# Curso: Programación Orientada a Objetos

Tema:

Método especial de Clase

# Sobrecarga de Operadores

```
# operador +
a = 2
b = 3
print(a + b)    # Resultado  5

a = 'Hola '
b = 'Mundo'
print(a + b)   #  Resultado  Hola Mundo

a = [1,2,3]
b = [6,7,8]
print(a + b)  # Resultado  [1, 2, 3, 6, 7, 8]

Objeto1 + objeto2
```

# Sobrecarga de Operadores

**Binary Operators:**

| OPERATOR | MAGIC METHOD |
|---|---|
| + | __add__(self, other) |
| – | __sub__(self, other) |
| * | __mul__(self, other) |
| / | __truediv__(self, other) |
| // | __floordiv__(self, other) |
| % | __mod__(self, other) |
| ** | __pow__(self, other) |

# Sobrecarga de Operadores

**Comparison Operators :**

| OPERATOR | MAGIC METHOD |
|---|---|
| < | __lt__(self, other) |
| > | __gt__(self, other) |
| <= | __le__(self, other) |
| >= | __ge__(self, other) |
| == | __eq__(self, other) |
| != | __ne__(self, other) |

# Sobrecarga de Operadores

**Assignment Operators :**

| OPERATOR | MAGIC METHOD |
|---|---|
| -= | __isub__(self, other) |
| += | __iadd__(self, other) |
| *= | __imul__(self, other) |
| /= | __idiv__(self, other) |
| //= | __ifloordiv__(self, other) |
| %= | __imod__(self, other) |
| **= | __ipow__(self, other) |

# Sobrecarga de Operadores

**Unary Operators :**

| OPERATOR | MAGIC METHOD |
| --- | --- |
| – | __neg__(self, other) |
| + | __pos__(self, other) |
| ~ | __invert__(self, other) |

# Sobrecarga de Operadores

```
class Persona:
    def __init__(self, nombre):
        self.nombre = nombre

def __add__(self, objeto2):
        return f’{self.nombre}   {objeto2.nombre}’

persona1 = Persona('Juan')
persona2 = Persona('Carlos’)

print(persona1 + persona2)
```

```
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def __add__(self, objeto2):
        return f’{self.nombre}   {objeto2.nombre}’

    def __sub__(self,objeto2):
        return self.edad - objeto2.edad

persona1 = Persona('Juan',28)
persona2 = Persona('Carlos',20)

print(persona1 + persona2)
print(persona1 - persona2)
```

# Método __add__()

Permite modificar la adición de dos objetos utilizando el operador “+”

Sintaxis : __add__(self, objeto2)

Ejemplo

```
Class Numero:
        def __init__(self, numero)
                self.numero=numero

        def __add__ (self, objeto2):
                s=str(self.numero + objeto2.numero)
                return s
n1= Numero(6)
n2=Numero(9)
Print (“La suma es:”, n1 + n2)
```

# Método __sub__()

Permite modificar la resta de dos objetos utilizando el operador “-”

Sintaxis : __sub__(self, objeto2)

Ejemplo

```
Class Numero:
        def __init__(self, numero)
                self.numero=numero

        def __sub__ (self, objeto2):
                s=str(self.numero - objeto2.numero)
                return s
n1= Numero(6)
n2=Numero(9)
Print (“La resta es:”, n1 - n2)
```

# Método __mul__()

Permite modificar la multiplicacion de dos objetos utilizando el operador “*”

Sintaxis : __mul__(self, objeto2)

Ejemplo

```
Class Numero:
        def __init__(self, numero)
                self.numero=numero

        def __mul__ (self, objeto2):
                s=str(self.numero * objeto2.numero)
                return s
n1= Numero(6)
n2=Numero(9)
Print (“La multiplicacion es:”, n1 * n2)
```

# Método __floordiv__()

Permite modificar la division de dos objetos utilizando el operador "//", el resultado será un numero entero

Sintaxis : __floordiv__(self, objeto2)

Ejemplo

```
Class Numero:
        def __init__(self, numero)
                self.numero=numero

        def __floordiv__ (self, objeto2):
                s=str(self.numero // objeto2.numero)
                return s
n1= Numero(6)
n2=Numero(9)
Print ("La division es:", n1 // n2)
```

## Problemas

1. Definir una clase llamada Punto con dos atributos x e y.
   Crearle el método especial __str__ para retornar un string con el formato (x,y).

2. Declarar una clase llamada Familia. Definir como atributos el nombre del padre, madre y una lista con los nombres de los hijos.
   Definir el método especial __str__ que retorne un string con el nombre del padre, la madre y de todos sus hijos.

3. Desarrollar un programa que implemente una clase llamada Jugador.
   Definir como atributo su nombre y puntaje.
   Codificar el método especial __str__ que retorne el nombre y si es principiante (menos de 1000 puntos) o experto (1000 o mas puntos).

4. Desarrollar una clase llamada lista, que permita pasar el método __init__ una lista de 3 valores enteros.
   Redefinir los operadores +, -, * y // con respecto a un valor entero.

Ejemplo Salida Suma:
[2,4,5]
Sumado +10 cada valor de la lista
[12,14,15]

5. Crear una clase persona que tenga como atributo el nombre y la edad.
El operador == retornara verdadero si las 2 personas tienen la misma edad, el operador '>' retornara 'True' si la edad del objeto de la izquierda tiene una edad mayor a la edad del objeto de la derecha del operador '>' y así sucesivamente.