# Detalle ejemplo Clases en Python (clases.py)

## 1. Clase Básica: Silla

python

```
class Silla:
    color = "verde"
    material = "madera"
    precio = 225

# Instanciar la clase
silla1 = Silla()
print(silla1.color, silla1.material, silla1.precio)  # Imprime: verde
madera 225
```

**Explicación**:

- La clase `Silla` tiene atributos de clase (`color`, `material`, `precio`). Todos los objetos de la clase compartirán los mismos valores.
- Es un buen ejemplo introductorio para entender cómo definir atributos de clase y cómo acceder a ellos desde una instancia.

---

## 2. Introducción al Constructor: `__init__`

python

```
# Clase con un constructor sin atributos definidos aún
class Mesilla:
    def __init__(self) -> None:
        pass
# Implementación válida para definir la estructura de la clase antes
de haber decidido los atributos o el comportamiento.
```

**Explicación**:

- Esta clase `Mesilla` utiliza `pass` para indicar que aún no se ha definido el comportamiento. Es útil para esbozar una clase durante el desarrollo inicial.

---

## 3. Clase con Atributos de Instancia: Mesa

python

```
class Mesa:
    def __init__(self, color, material, precio):
        self.color = color
        self.material = material
        self.precio = precio

# Crear un objeto de la clase Mesa
```

```
mesa1 = Mesa("gris", "aluminio", 2750)
print(mesa1.material)  # Imprime: aluminio
```

**Explicación**:

- Aquí se introducen **atributos de instancia** (`color`, `material`, `precio`) que son únicos para cada objeto creado.
- Los estudiantes pueden ver cómo inicializar los valores al momento de crear el objeto.

---

## 4. Métodos de Instancia: Cambiar Atributos

```
python

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

    def cambiar_ciudad(self, nueva_ciudad):
        """Método que permite modificar el atributo ciudad del
objeto."""
        self.ciudad = nueva_ciudad

# Crear una instancia de la clase Persona
persona1 = Persona("Juan", 27, "Madrid")
print(persona1.ciudad)  # Imprime: Madrid
persona1.cambiar_ciudad("Barcelona")
print(persona1.ciudad)  # Imprime: Barcelona
```

**Explicación**:

- Se muestra cómo definir y usar un **método** (`cambiar_ciudad`) para modificar un atributo del objeto.

---

## 5. Método `__str__`: Definir Cómo Imprimir Objetos

```
python

class Coche:
    def __init__(self, marca, modelo, precio):
        self.marca = marca
        self.modelo = modelo
        self.precio = precio

    def __str__(self):
        """Personaliza la salida de los objetos de la clase."""
        return f"Este coche, de la marca {self.marca}, es el modelo
{self.modelo} y tiene un precio de {self.precio:,.2f}€"

# Crear un objeto de la clase Coche
```

```
cochazo1 = Coche("BMW", "850i M", 137975.88)
print(cochazo1)  # Imprime: Este coche, de la marca BMW, es el modelo
850i M y tiene un precio de 137,975.88€
```

**Explicación**:

- El método **`__str__`** permite definir cómo se debe imprimir el objeto cuando se usa `print()`.
- Mejora la legibilidad y proporciona información útil del objeto en un formato personalizado.

---

## 6. Sobrecarga del Operador **`__add__`**

### Ejemplo Mejorado: Sumar Cantidades de Cuentas

```
python

class Cuenta:
    def __init__(self, titular, cantidad):
        self.titular = titular
        self.cantidad = cantidad

    def __add__(self, otra_cuenta):
        """Crea una nueva cuenta combinando las cantidades y
titulares."""
        nueva_cantidad = self.cantidad + otra_cuenta.cantidad
        nuevo_titular = f"{self.titular} y {otra_cuenta.titular}"
        return Cuenta(nuevo_titular, nueva_cantidad)

    def __str__(self):
        return f"Titular(es): {self.titular}, Cantidad:
{self.cantidad:.2f}€"

# Crear dos cuentas y sumarlas
cuenta1 = Cuenta('12345678J', 300000.00)
cuenta2 = Cuenta('98765432J', 400000.00)
cuenta_combinada = cuenta1 + cuenta2

print(cuenta_combinada)  # Imprime: Titular(es): 12345678J y
98765432J, Cantidad: 700000.00€
```

**Explicación**:

- En este ejemplo, `__add__` crea un **nuevo objeto `Cuenta`**, combinando los valores de dos cuentas existentes, manteniendo consistencia y lógica.
- También se añadió el método `__str__` para mejorar la presentación del resultado.

---

## 7. Ejercicio Práctico: Sobrecarga del Operador **`__add__`** para una Cartera de Bonos

```
python

class Bono:
    def __init__(self, referencia, valor):
        self.referencia = referencia
        self.valor = valor

    def __add__(self, otro_bono):
        """Crea una nueva instancia de Bono que combina referencias y
valores."""
        nueva_referencia = f"{self.referencia} y
{otro_bono.referencia}"
        suma_valores = self.valor + otro_bono.valor
        return Bono(nueva_referencia, suma_valores)

    def __str__(self):
        return f"Referencias: {self.referencia}, Inversión:
{self.valor:,.2f}€"

# Crear instancias de la clase Bono
bono1 = Bono('1234J', 300000.00)
bono2 = Bono('5678J', 400000.00)
bono3 = Bono('9876W', 600000.00)

# Utilizar el método __add__ encadenado para crear una cartera de
bonos
cartera = bono1 + bono2 + bono3

# Imprimir la cartera de bonos
print(cartera)  # Imprime: Referencias: 1234J y 5678J y 9876W,
Inversión: 1,300,000.00€
```

**Explicación**:

- Este ejercicio permite a los estudiantes comprender cómo se pueden **encadenar operaciones** y cómo se puede crear un nuevo objeto a partir de la combinación de varios.
- El formato de impresión está personalizado para proporcionar información detallada de la inversión.

---

## 8. Ejercicio de Clase: Crear y Sumar Bonos

**Enunciado:**

Crea tres instancias de la clase `Bono` con diferentes referencias y valores. Usa el operador `+` sobrecargado para combinar los bonos y crear una cartera. Imprime la referencia y el valor total de la cartera.

**Solución**:

```
python

# Crear las instancias de la clase Bono
bono1 = Bono('1111A', 100000.00)
```

```
bono2 = Bono('2222B', 150000.00)
bono3 = Bono('3333C', 200000.00)

# Crear la cartera usando la sobrecarga del operador __add__
cartera = bono1 + bono2 + bono3

# Imprimir la cartera
print(cartera)  # Salida: Referencias: 1111A y 2222B y 3333C,
Inversión: 450,000.00€
```