**1. Defina los siguientes:**

* **¿Qué es un sistema embebido?**
* Un Sistema Embebido es un sistema electrónico diseñado para realizar pocas funciones en tiempo real. Ya que son diseñados para cubrir necesidades específicas  
  **2. Defina los ejem,plos: ¿Mencione 5 ejemplos de sistemas embebidos?**
* Sistemas GPS
* Cajeros automáticos
* Sistemas de control de salud (SmartWatch)
* Sistemas de calefacción.
* Sistemas de automatización de funciones ( domótica).

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

Las similitudes es que en base a componentes estas tres requieren de una ROM o RAM y que en la que se pueden complementar su uso con Software para la creación de otras funciones (JAVA), y en diferencias que el consumo de energía y la capacidad de funciones de estos varía mucho en función de las capacidades de un sistema embebido ya que este solo puede cumplir ciertas funciones o necesidades, en cambio un sistema operativo puede abarcar una gran cantidad de operaciones de todo tipo.

#### ****4. ¿A que se referirán los términos MCU y MPU?****

* **Explicar a que se refiera cada una de ellas.**
* **Mencione 2 ejemplos(components) de cada uno de ellos.**

MCU se refiere a microcontroladores y MPU a microprocesadores.

Originalmente, un MCU integraba CPU, memoria y periféricos en un solo chip. Pero actualmente, se le puede añadir memoria externa adicional, ya que los MCU son lo suficientemente potentes como para admitir aplicaciones más sofisticadas.

Ejemplos de MCU pueden ser los módulos de Arduino, ESP32, y los microcontroladores PIC16,17,18.

Ejemplos de MPU pueden ser los procesadores que desarrollaba intel y amd como Pentium,Celeron,Athlon.

**5. Defina cuáles son los pilares de POO y cuales los 6. Componentes en lo que se basa POO?**

**Abstracción:** Es cuando separamos los datos de un objeto para luego generar un molde (una clase).

1. **Encapsulamiento:** Lo puedes utilizar cuando deseas que ciertos métodos o propiedades sean inviolables o inalterables.
2. **Herencia:** Nos permite crear nuevas clases a partir de otras. Si tuviéramos una clase “Autos” y quisiéramos crear unas clases “Auto deportivo” o “Auto clásico”, podríamos tomar varias propiedades y métodos de la clase “Autos”. Esto nos da una jerarquía de padre e hijo.
3. **Polimorfismo:** Proviene de Poli = muchas, morfismo = formas. Se utiliza para crear métodos con el mismo nombre pero con diferente comportamiento.

6. Los componentes en los que se basa la POO son:

1. **Clases**: Las clases pueden ser definidas como un molde que contendrá todas las características y acciones con las cuales podemos construir N cantidad de objetos.
2. **Propiedades o atributos:** Las propiedades son las características de una clase, tomando como ejemplo la clase humanos, las propiedades podrían ser: nombre, el género, la altura, color de cabello, color de piel, etc.
3. **Métodos:** Los métodos son las acciones que una clase puede realizar, siguiendo el mismo ejemplo anterior, estas podrían ser: caminar, comer, dormir, soñar, respirar, nadar, etc.
4. **Objetos:** Son aquellos que tienen propiedades y comportamientos, estos pueden ser físicos o conceptuales.  
   Técnicamente, los objetos son instancias de una clase, vendría siendo cuando ya le colocamos un “nombre” a nuestras clases (moldes). Por ejemplo: El objeto “Arnell”, quien es una instancia de la clase humanos.

#### ****7. Defina los siguientes:****

* **Multiplataforma.**
* **Multiparadigma.**
* **Multipropósito.**
* **Lenguaje interpretado**

**Multiplataforma:** se usa para referirse a los sistemas operativos, lenguajes de programación, etc. pero estos pueden funciones en diversas plataformas de cada rama. (Ejemplos de programas de un sistema operativo Windows pueden funcionar correctamente en un sistema Linux).

**Multiparadigma:**se usa para decir que una forma de programación o paradigma, puede ser ejecutado en el lenguaje de programación  y soportarlo, también añadir más paradigmas que igualmente soportará ( Como el C++ puede combinar el POO con otros paradigmas).

**Multipropósito:**Un ejemplo en el Python es que puede ser usado para desarrollar tanto scripts sencillos como para desarrollar sitios web dinámicos, tanto en el ámbito educativo como el profesional.

**Lenguaje Interpretado:** Se dice que en un lenguaje es interpretado si el intérprete del lenguaje (o un programa que lo implementa) ejecuta las instrucciones una a una, sin hacer una compilación previa.

8. Encapsulación

Lo puedes utilizar cuando deseas que ciertos métodos o propiedades sean inviolables o inalterables.

9. Herencia.

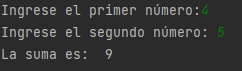
Lo que hereda ps

10. Clase- Objeto – instancia

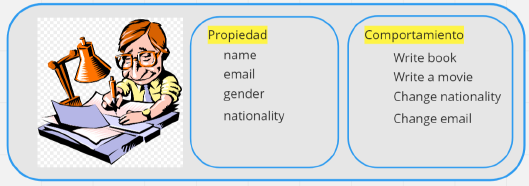
1. **Clases**: Las clases pueden ser definidas como un molde que contendrá todas las características y acciones con las cuales podemos construir N cantidad de objetos.
2. **Métodos:** Los métodos son las acciones que una clase puede realizar, siguiendo el mismo ejemplo anterior, estas podrían ser: caminar, comer, dormir, soñar, respirar, nadar, etc.
3. Instancia: **Instanciar** objetos es el proceso de generar un ejemplar de una **clase**, es decir, la **clase** es como una declaración de una forma y el objeto es un caso o elemento concreto **que** responde a esa forma. ... Podemos crear infinitos objetos a partir de una **clase**, cada uno sería una instancia de la **clase** o un ejemplar de esa **clase**.

11. Llevar el siguiente código JAVA a Python.

n1 = int(input("Ingrese el primer número:"))  
n2 = int(input("Ingrese el segundo número: "))  
sum = n1 + n2  
print("La suma es: ", sum)



12. Crear el código JAVA y Python para el siguiente análisis.

java

public class Persona

{

    private String nombre;

    private String email;

    private String genero;

    private String nationality;

    // constructor

    public Persona(String name ,String email,String genero,String nacionalidad)

    {

        this.nombre = name;

        this.email = email;

        this.genero = genero;

        this.nationality = nacionalidad;

    }

    //Métodos

    public void escribeLibro()

    {

        System.out.println("Escribió un libro");

    }

    public void escribePelicula()

    {

        System.out.println("Escribió una pelicula");

    }

    //Método para establecer nacionalidad

    public void setNationality(String nuevaNatio)

    {

        nationality = nuevaNatio;

    }

    //Método para establecer nuevo email

    public void setEmail(String nuevoEmail)

    {

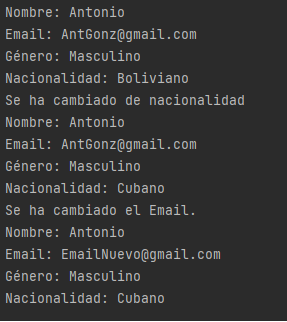
        email = nuevoEmail;

    }

}

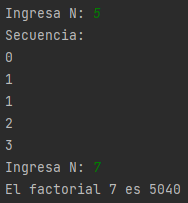
PYTHON:

class Persona:  
 \_\_name = None  
 \_\_email = None  
 \_\_gender = None  
 \_\_nationality = None  
  
 def \_\_init\_\_(self, name, email, gender, nationality):  
 self.\_\_name = name  
 self.\_\_email = email  
 self.\_\_gender = gender  
 self.\_\_nationality = nationality  
  
 #sobreescribe al metodo self y lo imprime  
 def \_\_str\_\_(self): ## \n salto de linea --> alt+92 = \  
 return f'Nombre: {self.\_\_name} \nEmail: {self.\_\_email} \nGénero: {self.\_\_gender} \nNacionalidad: {self.\_\_nationality}'  
 #encapsulacion  
 ## getters  
 def get\_nombre(self):  
 return self.\_\_name  
 ## setters  
 def set\_nombre(self, nuevo\_nombre):  
 self.\_\_name = nuevo\_nombre  
  
 ##Métodos:  
 def write\_book(self, name ):  
 print('Ha escrito un libro')  
  
 def write\_movie(self, name ):  
 print('Ha escrito una película')  
  
 def change\_nationality(self, new\_nationality ):  
 self.\_\_nationality = new\_nationality  
 print('Se ha cambiado de nacionalidad')  
  
 def change\_email(self, new\_email ):  
 self.\_\_email = new\_email  
 print('Se ha cambiado el Email.')  
  
  
persona1 = Persona('Antonio', 'AntGonz@gmail.com', 'Masculino','Boliviano')  
print(persona1)  
persona1.change\_nationality('Cubano')  
print(persona1)  
persona1.change\_email('EmailNuevo@gmail.com')  
print(persona1)

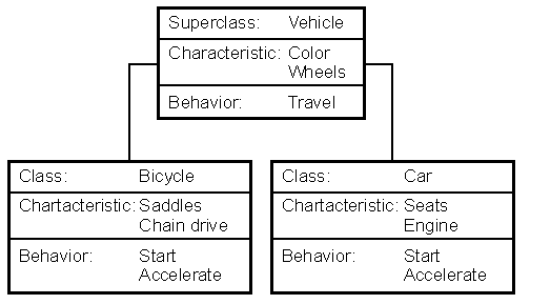


13.Crear un programa Python que genere los primeros N números de la serie Fibonacci.

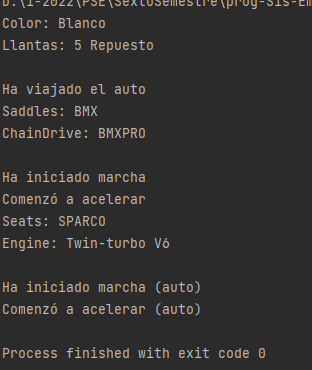
class Series\_and\_strings:  
  
 def \_\_init\_\_(self):  
 n = None  
  
 def fibonacci(self):  
 n = int(input("Ingresa N: "))  
 n1 = 0  
 n2 = 1  
 aux = 0  
 print("Secuencia:")  
 while aux < n:  
 print(n1)  
 r = n1 + n2  
 n1 = n2  
 n2 = r  
 aux += 1  
   
Series\_and\_strings1 = Series\_and\_strings()  
Series\_and\_strings1.fibonacci()



14.POO - Crear las clases necesarias para resolver el siguiente planteamiento.



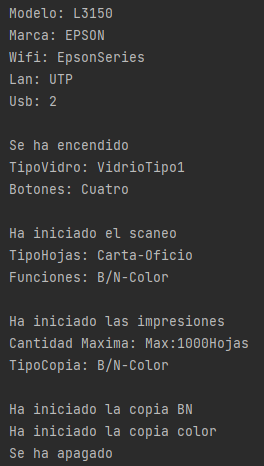
class Vehicle:  
 Color = None  
 Wheels = None  
  
 def \_\_init\_\_(self, color, wheels):  
 self.Color = color  
 self.Wheels = wheels  
  
 def \_\_str\_\_(self): ## \n salto de linea --> alt+92 = \  
 return f'Color: {self.Color} \nLlantas: {self.Wheels} \n'  
  
 def travel(self):  
 print("Ha viajado el auto")  
  
Vehicle1 = Vehicle('Blanco','5 Repuesto')  
print(Vehicle1)  
Vehicle1.travel()  
  
  
class Bicycle(Vehicle):  
 saddles = None  
 chain\_drive = None  
  
 def \_\_init\_\_(self, color,wheels,saddles,chain):  
 Vehicle.\_\_init\_\_(self, color, wheels)  
 self.saddles = saddles  
 self.chain\_drive = chain  
  
 def \_\_str\_\_(self): ## \n salto de linea --> alt+92 = \  
 return f'Saddles: {self.saddles} \nChainDrive: {self.chain\_drive} \n'  
  
 def Start(self):  
 print("Ha iniciado marcha")  
  
 def Accerelate(self):  
 print("Comenzó a acelerar")  
  
  
  
Bicycle1 = Bicycle('Rojo','2','BMX','BMXPRO')  
print(Bicycle1)  
Bicycle1.Start()  
Bicycle1.Accerelate()  
  
  
class Car(Vehicle):  
 seats = None  
 engine = None  
  
 def \_\_init\_\_(self, color, wheels, seats, engine):  
 Vehicle.\_\_init\_\_(self, color, wheels)  
 self.seats = seats  
 self.engine = engine  
  
 def \_\_str\_\_(self): ## \n salto de linea --> alt+92 = \  
 return f'Seats: {self.seats} \nEngine: {self.engine} \n'  
  
 def Start(self):  
 print("Ha iniciado marcha (auto)")  
  
 def Accerelate(self):  
 print("Comenzó a acelerar (auto)")  
  
Car1 = Car('Verde','6','SPARCO','Twin-turbo V6')  
print(Car1)  
Car1.Start()  
Car1.Accerelate()



15.Realizar un análisis para el siguiente escenario.



class PoweredDevice:  
 modelo = None  
 marca = None  
 wifi = None  
 lan = None  
 usb = None  
  
 def \_\_init\_\_(self, modelo, marca, wf, lan, usb):  
 self.modelo = modelo  
 self.marca = marca  
 self.wifi = wf  
 self.lan = lan  
 self.usb = usb  
  
 def \_\_str\_\_(self): ## \n salto de linea --> alt+92 = \  
 return f'Modelo: {self.modelo} \nMarca: {self.marca} \nWifi: {self.wifi} \nLan: {self.lan} \nUsb: {self.usb} \n'  
  
 def encender(self):  
 print("Se ha encendido")  
  
 def apagar(self):  
 print("Se ha apagado")  
  
PoweredDevice1 = PoweredDevice('L3150','EPSON','EpsonSeries','UTP','2')  
print(PoweredDevice1)  
PoweredDevice1.encender()  
  
class Scanner(PoweredDevice):  
 tipoVidrio = None  
 Botones = None  
  
 def \_\_init\_\_(self, modelo, marca, wf, lan, usb, tipo, btn):  
 PoweredDevice.\_\_init\_\_(self, modelo, marca, wf, lan, usb)  
 self.tipoVidrio = tipo  
 self.Botones = btn  
  
 def \_\_str\_\_(self): ## \n salto de linea --> alt+92 = \  
 return f'TipoVidro: {self.tipoVidrio} \nBotones: {self.Botones} \n'  
  
 def Scanear(self):  
 print("Ha iniciado el scaneo")  
  
Scanner1 = Scanner('Scan1','SacnnerpRo','NO','SI','SI','VidrioTipo1','Cuatro')  
print(Scanner1)  
Scanner1.Scanear()  
  
  
class Printer(PoweredDevice):  
 tipoHojas = None  
 Funciones = None  
  
 def \_\_init\_\_(self, modelo, marca, wf, lan, usb, tipo, fun):  
 PoweredDevice.\_\_init\_\_(self, modelo, marca, wf, lan, usb)  
 self.tipoHojas = tipo  
 self.Funciones = fun  
  
 def \_\_str\_\_(self): ## \n salto de linea --> alt+92 = \  
 return f'TipoHojas: {self.tipoHojas} \nFunciones: {self.Funciones} \n'  
  
 def Imprimir(self):  
 print("Ha iniciado las impresiones")  
  
  
Printer1 = Printer('Printeer1', 'PrintpRo', 'SI', 'SI', 'SI', 'Carta-Oficio', 'B/N-Color')  
print(Printer1)  
Printer1.Imprimir()  
  
  
class Copier(PoweredDevice):  
 cantiMax = None  
 tipoCopia = None  
  
 def \_\_init\_\_(self, modelo, marca, wf, lan, usb, canti, tipoC):  
 PoweredDevice.\_\_init\_\_(self, modelo, marca, wf, lan, usb)  
 self.cantiMax = canti  
 self.tipoCopia = tipoC  
  
 def \_\_str\_\_(self): ## \n salto de linea --> alt+92 = \  
 return f'Cantidad Maxima: {self.cantiMax} \nTipoCopia: {self.tipoCopia} \n'  
  
 def CopierBN(self):  
 print("Ha iniciado la copia BN")  
 def CopierColor(self):  
 print("Ha iniciado la copia color")  
  
  
Copier1 = Copier('Scan1', 'SacnnerpRo', 'NO', 'NO', 'NO', 'Max:1000Hojas', 'B/N-Color')  
print(Copier1)  
Copier1.CopierBN()  
Copier1.CopierColor()  
  
PoweredDevice1.apagar()



16.Ejercicio de planteamiento.

class Mascota:  
 nombreMascota = None  
 edadMascota = int  
 tipoMascota = None  
  
 # constructor  
 def \_\_init\_\_(self, name, edad, tipo):  
 self.nombreMascota = name  
 self.edadMascota = edad  
 self.tipoMascota = tipo  
  
 def \_\_str\_\_(self): ## \n salto de linea --> alt+92 = \  
 return f'NombreMasctoa: {self.nombreMascota} \nEdad: {self.edadMascota} \nTipoMascota:{self.tipoMascota}'  
 # encapsulacion  
 ## getters  
 def get\_nombre(self):  
 return self.nombreMascota  
  
 ## setters  
 def set\_nombre(self, nuevo\_nombre):  
 self.nombreMascota = nuevo\_nombre  
 ##instanciar  
  
Mascota1 = Mascota('Perro',10,'Canino')  
print(Mascota1)  
 ### herencia  
class Duenio(Mascota):  
 ci\_duenio = int  
 nombreDuenio = None  
 celularDuenio = int  
  
 def \_\_init\_\_(self, nameM, edadM, tipoM, ci, nameD, celD):  
 Mascota.\_\_init\_\_(self, nameM, edadM, tipoM)  
 self.ci\_duenio = ci  
 self.nombreDuenio = nameD  
 self.celularDuenio = celD  
  
 def \_\_str\_\_(self): ## \n salto de linea --> alt+92 = \  
 return f'ciD: {self.ci\_duenio} \nNombreDue: {self.nombreDuenio} \n Cel: {self.celularDuenio}\n'  
  
Duenio1 = Duenio('PerroPrueba',12,'Felino',2132,'Antonio',321)  
print(Duenio1)  
  
class HistorialClinico(Mascota):  
 nroHistorial = int  
 doctorHistorial = None  
 def \_\_init\_\_(self,nameM, edadM, tipoM, nroH, doctor):  
 Mascota.\_\_init\_\_(self, nameM, edadM, tipoM)  
 self.nroHistorial = nroH  
 self.doctorHistorial = doctor  
 def \_\_str\_\_(self):  
 return f'NombreMascota: {self.nombreMascota} \nEdad: {self.edadMascota} \nTipoMascota:{self.tipoMascota}\nNroHistorial:{self.nroHistorial}\nCel:{self.doctorHistorial}'  
  
  
HistorialClinico1 = HistorialClinico('Perro',9,'Canino',23,'RicardoPerez')  
print(HistorialClinico1)

