Python Programming

Módulo 3

Herencia

Herencia

Cuando una clase B hereda de una clase A, aquélla conserva todos los métodos y atributos de ésta.

```
class ClaseA:
    def __init__(self):
        self.a = 1

class ClaseB(ClaseA):
    pass

mi_objeto = ClaseB()
print(mi_objeto.a)
```

En este código, `mi_objeto` es una instancia de `ClaseB`, que no ha definido ningún atributo `a`. Sin embargo, lo hereda de la `ClaseA`. La `ClaseB` podría definir nuevos atributos y funciones, e incluso reemplazar a los de su clase padre.

```
class ClaseB(ClaseA):
    def __init__(self):
        self.b = 2
```

En este caso, definimos el método de inicialización `__init__()` y creamos un nuevo atributo `b`. No obstante, esto reemplaza la antigua definición de `__init__()` en `ClaseA` de modo que, por ejemplo, `self.a` no se habrá definido. Siempre que reemplazamos un método, para permitir que las funciones con el mismo nombre en las clases padres se ejecuten debemos emplear la función `super()` del siguiente modo.

```
class ClaseB(ClaseA):
    def __init__(self):
        super().__init__()
        self.b = 2
```

Como alternativa también es posible llamar a la función `__init__()` de la clase padre directamente.

```
class ClaseB(ClaseA):
    def __init__(self):
        ClaseA.__init__(self)
        self.b = 2
```

¡Muchas gracias!

¡Sigamos trabajando!