CLASES

Beneficios OPP:

Permite reutilizar código en diferentes partes del programa.

Las clases proporcionan una estructura clara y jerárquica para el código

Los objetos ocultan detalles internos reduciendo la dependencia entre diferentes partes del código.

Permite que objetos de diferentes clases respondan de manera diferente a los mismos mensajes

Clases: (class) plantilla que define las propiedades y comportamientos de un objeto.

Atributos: (__init__) características que le damos a un objeto.

Métodos: funciones que definen cada objeto.

Sintaxis de las clases:

```
class Mascota:
    def  __init__ (self, animal, edad, raza):
        self.animal = animal
        self. edad = edad
        self.raza = raza
        self.hogar = 'domicilio'       **parámetro por defecto
```

felix = Mascota('gato', 1, 'gato común')

Para **acceder a los atributos**: nombreclase.nombreatributo:

felix.animal → output: 'gato'

felix.hogar → output: 'domicilio'

felix.raza → output: 'gato común'

Para **cambiar el valor** definido: nombreclase.nombreatributo = nuevovalor

felix.edad = 2

Sintaxis de los métodos: (cumple, rango edad, vacunas)

```
class Mascota:
    def  __init__ (self, animal, edad, raza):
        self.animal = animal
        self. edad = edad
        self.raza = raza
        self.hogar = ‘domicilio’       **parámetro por defecto

    def  cumple(self):
        self. edad += 1
        return f’la edad de nuestra mascota es de {self.edad}'

    def  rango_edad(self):
        if self.edad < 2:
          return 'junior'
        elif self.edad > 10:
          return 'senior'
        else:
          return 'adulto'

    def vacunas(self, numero_vacunas):
        self.numero_vacunas = numero_vacunas

        if self.animal == "gato":
            if numero_vacunas < 3:
              return "deberías ponerle todas las vacunas"
            else:
              return "Tu gatito esta seguro"

        elif self.animal == "perro":
            if numero_vacunas < 5:
              return "deberías ponerle todas las vacunas"
            else:
              return "tu perrete esta seguro"

        elif self.animal == "pez":
            return "Tu mascota no necesita vacunas"

        else:
            return "todavía no sabemos cuantas vacunas necesita tu mascota"
```

Iterar por las funciones de la clase

Como ha llegado su cumple queremos modificar la edad.
Llamamos a la función cumple

felix.cumple()

Se ejecuta el código que hay dentro que modifica el atributo edad. Si ahora llamamos al atributo edad para conocer su valor

felix.edad → output: 2

La función vacunas recibía un parámetro, por lo que hay que especificarlo al llamar a la función

felix.vacunas(2) → output: 'deberías ponerle todas las vacunas'

felix.numero_vacunas(2) → output: 2

Para **conocer los atributos** de la clase:

felix.__dict__ → output: {'animal': 'gato',
'edad': 2,
'raza': 'gato común',
'hogar': 'domicilio',
'numero_vacunas': 2}

Herencia de clases

Para crear **nuevas clases a partir de las** que ya están **definidas**. Las clases hijas **heredan los atributos y métodos** de las clases madre.

Sintaxis:
class Perros (Mascota)

perro = Perros('perro', 15, 'coli')

Para **saber si un objeto es herencia** de una superclase **isinstance**(variable, Clase)
isinstance(perro, Mascota) → output: True

Para **ampliar una subclase** / clase hija, en el instructor .__init__ añadimos super() para indicar que son los mismos que los de la clase madre.

```
class Perros(Mascota):
  def __init__(self, animal, edad, raza):
    super().__init__(animal, edad, raza)
     # creamos el nuevo método al que llamaremos pasear

  def pasear(self, optimo_paseos = 5):
    num_paseos = int(input("Cuántos paseos le das a tu perro?"))
    print(f"Sacas {num_paseos} veces a tu perrete")

    if num_paseos < optimo_paseos:
      return "Saca al perrete!!!!"
    else:
      return "Que bien! Tu perrete sale mucho a la calle a correr!"
```

Al cambiar la clase hija, volvemos a crear la instancia

```
perro = Perros("perro", 15, "coli")
```

perro.pasear() → output: Sacas 7 veces a tu perrete
'Que bien! Tu perrete sale mucho a la calle a correr!'

Para **conocer** toda la **información** sobre una **clase:** print(help(Perros))