FACULTAD: INGENIERIA
CARRERA: INGENIERIA DE SISTEMAS
SEMESTRE: 6to
ASIGNATURA: PSE-611
NOMBRE: CORDOVA VARGAS HARRISON

MANEJO DE CONCEPTOS.

1. ¿Qué es un sistema embebido?

es un sistema de computación diseñado para realizar funciones específicas, y cuyos componentes se encuentran integrados en una placa base. 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. ¿Mencione 5 ejemplos de sistemas embebidos?

- Sistemas de calefacción central.
- Sistemas GPS.
- Rastreadores de fitness.
- Dispositivos médicos.
- Sistemas de automoción.
- Tránsito y cobro de tarifas

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

La diferencia entre un sistema operativo móvil (SO) y un sistema operativo de computadora tiene que ver con cómo las compañías tecnológicas individuales han implementado varias versiones de los sistemas operativos que proporcionan los entornos fundamentales para las aplicaciones de software tradicionales, así como las nuevas aplicaciones móviles

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

MCU: Podemos entender un microcontrolador como un computador dedicado**.** Cuando decimos que son computadores dedicados, nos referimos a la capacidad limitada que suelen tener. Son pequeños, con velocidad relativamente baja y un diseño sencillo y ligero. En nuestro

computador de casa tenemos el procesador, por un lado, la RAM por otro, etc.

MPU: Cuando hablamos de microprocesador**,** debemos tener en cuenta la evolución del término. Inicialmente, el procesador estaba formado por elementos independientes interconectados entre sí mediante buses. Por ejemplo, los registros, el oscilador que da la señal de clock, la ALU, todos eran componentes separados.

5. ¿Cuáles son los pilares de POO?

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

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

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.

Polimorfismo: Proviene de Poli = muchas, morfismo = formas. Se utiliza para crear métodos con el mismo nombre, pero con diferente comportamiento.

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

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.

Propiedades: 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.

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.

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.

En informática, se denomina multiplataforma a un atributo conferido a programas informáticos o métodos y conceptos de cómputo que son implementados, y operan internamente en múltiples plataformas informáticas.

Multiparadigma.

La Programación Multiparadigma es una práctica que emerge como resultado de la co-existencia de los paradigmas orientado a objetos, procedural, declarativo y funcional buscando **mejorar la producción en el desarrollo de proyectos.**

Multipropósito.

Que tiene muchos usos o que sirve para muchas cosas, esto en programación es muy fundamental, ya que como tiene varios propósitos se puede usar para muchas cosas a la hora de generar la programación.

Lenguaje interpretado

Un lenguaje interpretado es un lenguaje de programación para el que la mayoría de sus implementaciones ejecuta las instrucciones directamente, sin una previa compilación del programa a instrucciones en lenguaje máquina.

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

los datos miembros de un objeto de manera que solo se pueda cambiar mediante las operaciones definidas para ese objeto.

```
class Persona:
     __fullname: None
     __lastname: None
     __age: None

   def __init__(self, fullname, lastname, age):
      self.__fullname = fullname
      self.__lastname = lastname
      self.__age = age

   def __str__(self):
        return f"fullname:{self.__fullname}, \nlastname:{self.__lastname},
\nage:{self.__age}"
```

```
per1 = Persona('haris', 'Cordova', 22)
print(per1)
```

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

Es el mecanismo por el cual una clase permite heredar las características (atributos y métodos) de otra clase.

```
class Persona:
    __fullname: None
    __lastname: None
    __age: None

  def __init__(self, fullname, lastname, age):
     self.__fullname = fullname
     self.__lastname = lastname
     self.__age = age

  def __str__(self):
     return f'fullname:{self.__fullname}, \nlastname:{self.__lastname},
\nage:{self.__age}'

  def set_edad(self, nueva_edad):
     self.__age = nueva_edad

per1 = Persona('Harrison', 'Cordova', 20)
print(per1)

class Estudiante(Persona):
    __curso = None
    __email = None

def __init__(self, fullname, lastname, age, curso, email):
    Persona.__init__(self, fullname, lastname, age)
    self.__curso = curso
    self.__email = email

def __str__(self):
    return Persona.__str__(self) + f' \nCurso{self.__curso}
\nCorreo:{self.__email}'
```

```
def modifica_edad(self, nueva_edad):
    Persona.set_edad(self, nueva_edad)

est1 = Estudiante('haz', 'Cord', 20, '214-Lab comp 2', 'har@gmail.com')
print(est1)
est1.modifica_edad(50)
print(est1)
```

10.Defina los siguientes:

Que es una Clase

es una plantilla para la creación de objetos de datos según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos, como los sustantivos en el lenguaje. Cada clase es un modelo que define un conjunto de variables y métodos apropiados para operar con dichos datos.

Que es un Objeto

Un objeto es una unidad dentro de un programa informático que tiene un estado, y un comportamiento. Es decir, tiene una serie de datos almacenados y tareas que realiza con esos datos en el tiempo de ejecución. El objeto se puede crear instanciando clases

Que es una instancia.

Se llama instancia a todo objeto que derive de algún otro. De esta forma, todos los objetos son instancias de algún otro, menos la clase Object que es la madre de todas. Clases: Descripción de objeto. Consta de una serie de métodos y datos que resumen las características de este objeto.

11.Llevar el siguiente código JAVA a Python.

ejercicio.py

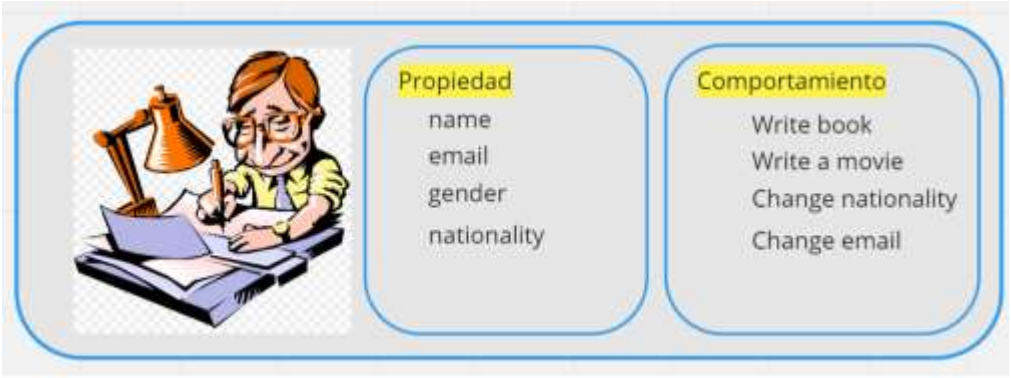
PROGRAMACION DE EMBEBIDOS > PRACTICA H2 > ejercicio.py > ...

```
1   class Main:
2       print("Enter two numbers")
3       first = 10
4       second = 20
5
6       print(str(first) + " " + str(second))
7
8       sum = first + second
9       print("The sum is: " + str(sum))
10
11
12
13
```

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN

```
The sum is: 30
PS C:\xampp\htdocs> & C:/Users/Usuario/AppData/Local/Programs/Python/Python310/python.exe "c:/xampp/htdocs/PROGRAMACION DE EMBEBIDOS/PRACTICA H2/ejercicio.py"
Enter two numbers
10 20
The sum is: 30
PS C:\xampp\htdocs>
```

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



```python
class Libro:
    nombre = None
    email = None
    genero = None
    nacionalidad = None

 def __init__(self, nombre, email, genero, nacionalidad):
    self.nombre = nombre
    self.email = email
    self.genero = genero
    self.nacionalidad = nacionalidad

 def __str__(self):
    return f'\nNombre: {self.nombre}, \nEmail: {self.email}, \nGenero:
{self.genero}, \nNacionalidad: {self.nacionalidad}'

 def write_book(self):
    return f'new book'

 def write_movie(self):
    return f'new movie'

 def edit_email(self, new_email):
    self.email = new_email

 def edit_nacionalidad(self, new_nacionalidad):
    self.nacionalidad = new_nacionalidad

lec1 = Libro('Mar', 'harrs@gmail.com', 'Masculino', 'ballivian')
print(lec1)
print('book:', lec1.write_book())
print('movie:', lec1.write_movie())
print('edit email')
lec1.edit_email('ras@gmail.com')
print(lec1)
print('edit nacionalidad')
lec1.edit_nacionalidad('chile')
print(lec1)
```

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

```
def fibonacci(num):
    arr = [0, 1]
    if num == 1:
        print('0')

    elif num == 2:
        print('[0,', '1]')
    else:
        while (len(arr) < num):
            arr.append(0)
        if (num == 0 or num == 1):
            return 1
    else:
        arr[0] = 0
        arr[1] = 1
        for i in range(2, num):
            arr[i] = arr[i - 1] + arr[i - 2]
        print(arr)

fibonacci(num = int(input("Ingrese el valor de N: ")))
```

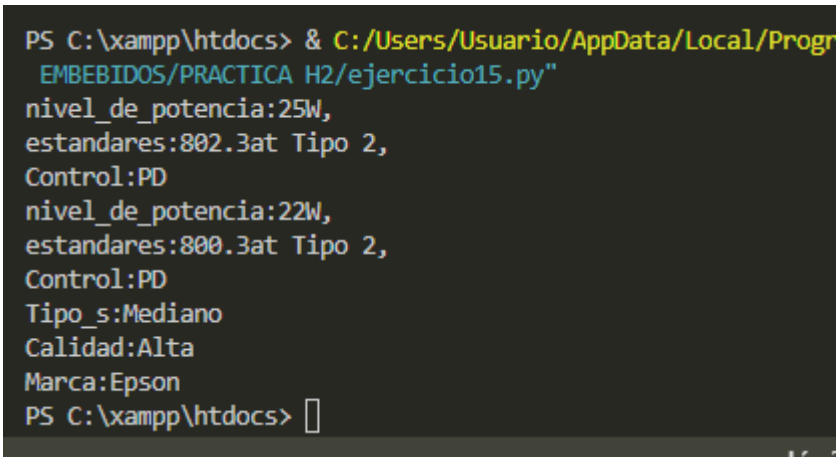
14.POO - Crear las clases necesarias para resolver el siguiente planteamiento.

```
class Vehiculo():
    color = None
    wheels = None

 def __init__(self, color, wheels):
     self.color = color
     self.wheels = wheels

 def __str__(self):
      return "Vehiculo Color: {}, Cantidad ruedas  {} ".format(self.color,
self.wheels)

class Car(Vehiculo):

 def __init__(self, color, wheels, seats, engine):
     Vehiculo.__init__(self, color, wheels)
     self.seats = seats
     self.engine = engine
```

```
 def _start_(self):
    print("Encendiendo Vehiculo")

 def _accelerate_(self):
     print("Acelerando Vehiculo")

 def __str__(self):
     return Vehiculo.__str__(self) + ", {} km/h, {} cc".format(self.seats,
self.engine,
self._start_(),


 c = Car("negro", 6, 200, 300)
 print(c)
 Car._start_
 Car._accelerate_

 class Bicycle(Vehiculo):
     def __init__(self, color, wheels, saddles, chain):
        Vehiculo.__init__(self, color, wheels)
        self.saddles = saddles
        self.chain = chain

     def _startb_(self):
        print("Iniciando Bicicleta")

     def _accelerateb_(self):
        print("Acelerando Bicicleta")

     def __str__(self):
        return Vehiculo.__str__(self) + ", {} sillas, {}
cond.".format(self.saddles, self.chain, self._startb_(), self._accelerateb_())


b = Bicycle("Amarillo", 3, 2, 50)
print(b)
Bicycle._startb_
Bicycle._accelerateb_
```

15.Realizar un análisis para el siguiente escenario.

```
class Cannon:
    potencia: None
    estandares: None
    marc: None

    def __init__(self, potencia, estandares, marc):
        self.potencia = potencia
        self.estandares = estandares
        self.marc = marc

    def __str__(self):
         return f'nivel_de_potencia:{self.potencia},
\nestandares:{self.estandares}, \nControl:{self.marc} '

    def set_edad(self, nueva_edad):
        self.age = nueva_edad

power1 = Cannon('25W', '802.3at Tipo 2', 'PD')
print(power1)

class Scanner(Cannon):

    tipo_s = None
    calidad = None
    modelo = None

    def __init__(self, potencia, estandares, control, tipo_s, calidad,
modelo):
        Cannon.__init__(self, potencia, estandares, control)
        self.tipo_s = tipo_s
        self.calidad = calidad
        self.modelo = modelo

    def __str__(self):
        return Cannon.__str__(self) + f' \nTipo_s:{self.tipo_s}
\nCalidad:{self.calidad} \nMarca:{self.modelo}'


est1 = Scanner('22W', '800.3at Tipo 2', 'PD', 'Mediano', 'Alta', 'Epson')
print(est1)
```


PS C:\xampp\htdocs> & C:/Users/Usuario/AppData/Local/Progr
EMBEBIDOS/PRACTICA H2/ejercicio15.py"
nivel_de_potencia:25W,
estandares:802.3at Tipo 2,
Control:PD
nivel_de_potencia:22W,
estandares:800.3at Tipo 2,
Control:PD
Tipo_s:Mediano
Calidad:Alta
Marca:Epson
PS C:\xampp\htdocs>