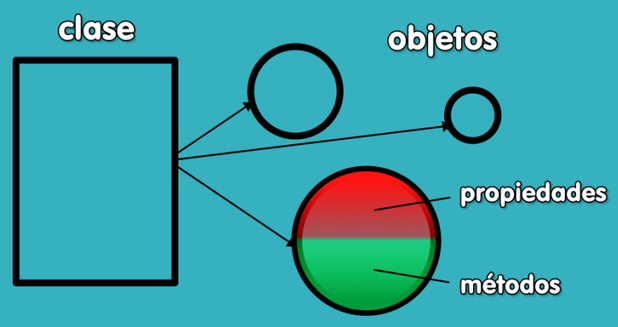
PROGRAMACIÓN ORIENTADA A OBJETOS EN PYTHON



POO: es un modelo de programación en el que el diseño de software se organiza alrededor de datos u objetos, en vez de usar funciones y lógica. Se enfoca en los objetos que los programadores necesitan manipular, en lugar de centrarse en la lógica necesaria para esa manipulación.

POLIMORFISMO

```
class Coche():

    ruedas=4

    def desplazamiento(self):
        print("El coche se esta desplazando sobre 4 ruedas")

class Moto():

    ruedas=2

    def desplazamiento(self):
        print("La moto se esta desplazando sobre 2 ruedas")
```

Es la capacidad que tienen los objetos de diferentes clases para usar un comportamiento o atributo del mismo nombre pero con diferente valor.

HERENCIA

Para aplicar la herencia en python debemos crear una súper clase o clase padre la cual tendrá los atributos y comportamientos principales que tendrán todas las clases derivadas de la clase padre

```
class Padre():

    caballo="negro"
    ojos="azules"

    def conducir_coche(self):
        print ("La persona sabe conducir coches")

class Hijo(Padre):

    def conducir_moto(self):
        print ("La persona sabe conducir moto")

persona=Hijo()

print(persona.caballo)

print(persona.ojos)

persona.conducir_coche()

persona.conducir_moto()
```

CLASES, OBJETOS Y METODOS

```
class Coche():

    ruedas=4

    def desplazamiento(self):
        print("El coche se esta desplazando sobre 4 ruedas")

miVehiculo=Coche()

print("Mi coche tiene ", miVehiculo.ruedas, " ruedas")

miVehiculo.desplazamiento()
```

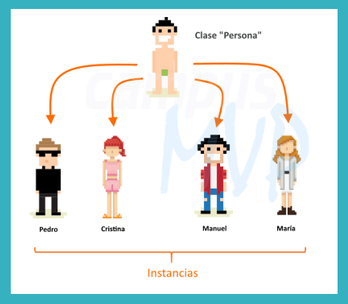
Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código,

ENCAPSULAMIENTO

```
class Ejemplo():

    def publico(self):
        return "Soy un método público, a la vista de todo"
    def __privado(self):
        return "Soy un metodo privado, para ti no existo"

objeto = Ejemplo()

print(objeto.publico())

print(objeto.__privado())
```

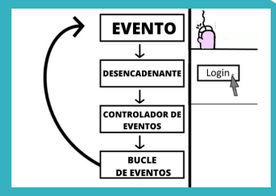
ABSTRACCIÓN

```python
import abc
from abc import ABC
# Declaramos nuestra clase
class Animal(ABC):
    # Asignar nombre
    def set_name(self, name):
        passS
    # Obtener nombre
    def get_name(self):
        pass
    # Definimos las propiedades
    name = abc.abstractproperty(get_name, set_name)
class Perro(Animal):
    def __init__(self):
      pass
    # Propiedad para obtener el nombre
    @property
    def name(self):
        return self._name
    # Propiedad para asignar el nombre
    @name.setter
    def name(self, name):
        self._name = name
perro = Perro() # Instancia de Perro
perro.name = "Marly" # Asignación del nombre
print(perro.name) # Obtenemos el valor de name
```



INSTANCIAS

```python
#instancias
class Perros (object):
    'Clase para los perros' #Descripcion
    Collar = True #Variable de clase Estática
    def __init__(self, salud, hambre):
        self.salud = salud #Variable de Instancia
        self.hambre = hambre #Variable de Instancia

Dogo = Perros(100, 50)
print(Dogo.hambre)
```



EVENTOS

```python
# Definir una variable
i = 5

# Ejecutar este ciclo mientras i es menor que 15
while i < 15:
    # Mostrar un mensaje
    print("¡Hola mundo!")
```

Acciones mediante las cuales el objeto reconoce que se está interactuando con él.
En este caso el valor de la variable i nunca se actualiza (siempre es 5). Por lo tanto, la condición i < 15 siempre es verdadera (True) y el ciclo nunca se detiene.

HTTPS://UNIVERSIDADEUROPEA.COM/BLOG/PROGRAMACION-ORIENTADA-OBJETOS/#:~:TEXT=LA%20PROGRAMACI%C3%B3N%20ORIENTADA%20A%20OBJETOS%20ES%20UN%20MODELO%20DE%20PROGRAMACI%C3%B3N,L%C3%B3GICA%20NECESARIA%20PARA%20ESA%20MANIPULACI%C3%B3N.
HTTPS://ES.WIKIPEDIA.ORG/WIKI/PYTHON
HTTPS://UNIPYTHON.COM/PROGRAMACION-ORIENTADA-OBJETOS-PYTHON/
HTTPS://PYTHONES.NET/VARIABLES-DE-CLASES/
HTTPS://PYTHONDIARIO.COM/2018/08/CLASES-ABSTRACTAS-EJEMPLOS-PRACTICOS.HTML
HTTPS://WWW.FREECODECAMP.ORG/ESPANOL/NEWS/PYTHON-TUTORIAL-CICLOS-WHILE/