Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Carlos Gallegos

Israel Ledesma Alcala

Grupo: 941

Programación Orientada a Objetos

Paradigmas de la programación

Viernes 17 de Septiembre del 2025

Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Conceptos POO

Clase:

class Book(Item):

    def __init__(self, item_id, title, author, year, genre, quantity)

Aqui Book es una clase que representa un libro y sus atributos son autor año,genero título etc.

Objeto:

book = Book(i, t, a, y, g, q)

lib.add_item(book)

Aquí se crea un objeto book con los datos que ingresa el usuario y se agrega a la biblioteca.

Herencia:

class Item(ABC):

class Book(Item):

class Magazine(Item):

La clase book y magazine heredan de ítem lo que significa que comparten los mismos atributos.

Encapsulamiento:

self._id = item_id

self._title = title

El guión bajo indica que son atributos protegidos.

Abstracción:

```
class Item(ABC):
    @abstractmethod
    def display_info(self):
        pass
```

Item es una clase abstracta que obliga a las demás subclases a implementar el método display_info().

Polimorfismo:

```
for item in lib.items:
    item.display_info()
```

se comporta distinto según el tipo de ítem.

Comparación entre la versión en C y la versión en Python.

En C, se definen las estructuras y las funciones por separado mientras que en python todo va dentro de una clase y nomas se usan unos métodos para operar sobre lo datos en C no hay herencia ni abstracción mientras que en python si,y el encapsulamiento en C no protege los atributos y en python si se encarga de protegerlos.

Conclusión

POO te ayuda a que tu código sea mucho más legible y poder tenerlo de una manera mucho más organizada así como lógico lo que ayuda a poder programar mucho mejor y sin tantos errores.