# **Conceptos de programación orientada a objetos: Clase y Objeto**

**Concepto de Poo:**  
Es un paradigma de programación que basa su estructura en los objetos, que son entidades o elementos que poseen datos y comportamientos, y que pertenecen a la situación que se esta analizando, teniendo una participación fundamental.

La manera en que los objetos se relacionan e interactúan entre sí, permite identificar la dinámica del contexto.

Los objetos se agrupan en clases, que son plantillas que definen los atributos y métodos que comparten los objetos de una misma clase, además de un mismo objetivo.

**Explicación2:** Es un estilo que trata los datos como objetos con atributos y métodos que pueden aplicarse a estos objetos y también ser heredados por otros objetos.

**Concepto de objeto:**Un objeto es una entidad independiente con sus propios datos y programación.

El concepto renovador de la tecnología de Orientación a Objetos es la suma de funciones a elementos de datos, a esta unión se le llama **encapsulamiento.**

**Ejemplo:**

Un objeto Auto contiene ruedas, motor, velocidad, color, etc. Llamados atributos.  
Encapsulados con estos datos se encuentran los métodos para arrancar, detenerse, doblar, frenar, etc.

La responsabilidad de un objeto consiste en realizar las acciones apropiadas y mantener actualizados sus datos internos.

Cuando otra parte del programa (otros objetos) necesitan que el auto realice alguna de estas tareas (por ejemplo, arrancar) le envía un mensaje. A estos objetos que envían mensajes no les interesa la manera en que el objeto auto lleva a cabo sus tareas ni las estructuras de datos que maneja, por ello, están ocultos.

Entonces, un objeto contiene información pública, lo que necesitan los otros objetos para interactuar con el e información privada, interna, lo que necesita el objeto para operar y que es irrelevante para los otros objetos de la aplicación.

# **Declaración de una clase y creación de objetos**

Poo se basa en la definición de clases; a diferencia de la programación estructurada, que esta centrada en las funciones.

Una clase es un molde del que luego se pueden crear múltiples objetos, con similares características.

Un poco mas abajo se define una clase Persona y luego se crean dos objetos de dicha clase.

Una clase es una plantilla (molde), que define atributos (lo que conocemos como variables) y métodos (lo que conocemos como funciones).

Debemos declarar una clase antes de poder crear objetos (instancias) de esa clase. Al crear un objeto de una clase, se dice que se crea una instancia de la clase o un objeto apropiadamente dicho.

**Self:**

Referencia del objeto que llama al método. Se utiliza para hacer referencia a la instancia actual de una clase. En este contexto, se refiere al objeto en si mismo que esta siendo manipulado o utilizado en un programa.

**Explicación:**Se agrega self para que pueda acceder a las variables de instancia “Nombre” y “edad”

# **Ejercicios**

**Problema1:**Implementaremos una clase Persona que tendrá como atributo (variable) su nombre y dos métodos (funciones), uno de dichos métodos inicializará el atributo nombre y el siguiente método mostrará en la pantalla el contenido de este  
Definir dos objetos de la clase Persona.

Def clase\_1():

class Persona:

def inicializar(self, nom):

self.nombre = nom

def imprimir(self):

print(“Nombre”,self.nombre)

persona1 = Persona()

persona1.inicializar(“Pedro)  
 persona1.imprimir()

**Explicación:** Una clase es un molde, porque después podemos definir muchos objetos de dicha clase, guardando los nuevos métodos y/o funciones de cada objeto que se añade. Esto ahorra la necesidad de (como en la programación estructurada) crear variable por variable y evitar confusiones.

**Problema2:**Imprimir una clase llamada Alumno que tenga como atributos su nombre y su nota. Definir los métodos para inicializar sus atributos, imprimirlos y mostrar un mensaje si esta regular (nota mayor o igual a 4)

Definir dos objetos de la clase Alumno.

class Alumno:

def inicializar(self,nombre,nota):

self.nombre=nombre

self.nota=nota

def imprimir(self):

print("Nombre:",self.nombre)

print("Edad:",self.nota)

def mostrar\_estado(self):

if self.nota>=4:

print(self.nombre, "Esta regular")

else:

print(self.nombre, "Esta libre")

alumno1 = Alumno()

alumno1.inicializar("Juan",2)

alumno1.imprimir()

alumno1.mostrar\_estado()

alumno2 = Alumno()

alumno2.inicializar("Felipe",10)

alumno2.imprimir()

alumno2.mostrar\_estado()

**Explicación:**

Self.nombre = nombre

“Self.nombre” hace referencia al atributo “nombre” de la clase Alumno  
“Nombre” sin antecederle nada, estamos accediendo al parámetro “nombre”  
**Pueden llamarse igual un atributo que el parámetro “nombre”**

Def imprimir(self):

El método imprimir puede acceder al método self.nombre, self.nota

**Problema3:**

Confeccionar una clase que permita cargar el nombre y la edad de una persona. Mostrar los datos cargados. Imprimir un mensaje si es mayor de edad (edad >=18).

def validar():

while True:

try:

nombre = input("Ingrese nombre: ")

if nombre.isnumeric() or any(i.isnumeric() for i in nombre) or not nombre.isalpha() or nombre.count(" ")>=1:

continue

return nombre.capitalize()

except ValueError:

continue

def validar1():

while True:

try:

edad = int(input("Ingrese la edad: "))

if edad <= 0 or not 1<=edad<=100:

continue

return edad

except ValueError:

continue

class Persona:

def datos(self,nombre,edad):

self.nombre=nombre

self.edad=edad

def imprimir(self):

print("Nombre:",self.nombre)

print("Edad:",self.edad)

def mayor(self):

if self.edad >=18:

print(self.nombre, "Es mayor de edad")

else:

print(self.nombre, "es menor de edad")

persona1 = Persona()

persona1.datos(validar(),validar1())

persona1.imprimir()

persona1.mayor()

**Persona 4:**

Desarrollar un programa que carge los lados de un triangulo por teclado e implemente los siguientes métodos: inicializar los atributos, imprimir el valor del lado mayor y otro método que muestre si es equilátero o no.  
El nombre de la clase llamarla Triangulo.

class Triangulo:

def datos(self):

self.lado1=int(input("Ingrese lado 1: "))

self.lado2=int(input("Ingrese lado 2: "))

self.lado3=int(input("Ingrese lado 3: "))

def imprimir(self):

print("Lado 1:",self.lado1)

print("Lado 2:",self.lado2)

print("Lado 3:",self.lado3)

def lado\_mayor(self):

print("Lado mayor")

if self.lado1>self.lado2 and self.lado1>self.lado3:

print(self.lado1)

elif self.lado2>self.lado1 and self.lado2>self.lado3:

print(self.lado2)

else:

print(self.lado3)

def equilatero(self):

if self.lado1==self.lado2 and self.lado1==self.lado3:

print("El triangulo es equilatero")

else:

print("El triangulo no es equilatero")

triangulo1=Triangulo()

triangulo1.datos()

triangulo1.imprimir()

triangulo1.lado\_mayor()

triangulo1.equilatero()

# **Método \_\_init\_\_ de la clase**

El método \_\_init\_\_ es un método especial de una clase en Python. El objetivo fundamental del método \_\_init\_\_ es inicializar los atributos del objeto que creamos.

Básicamente el método \_\_init\_\_ reemplaza al método inicializar que habíamos hecho en el concepto anterior.

La ventaja de implementar el método \_\_init\_\_ en lugar del método inicializar son:

1. El método \_\_init\_\_ es el primer método que se ejecuta cuando se crea un método.
2. El método \_\_init\_\_se llama automáticamente. Es decir, es imposible de olvidarse de llamarlo ya que se llamara automáticamente.
3. Quien utiliza POO en Python conoce el objetivo de este método.

Otras características del método \_\_init\_\_son:

* Se ejecuta inmediatamente luego de crear un objeto
* El método \_\_init\_\_no puede retornar dato
* El método \_\_init\_\_ puede recibir parámetros que se utilizan normalmente para inicializar atributos.
* El método \_\_init\_\_ es un método opcional, de todos modos es muy común declararlo.

Veamos la sintaxis del constructor:

Def \_\_init\_\_(parámetros):

[algoritmo]

**Ejercicio1:**

Confeccionar una clase que represente un empleado. Definir como atributos su nombre y su sueldo.  
En el método \_\_init\_\_ cargar los atributos por teclado y luego en otro método imprimir los datos y por último uno que imprima un mensaje si debe pagar impuestos (Si el sueldo supera a 3000)

def nombredef():

while True:

nombre = input("Ingrese nombre: ")

if nombre.isnumeric() or any(i.isnumeric() for i in nombre) or not nombre.isalpha() or nombre.count(" ")>=1 or len(nombre)<=4:

continue

return nombre.capitalize()

def edaddef():

while True:

try:

edad = int(input("Ingrese la edad: "))

while edad<=0 or not 18<=edad<=100:

edad=int(input("Ingrese la edad: "))

sueldo=float(input("Ingrese su sueldo: "))

while sueldo<=0:

sueldo=int(input("Ingrese su sueldo: "))

return edad,sueldo

except ValueError:

continue

class Empleado:

def \_\_init\_\_(self):

self.nombre = nombredef()

self.edad,self.sueldo = edaddef()

def imprimir(self):

print(f"Nombre :{self.nombre}")

print(f"Edad: {self.edad}")

print(f"Sueldo: {self.sueldo}")

def impuestos(self):

if self.sueldo>=3000:

print("Debe pagar impuestos")

else:

print("No debe pagar impuestos")

empleado1 = Empleado()

empleado1.imprimir()

empleado1.impuestos()

**Ejercicio2:**

Desarrollar una clase que represente un punto en el plano y tenga los siguientes métodos.  
Inicializar los valores de x e y que llegan como parámetros, imprimir en que cuadrante se encuentra dicho punto (concepto matemático, primer cuadrante si x e y son positivos, si x<0 e y>0 segundo cuadrante, etc)

class Punto:

def \_\_init\_\_(self,x,y):

self.x=x

self.y=y

def imprimir(self):

print("Coordenada del punto")

print(f"{self.x},{self.y}")

def imprimir\_cuadrante(self):

if self.x>0 and self.y>0:

print("Primer cuadrante")

elif self.x <0 and self.y>0:

print("Segundo cuadrante")

elif self.x<0 and self.y<0:

print("Tercer cuadrante")

elif self.x<0 and self.y<0:

print("Cuarto cuadrante")

else:

print("Esta en un eje")

punto1=Punto(10,3)

punto1.imprimir()

punto1.imprimir\_cuadrante()

print("-"\*20)

punto2=Punto(0,0)

punto2.imprimir()

punto2.imprimir\_cuadrante()

print("-"\*20)

punto3=Punto(-1,-2)

punto3.imprimir()

punto3.imprimir\_cuadrante()

**Ejercicio3:**

Desarrollar una clase que represente un Cuadrado y tenga los siguientes métodos:  
Inicializar el valor del lado llegando como parámetro al método \_\_init\_\_(definir un atributo llamado lado), imprimir su parámetro y su superficie.

def forma1():

class Cuadrado:

def \_\_init\_\_(self, lado):

self.lado=lado

def imprimir(self):

print(self.lado)

def calculo(self):

self.perimetro=self.lado \*4

self.area=self.lado \* self.lado

print("Perimetro del cuadrado:",self.perimetro)

print("Area del cuadrado:",self.area)

cuadrado1 = Cuadrado(2)

cuadrado1.imprimir()

cuadrado1.calculo()

def forma2():

def enterodef():

while True:

try:

numero = int(input("Ingrese un lado: "))

if numero==0:

continue

return numero

except ValueError:

continue

class Cuadrado:

def \_\_init\_\_(self):

self.lado=enterodef()

def perimetrodef(self):

return self.lado\*4

def areadef(self):

return self.lado\*self.lado

def calculo(self):

self.perimetro=self.perimetrodef()

self.area=self.areadef()

print("El perimetro del cuadrado es: ",self.perimetro)

print("El area del cuadrado es: ",self.area)

cuadrado1 = Cuadrado()

cuadrado1.calculo()

def forma3():

class Cuadrado:

def \_\_init\_\_(self,valor):

self.valor=valor

def calculo(self):

perimetro=self.valor\*4

area=self.valor\*self.valor

print(f"El perimetro es: {perimetro}"),print(f"El area es: {area}")

cuadrado1 = Cuadrado(2)

cuadrado1.calculo()

forma3()

**Ejercicio4-5:**

Implementar la clase Operaciones. Se deben cargar dos valores enteros por teclado en el método \_\_init\_\_. Calcular su suma, resta, multiplicación y división, cada una en un método, imprimir los resultados.

class Operaciones:

def \_\_init\_\_(self):

self.valor1=int(input("Ingresa un valor: "))

self.valor2=int(input("Ingresa otro valor: "))

def calculo(self):

print(f"La suma es: {self.valor1+self.valor2}")

print(f"La resta es: {self.valor1-self.valor2}")

print(f"La multiplicacion es: {self.valor1\*self.valor2}")

print(f"La division es: {self.valor1/self.valor2}")

operacion1 = Operaciones()

operacion1.calculo()

# **Llamada de métodos desde otro método de la misma clase**

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

Empleado = Empleado(“Diego”,2000)  
Empleado.paga\_impuestos()

Utilizamos la sintaxis:  
[nombre del objeto].[nombre del método]

Es decir, antecedemos el nombre del método el nombre del objeto y el operador punto

**Ahora bien, 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]**

**Es importante tener en cuenta que esto solo se puede hacer cuando estamos dentro de la misma clase.**

**Problema1:**

Plantear una clase Operaciones que solicite en 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.division()  
 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):  
 multiplicar = self.valor1 \* self.valor2  
 print("La multiplicacion es: ",multiplicar)  
 def division(self):  
 try:  
 print("La division es: ", round(self.valor1 / self.valor2,2))  
 except ZeroDivisionError:  
 print("Error")  
  
operacion1 = Operaciones()

**Problema2:**

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 7.
4. Finalizar Programa
5. class Alumnos:  
    def \_\_init\_\_(self):  
    self.nombre = []  
    self.notas = []  
    def menu(self):  
    opcion = 0  
    while opcion != 4:  
    print("1.Cargar Alumnos")  
    print("2.Listado de Alumnos")  
    print("3.Listado de Alumnos con notas mayores o iguales a 7")  
    print("4.Finalizar el programa")  
    opcion = int(input("Ingrese su opcion: "))  
    if opcion ==1:  
    self.cargar()  
    elif opcion ==2:  
    self.listar()  
    elif opcion ==3:  
    self.NotasAltas  
    def cargar(self):  
    for i in range(5):  
    nom=input(f"Ingrese el nombre del alumno {i+1}: ")  
    self.nombre.append(nom)  
    no=int(input(f"Ingrese la nota de {nom}: "))  
    self.notas.append(no)  
    def listar(self):  
    print("Listado Completo de Alumnos")  
    for i in range(5):  
    print(self.nombre[i],self.notas[i])  
    def NotasAltas(self):  
    print("Listado de Notas altas")  
    for i in range(5):  
    if self.notas[i]>=7:  
    print(self.nombre[i], self.notas[i])  
     
   alumno = Alumnos()  
   alumno.menu()

**Problema3:**

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 de 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 el programa

class Agenda:  
 def \_\_init\_\_(self):  
 self.contactos={}  
 def menu(self):  
 opcion = 0  
 while opcion!=5:  
 print("1 - Carga de un contacto en la agenda")  
 print("2 - Listado completo de la agenda")  
 print("3 - Consulta ingresando el nombre de una persona")  
 print("4 - Modificacion del telefono y mail")  
 print("5 - Finalizar el programa")  
 opcion = int(input("Ingrese su opcion"))  
 if opcion ==1:  
 self.cargar()  
 elif opcion ==2:  
 self.listado()  
 elif opcion ==3:  
 self.consulta()  
 elif opcion ==4:  
 self.modificacion()  
 def cargar(self):  
 nombre = input("Ingrese el nombre de la persona: ")  
 telefono = input("Ingrese el telefono: ")  
 mail = input("Ingrese el mail: ")  
 self.contactos[nombre]= (telefono,mail)  
 def listado(self):  
 print("Listado de la agenda")  
 for i in self.contactos:  
 print(i, self.contactos[i][0],self.contactos[i][1])  
 def consulta(self):  
 nombre = input("Ingrese el nombre de la persona a consultar: ")  
 if nombre in self.contactos:  
 print(nombre,"su telefono es: ",self.contactos[nombre][0], "y su mail es: ",self.contactos[nombre][1])  
 else:  
 print("No exite un contacto con dicho nombre")  
 def modificacion(self):  
 nombre = input("Ingrese el nombre de la persona a modificar su telefono y mail")  
 if nombre in self.contactos:  
 telefono=input("Ingrese nuevo telefono")  
 mail = input("Ingrese el nuevo mail")  
 self.contactos[nombre]=(telefono,mail)  
 else:  
 print("No existe un contacto con dicho nombre")  
agenda1 = Agenda()  
agenda1.menu()

# **Colaboración de clases**

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

Plantearemos un problema separando las actividades en dos clases.

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

Cliente:  
 atributos:  
 -nombre  
 -monto  
 métodos:  
 \_\_init\_\_  
 depositar  
 extraer  
 retornar\_monto  
 imprimir

Banco:  
 atributos:  
 3 cliente (3 objetos de la clase 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 retornar(self):  
 return self.monto  
 def imprimir(self):  
 print(self.nombre,"tiene depositado la suma de",self.monto)  
  
class Banco:  
 def \_\_init\_\_(self):  
 self.cliente1 = Cliente("Richard")  
 self.cliente2 = Cliente("Skarlett")  
 self.cliente3 = Cliente("Felipe")  
  
 def operar(self):  
 total = self.cliente1.retornar() + self.cliente2.retornar() + self.cliente3.retornar()  
 print("Total de dinero del banco: ",total)  
 self.cliente1.imprimir()  
 self.cliente2.imprimir()  
 self.cliente3.imprimir()