

Curso: Programación Orientada a Objetos

Tema:

Método `__init`

Llamada de métodos desde otro método de la misma clase



Llamada de métodos desde otro método de la misma clase

Hasta ahora todos los problemas planteados hemos llamado a los métodos desde donde definimos un objeto de dicha clase, por ejemplo:

```
empleado1=Empleado ("Diego",2000)  
Empleado1.paga_impuesto()
```

Utilizamos la sintaxis :

```
[nombre de objeto].[nombre del método]
```

Llamada de métodos desde otro método de la misma clase

Ahora que pasa si queremos llamar dentro de la clase a otro método que pertenece a la misma clase, la sintaxis es la siguiente:

`self.[nombre del método]`

Importante solo se puede hacer cuando estemos dentro de la misma clase

Problema:

Plantear una clase Operaciones que solicite el método `__init__` la carga de dos enteros e inmediatamente muestre su suma, resta, multiplicación y división. Hacer cada operación en otro método de la clase operación y llamarlos desde el mismo método `__init__`

class Operaciones:

```
def __init__(self):  
    self.valor1=int (input("Ingrese primer valor:"))  
    self.valor2=int (input("Ingrese segundo valor:"))  
    self.sumar()  
    self.restar()  
    self.multiplicar()  
    self.dividir()
```

```
def sumar (self):  
    suma=self.valor1 + self.valor2  
    print("La suma es:", suma)
```

```
def restar (self):  
    resta=self.valor1 - self.valor2  
    print("La resta es:", resta)
```

```
def multiplicar (self):  
    multi=self.valor1 * self.valor2  
    print("El producto es:", multi)
```

```
def dividir (self):  
    divi=self.valor1 / self.valor2  
    print("La division es:", divi)
```

#bloque principal

Operacion1 = Operaciones()

Ejercicios

1. Plantear una clase que administre dos listas de 5 nombres de alumnos y sus notas. Mostrar un menú de opciones que permita:
 1. Cargar alumnos.
 2. Listar alumnos
 3. Mostrar alumnos con notas mayores o iguales a 11
 4. Finalizar Programa

2. Confeccionar una clase que administre una agenda personal. Se debe almacenar el nombre de la persona, teléfono y mail.

Debe mostrar un menú con las siguientes opciones:

 1. Carga un contacto en la agenda
 2. Listado completo de la agenda
 3. Consulta ingresando el nombre de la persona
 4. Modificación de su teléfono y mail
 5. Finalizar programa

Colaboración de Clases

Normalmente un problema resuelto con la metodología de programación orientada a objetos no interviene una sola clase, sino que hay muchas clases que interactúan y se comunican. Plantearemos un problema:

Problema :

Un banco tiene 3 clientes que pueden hacer depósitos y extracciones. También el banco requiere que al final del día calcule la cantidad de dinero que hay depositado.

Solución:

Lo primero que hacemos es identificar las clases: **clase Cliente y clase Banco**
Luego definimos los atributos y los métodos de cada clase

Colaboración de Clases

Cliente

Atributos

nombre

monto

Métodos

__inti__

depositar

extraer

retornar monto

imprimir

Banco

Atributos

3 clientes (objeto cliente)

Métodos

__init__

operar

depósitos_totales

```
class Cliente:
    def __init__(self, nombre):
        self.nombre=nombre
        self.monto=0

    def depositar (self, monto):
        self.monto=self.monto+monto

    def extraer (self, monto):
        self.monto=self.monto-monto

    def retornar_monto(self):
        return self.monto

    def imprimir (self):
        print(f' {self.nombre} :, Tiene depositado la suma de:, {self.monto}')
```

```
class Banco:
    def __init__(self):
        self.cliente1=Cliente("Juan")
        self.cliente2=Cliente("Ana")
        self.cliente3=Cliente("Luis")

    def operar(self):
        self.cliente1.depositar(100)
        self.cliente2.depositar(150)
        self.cliente3.depositar(200)
        self.cliente3.extraer(150)

    def depositos_totales(self):
        total=self.cliente1.retornar_monto()+self.cliente2.retornar_monto()+self.cliente3.retornar_monto()
        print("Total del dinero del Banco:", total)
        self.cliente1.imprimir()
        self.cliente2.imprimir()
        self.cliente3.imprimir()

#Bloque Principal

banco1=Banco()
banco1.operar()
banco1.depositos_totales()
```

Problemas:

1. Plantear un programa que permita jugar a los dados. Las reglas de juego son: se tiran tres dados si los tres salen con el mismo valor mostrar un mensaje que “gano”, sino “perdió”
2. Plantear una clase Club y otra clase Socio.
La clase socio debe tener los siguientes atributos: nombre y la antigüedad en el club (años).
En el método `__init__` de la clase Socio pedir la carga por teclado del nombre y su antigüedad.
La clase Club debe tener como atributos 3 objetos de la clase Socio.
Definir una responsabilidad para imprimir el nombre del socio con mayor antigüedad del club.