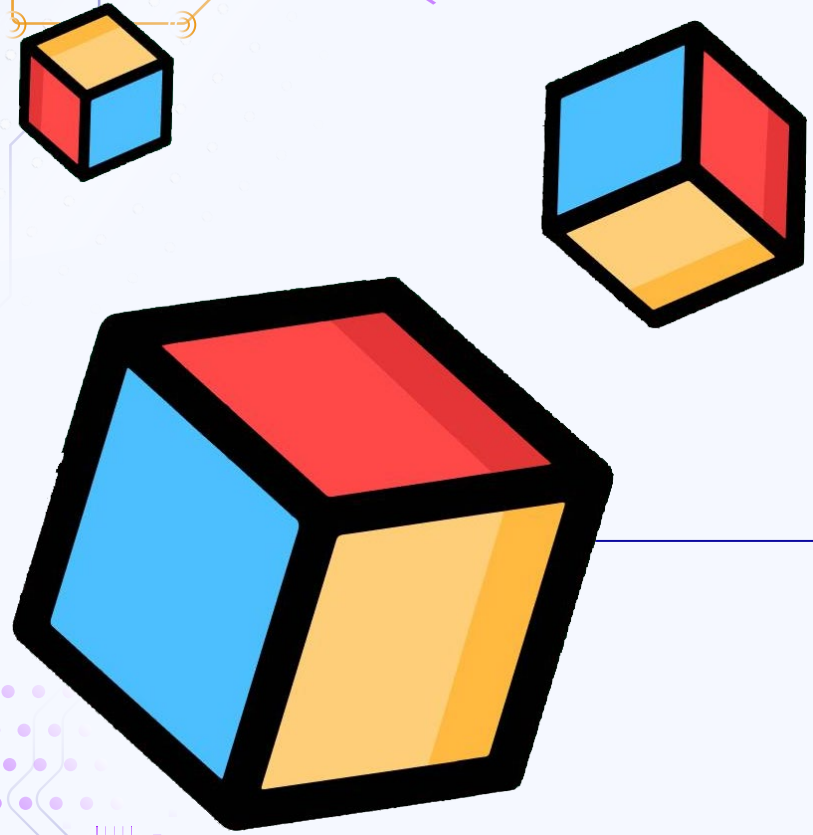


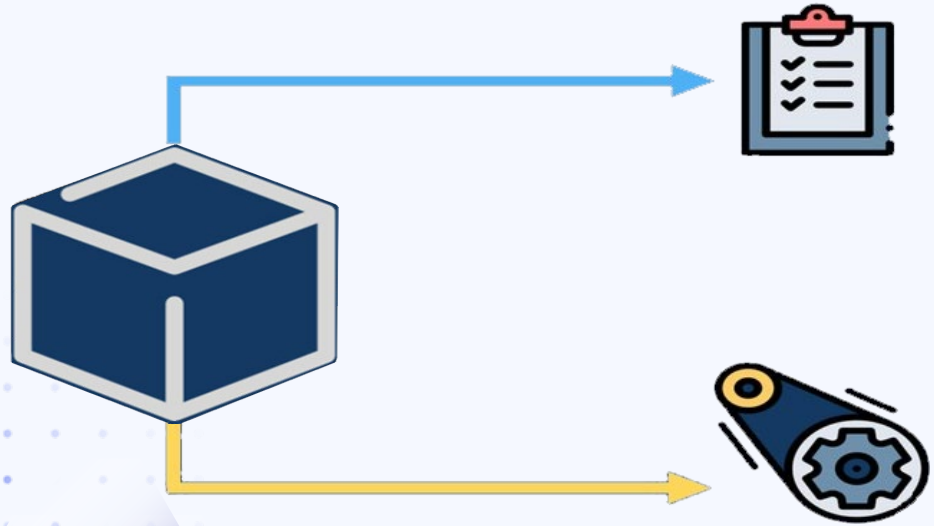
Programación Orientada a Objetos

¿Qué es la Programación Orientada a Objetos (POO)?

Es un paradigma de programación que utiliza "objetos" para representar atributos y métodos. Cada objeto es una instancia de una "clase", que define las propiedades y comportamientos que el objeto puede tener.

Clase:

Es como una plantilla o molde que define las características(atributos) y comportamientos(Métodos) de los objetos.



Atributos:

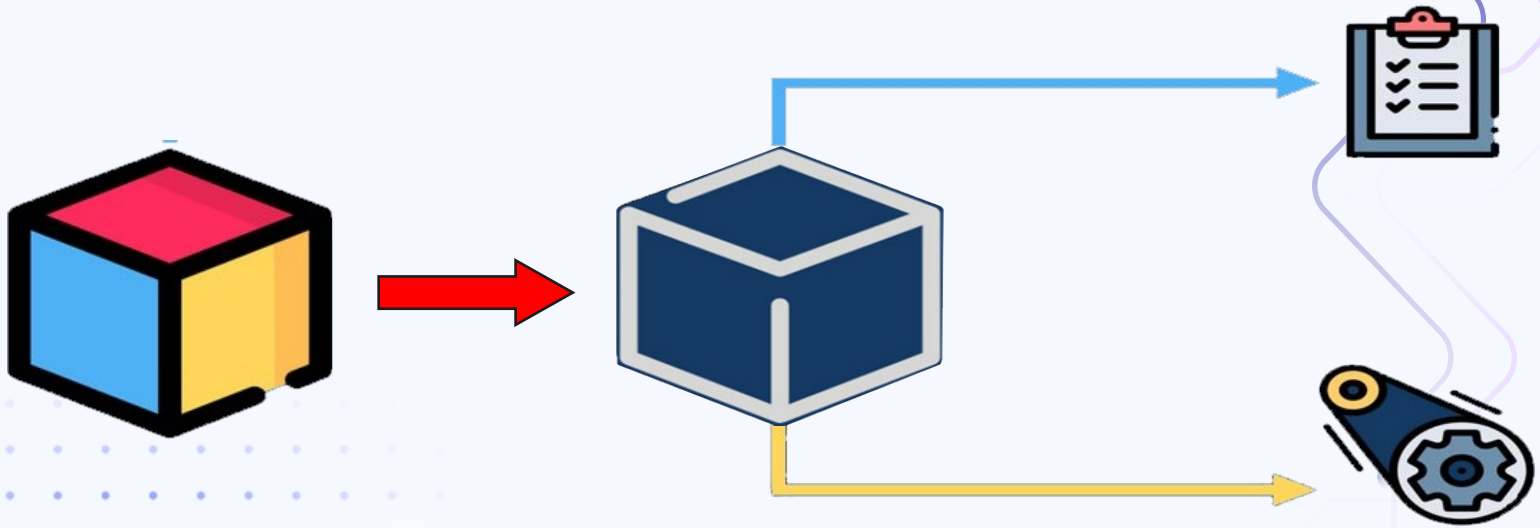
Son las propiedades de una clase. Definen las características de los objetos.

Métodos:

Son funciones definidas dentro de una clase que describen los comportamientos de los objetos.

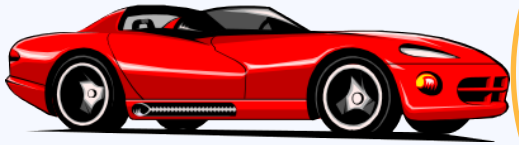
Objeto:

Es una instancia de una clase. Es un "ejemplar" de esa clase con valores específicos.



Ejemplo de Objeto:

Objeto



Todo objeto del mundo real tiene 2 componentes

Características (Atributos)

- Marca
- Modelo
- Color
- Velocidad máxima
- etc.

Comportamientos (Métodos)

- Poner en marcha
- Acelerar
- Frenar
- Retroceder
- etc.

En POO, un auto sería un objeto y la definición general de un auto sería una clase.

¿QUÉ ES LA PROGRAMACIÓN ORIENTADA A OBJETOS?

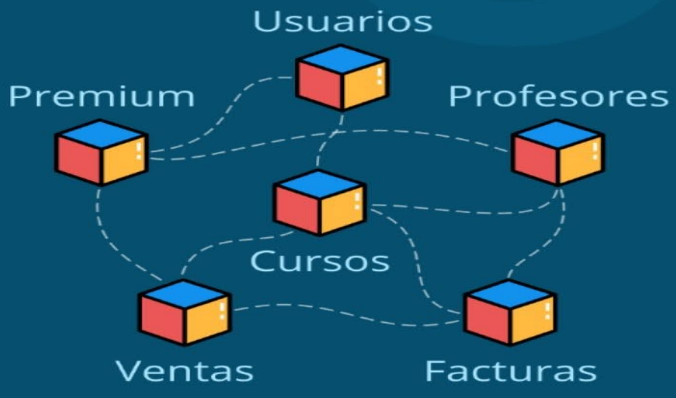
Es un paradigma de programación que organiza las funciones en entidades llamadas objetos.

- **Los objetos se crean a partir de una plantilla llamada clase**. Cada objeto es una instancia de su clase.

INSTANCIACIÓN

- Los objetos tienen **datos (atributos)** y **funcionalidades (métodos).**

- En una aplicación los objetos están separados **pero se comunican entre ellos.**



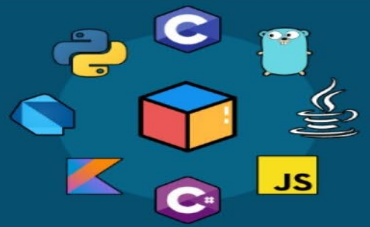
ATRIBUTOS

Nombres
Apellidos
Correo
Contraseña
Premium

MÉTODOS

Editar perfil
Iniciar sesión
Cerrar sesión
Cambiar contraseña
Pasar a premium

Puedes programar con este paradigma **en la mayoría de lenguajes.**



Pilares de la Programación Orientada a Objetos (POO)

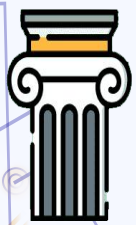
Encapsulamiento

Es el proceso de agrupar datos (atributos) y métodos que operan sobre esos datos en una sola clase. Esto ayuda a ocultar los detalles internos y a proteger los datos.

Herencia

Permite crear una nueva clase a partir de una clase existente. La nueva clase hereda atributos y métodos de la clase base.

Polimorfismo

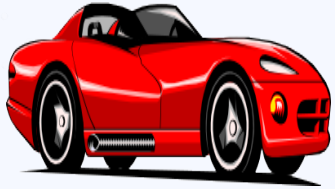


Permite usar una misma interfaz para diferentes tipos de objetos, es decir, diferentes clases pueden tener métodos con el mismo nombre pero comportamientos diferentes.

Abstracción

Es la simplificación de la complejidad al ocultar los detalles innecesarios y mostrar solo la funcionalidad esencial.

Ejemplo 1:



Características (Atributos)

- Marca
- Modelo
- Color
- Velocidad máxima
- etc.

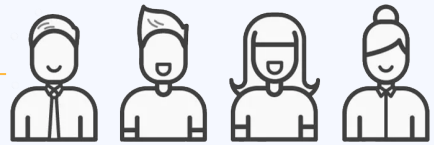
Comportamientos (Métodos)

- Poner en marcha
- Acelerar
- Frenar
- Retroceder
- etc.

Clase Autos

```
método constructor(marca, modelo, color, velocidad)
    marca = marca
    modelo = modelo
    color = color
    velocidad = velocidad

método poner marcha()
    imprimir("El auto está encendido")

método acelerar()
    imprimir("El auto está acelerando")

método frenar()
    imprimir("El auto está frenando")

método retroceder()
    imprimir("El auto está retrocediendo")
```

Ejemplo 2:



Características (Atributos)

- DNI
- Apellidos
- Nombres
- Edad

Comportamientos (Métodos)

- Caminar
- Correr
- Detener
- Saludar
- Cumplir Años

Clase Personas

```
método constructor(dni, apellidos, nombres, edad)
    dni = dni
    apellidos = apellidos
    nombres = nombres
    edad = edad

método caminar()
    imprimir(“La persona está caminando")

método correr()
    imprimir(“La persona está corriendo")

método detener()
    imprimir(“La persona se detuvo")

método saludar()
    imprimir(“La persona esta saludando")

método cumplir años()
    imprimir(“La persona esta de cumpleaños")
```

Ejemplo 3:

Características (Atributos)

- Titulo
- Autor
- Año
- Edición
- Paginas

Comportamientos (Métodos)

- Mostrar información
- Cambiar titulo

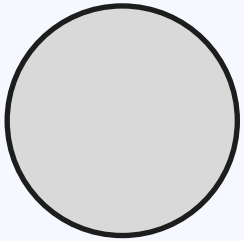
Clase Libros

método constructor(titulo, autor, annio, edición, paginas)
    titulo = titulo
    autor = autor
    annio = annio
    edición = edición
    paginas = paginas

método mostrar información()
    imprimir(“El titulo del libro es “, titulo)
    imprimir(“El autor es “, autor)
    imprimir(“El año de publicación es “, annio)
    imprimir(“Edición “, edición)
    Imprimir(“Tiene “, paginas, “paginas”)

método cambiar titulo(nuevo titulo)
    titulo = nuevo titulo
    imprimir(“El titulo se ha cambiado")

Ejemplo 4:



Características (Atributos)
- radio

Comportamientos (Métodos)
- Calcular área
- Calcular circunferencia
- Cambiar radio

Clase Circunferencias

método constructor(radio)
radio = radio

método calcular area()
retornar 3.141592 * radio^2

método calcular circunferencia()
retornar 2 * 3.141592 * radio

método cambiar radio(nuevo_radio)
radio = nuevo_radio
Imprimir (“El radio se ha cambiado a ”, radio)

Ejemplo 5:

Características (Atributos)
- Base
- altura

Comportamientos (Métodos)
- Calcular área
- Calcular perímetro
- Cambiar dimensiones

```
Clase Rectangulos

    método constructor(base, altura)
        base = base
        altura = altura

    método calcular area()
        retornar base * altura

    método calcular perimetro()
        retornar 2 * (base + altura)

    método cambiar dimensiones(nueva_base, nueva_altura )
        base = nueva_base
        altura = nueva_altura
        Imprimir (“Las dimensiones han cambiado”)
```

Como se hace en Python:

Clase Autos

método constructor(marca, modelo, color, velocidad)
marca = marca
modelo = modelo
color = color
velocidad = velocidad

método poner marcha()
imprimir("El auto está encendido")

método acelerar()
imprimir("El auto está acelerando")

método frenar()
imprimir("El auto está frenando")

método retroceder()
imprimir("El auto está retrocediendo")

```
class Autos:

    def __init__(self, marca, modelo, color, velocidad):
        self.marca = marca
        self.modelo = modelo
        self.color = color
        self.velocidad = velocidad

    def poner_marcha(self):
        print("El auto está encendido")

    def acelerar(self):
        print("El auto está acelerando")

    def frenar(self):
        print("El auto está frenando")

    def retroceder(self):
        print("El auto está retrocediendo")
```

Para instanciar una clase o crear un objeto ➔ auto1 = Autos("Chevrolet", "Onix", "Negro",260)