

# Introducción al Lenguaje de Programación en Python Usando el paradigma POO



# ¿Qué es un sistema embebido?



1. Un sistema embebido (también conocido como “empotrado”, “incrustado” o “integrado”) es un sistema de computación diseñado para realizar funciones específicas, y cuyos componentes se encuentran integrados en una placa base (en inglés. “motherboard”). El procesamiento central del sistema **se lleva a cabo gracias a un microcontrolador**, es decir, un microprocesador que incluye además interfaces de entrada/salida, así como una memoria de tamaño reducido en el mismo chip.
2. Estos sistemas pueden ser programados directamente en el lenguaje ensamblador del **microcontrolador** o **microprocesador** o utilizando otros lenguajes como C o C++ mediante compiladores específicos.



¿Mencione 5 ejemplos de sistemas embebidos?

- Windows Phone 7 Nodo
- Windows Phone 7.5 Mango
- Windows Phone 7.5 Refresh
- Windows Phone 7.5 Tango
- Windows Phone 7.8



¿Menciona las diferencias o similitudes entre un sistema operativo, un sistema móvil y un sistema embebido?

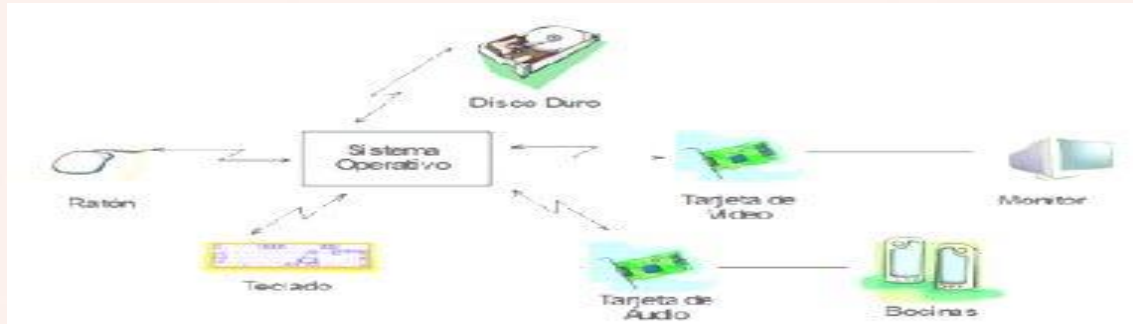




Figura 1: Interacción del Sistema Operativo con los periféricos (fuente: <http://www.comptechdoc.org/basic/basic/tut/osintro.html>).



Figura 2: Sistemas Operativos más comunes. (fuente: <http://fitecinformatica.wikispaces.com/Unidad+2+Sistemas+Operativos+%28S.O.%29>)



# ¿A que se referirán los términos MCU y MPU? Explique cada una de ellas.



Los **MCU** tienden a utilizarse en aplicaciones de ultra bajo consumo de energía, como controles remotos, sistemas eléctricos del consumidor y medidores inteligentes donde el énfasis del diseño está en la duración de la batería y muy poca o nada en la interacción de la UI. También se usan cuando se necesita una conducta altamente determinística.

Los **MPU** son ideales para aplicaciones del consumidor o industriales basadas en sistema operativo que pudieran incluir un uso intensivo de sistemas de computación y requerir conectividad múltiple de alta velocidad o de múltiples interfaces de usuario.





# ¿Cuáles son los pilares de POO?

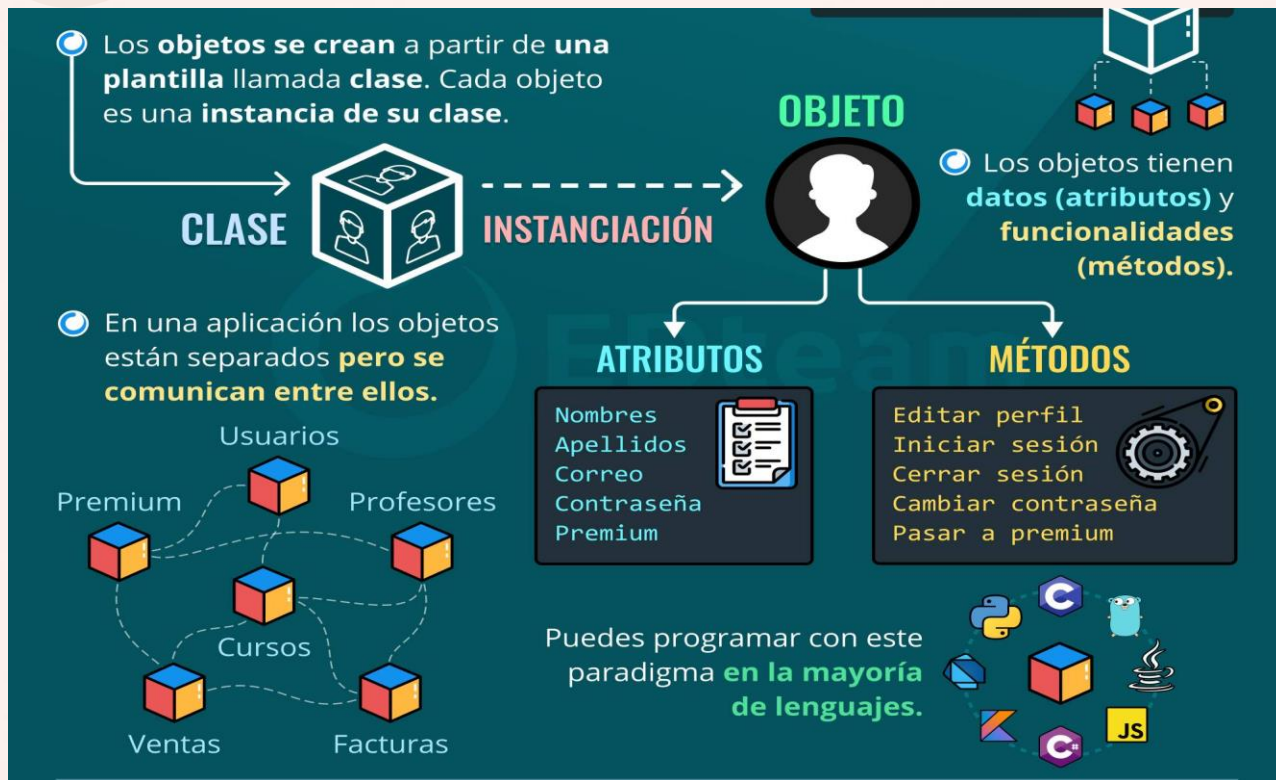


- Encapsulamiento.
- Abstracción.
- Herencia.
- Polimorfismo.





¿Mencione los componentes en lo que se basa POO?. Y explicar cada una de ellas.





# Defina los siguientes:



01

## Multiplataforma

Que puede utilizarse en diversos entornos o sistemas operativos.

03

## Multiparadigma

programas usando más de un estilo de programación

02

## Multipropósito

Cuando tienen muchas funcionalidades

04

## Lenguaje interpretado

Es el lenguaje cuyo código no necesita ser pre procesado mediante un compilador.



Defina a que se refiere cuando se habla de encapsulación y muestre un ejemplo (Código en Python).


```
class Clase:
    atributo_clase = "Hola" # Accesible desde el exterior
    __atributo_clase = "Hola" # No accesible

    # No accesible desde el exterior
    def __mi_metodo(self):
        print("Haz algo")
        self.__variable = 0

    # Accesible desde el exterior
    def metodo_normal(self):
        # El método si es accesible desde el interior
        self.__mi_metodo()

mi_clase = Clase()
#mi_clase.__atributo_clase # Error! El atributo no es accesible
#mi_clase.__mi_metodo()    # Error! El método no es accesible
mi_clase.atributo_clase    # Ok!
mi_clase.metodo_normal()   # Ok!
```

# Defina a que se refiere cuando se habla de herencia y muestre un ejemplo(Código en Python)?



```
class Animal:
    def __init__(self, especie, edad):
        self.especie = especie
        self.edad = edad

    # Método genérico pero con implementación particular
    def hablar(self):
        # Método vacío
        pass

    # Método genérico pero con implementación particular
    def moverse(self):
        # Método vacío
        pass

    # Método genérico con la misma implementación
    def describeme(self):
        print("Soy un Animal del tipo", type(self).__name__)
```

```
class Perro(Animal):
    def hablar(self):
        print("Guau!")
    def moverse(self):
        print("Caminando con 4 patas")

class Vaca(Animal):
    def hablar(self):
        print("Muuu!")
    def moverse(self):
        print("Caminando con 4 patas")

class Abeja(Animal):
    def hablar(self):
        print("Bzzzz!")
    def moverse(self):
        print("Volando")

    # Nuevo método
    def picar(self):
        print("Picar!")
```





# Defina los siguientes:



01

## Que es una Clase

Estas nos permiten agrupar un conjunto de variables y funciones que veremos a continuación.

02


## Que es un Objeto

Los objetos son aquellos que tienen propiedades (atributos) y comportamientos.

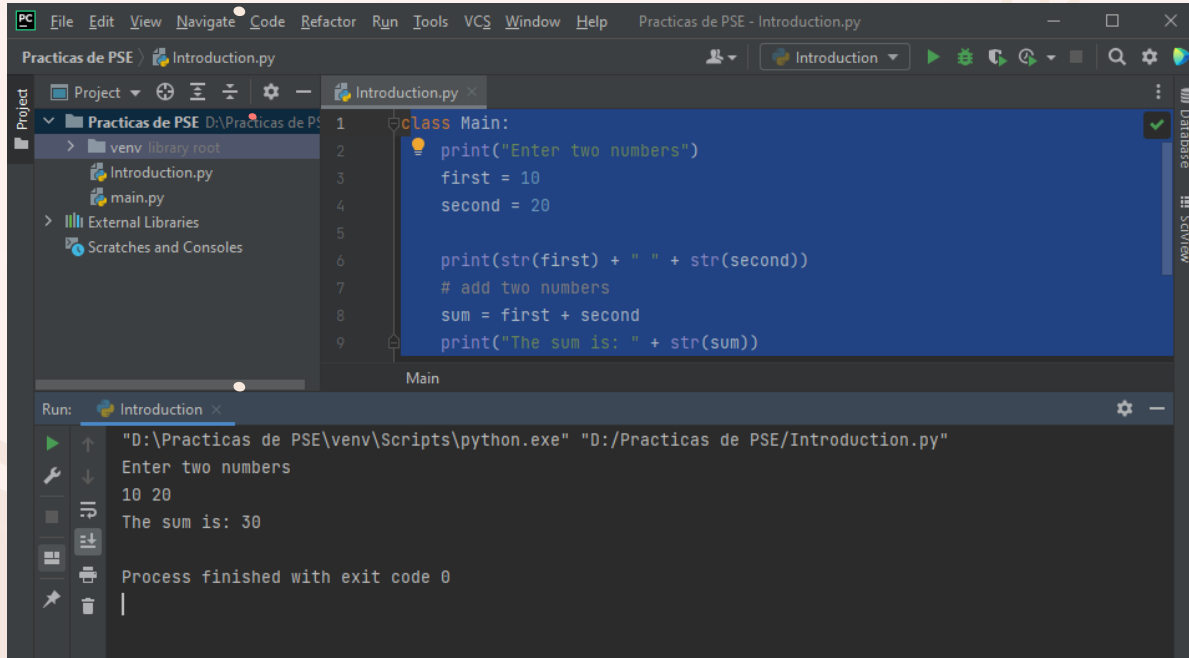
03

## Que es una instancia.

a todo objeto que derive de algún otro.



# 11. Llevar el siguiente código JAVA a Python.



The screenshot shows an IDE window titled "Practicas de PSE - Introduction.py". The editor displays a Python script with the following code:

```
1 class Main:
2     print("Enter two numbers")
3     first = 10
4     second = 20
5
6     print(str(first) + " " + str(second))
7     # add two numbers
8     sum = first + second
9     print("The sum is: " + str(sum))
```

The script is executed, and the output is shown in the Run console:

```
Run: Introduction x
"D:\Practicas de PSE\venv\Scripts\python.exe" "D:/Practicas de PSE/Introduction.py"
Enter two numbers
10 20
The sum is: 30
Process finished with exit code 0
```

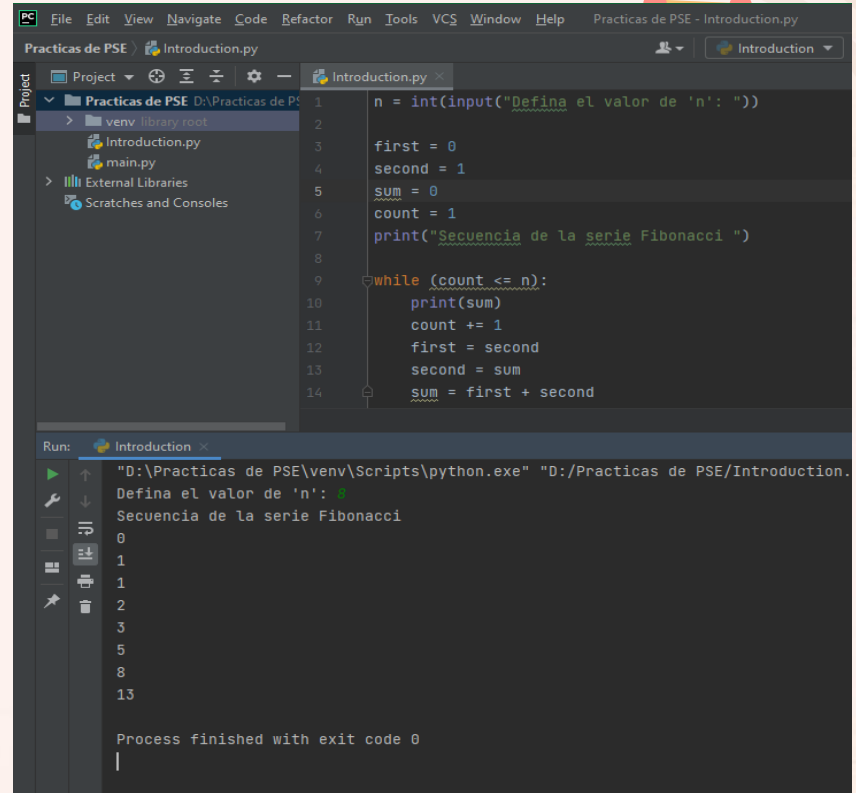
# Crear un programa Python que genere los primeros N números de la serie fibonacci.

- El programa tiene que leer un valor por consola.

■ Ejem:  $N = 8$

- Para el valor leído anteriormente, la salida debería ser:

■ 0, 1, 1, 2, 3, 5, 8, 13,



```
Practicas de PSE - Introduction.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help Practicas de PSE - Introduction.py

Project
  Project
  venv\library root
  Introduction.py
  main.py
  External Libraries
  Scratches and Consoles

1 n = int(input("Defina el valor de 'n': "))
2
3 first = 0
4 second = 1
5 sum = 0
6 count = 1
7 print("Secuencia de la serie Fibonacci ")
8
9 while (count <= n):
10     print(sum)
11     count += 1
12     first = second
13     second = sum
14     sum = first + second

Run: Introduction
"D:\Practicas de PSE\venv\Scripts\python.exe" "D:/Practicas de PSE/Introduction.py"
Defina el valor de 'n': 8
Secuencia de la serie Fibonacci
0
1
1
2
3
5
8
13

Process finished with exit code 0
```