Programación Orientada a Objetos

Sobrescritura de métodos

Índice

- Sobrescritura de métodos
- Uso de super
- Sobrescritura de métodos especiales
- Sobrecarga de operadores

Sobrescritura de métodos

- Sobrescribir un método es volver a definir en la subclase un método heredado de la superclase.
- Simplemente, se vuelve a crear con los mismos parámetros que el heredado

```
class Padre:
    def metodo(self):
        print("método del padre")
class Hija(Padre):
     def metodo(self):
        print("método del hijo")
h1=Hija()
h1.metodo() # imprime método del hijo
```

Uso de super

- La palabra reservada super puede utilizarse para llamar a un métdo, atributo o constructor de la superclase desde la subclase.
- Formato: super(ClaseHija,self).metodo()

```
class Padre:
    def metodo(self):
        print("método del padre")
class Hija(Padre):
    def metodo(self):
        super(Hija,self).metodo()
        print("método del hijo")
h1=Hija()
h1.metodo() # imprime: método del padre
                       método del hijo
```

Sobrescritura de métodos especiales

- Todas las clases cuentan con unos métodos especiales que se pueden sobrescribir:
  - __repr__(self). Devuelve una representación formal del objeto que pueda ser legible para python y que sea inequívoca, sin ambigüedades.
  - __str__(self). Devuelve una representación imprimible del objeto. A diferencia del anterior, no tiene que ser inequívoca, se trata de una representación libre
- A estos métodos no se les llama directamente, sino a través de las funciones especiales repr() y str():

```
class Prueba:
    def __repr__(self):
        return '{self.__class__.__name__}'.format(self=self)
    def __str__(self):
        return "Objeto prueba"
p=Prueba()
print(repr(p)) # imprime Prueba
print(str(p)) # imprime Objeto prueba
```

Sobrecarga de operadores

- En Python se pueden sobrecargar los operadores, es decir, definir una funcionalidad personalizada para el operador
- Dependiendo del operador, se deberán sobrescribir unos métodos específicos en la clase:
  - operador + : método __add__
  - operador - : método __sub__
  - operador * : método __mul__
  - operador / : método __div__