PRACTICA H2 SITEMAS EMBEBIDOS POO EN PYTHON

Universitario: Condori Ortiz Juan Mauricio

Docente: Ing. William Roddy Barra Paredes

Materia: Programación de Sistemas Embebidos

Sede: El Alto

Fecha: 05/04/2022

Manejo de conceptos.

1. ¿Qué es un sistema embebido?

R. Un sistema embebido es un sistema de computación diseñado para realizar funciones específicas, y cuyos componentes se encuentran integrados en una placa base.

2. ¿Mencione 5 ejemplos de sistemas embebidos?

R. Los 5 ejemplos son los siguientes:

- Sistemas de calefacción central.
- Sistemas GPS.
- Rastreadores de fitness.
- Dispositivos médicos.
- Sistemas de automoción.

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

R. 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. Mientras que el Sistema Embebido es embarcado o empotrado (integrado, incrustado) es un sistema de computación basado en un microprocesador o un microcontrolador. Esa es la principal diferencia entre estos 3.

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

R. MCU.- Un MCU es un microcontrolador de un solo chip, que se utiliza para controlar varios dispositivos. A diferencia de los microprocesadores de propósito general en los PCs, el MC se orienta a un número limitado de tareas y muchas veces a una sola tarea.

MPU.- Cuando hablamos de MPU 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.

Según se fue desarrollando la tecnología y la escala de integración, estos diferentes componentes se fueron fusionando dentro del mismo circuito. Así se pasó de tener un procesador formado por muchos circuitos integrados interconectados, a tener lo que denominamos microprocesador, que incorporaba todos estos elementos en un único circuito integrado.

5. ¿Cuáles son los pilares de POO?

R. Sus pilares son los siguientes:

Pilar 1: Encapsulación: La encapsulación es la característica de un lenguaje POO que permite que todo lo referente a un objeto quede aislado dentro de éste. Es decir, que todos los datos referentes a un objeto queden "encerrados" dentro de éste y sólo se puede acceder a ellos a través de los miembros que la clase proporcione (propiedades y métodos).

Pilar 2: Abstracción: Lo que implica es que la clase debe representar las características de la entidad hacia el mundo exterior, pero ocultando la complejidad que llevan aparejada. O sea, nos abstrae de la complejidad que haya dentro dándonos una serie de atributos y comportamientos (propiedades y funciones) que podemos usar sin preocuparnos de qué pasa por dentro cuando lo hagamos.

Pilar 3: Herencia: Desde el punto de vista de la genética, cuando una persona obtiene de sus padres ciertos rasgos (el color de los ojos o de la piel, una enfermedad genética, etc...) se dice que los hereda. Del mismo modo en POO cuando una clase hereda de otra obtiene todos los rasgos que tuviese la primera ósea el padre.

Pilar 4: Polimorfismo: En POO, el concepto de polimorfismo se refiere al hecho de que varios objetos de diferentes clases, pero con una base común, se pueden usar de manera indistinta, sin tener que saber de qué clase exacta son para poder hacerlo.

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

ellas.

Objeto.- Se trata de un ente abstracto usado en programación que permite separar los diferentes componentes de un programa, simplificando así su elaboración, depuración y posteriores mejoras.

Métodos.- Son aquellas funciones que permite efectuar el objeto y que nos rinden algún tipo de servicio durante el transcurso del programa. Determinan a su vez como va a responder el objeto cuando recibe un mensaje.

Eventos.- Son aquellas acciones mediante las cuales el objeto reconoce que se está interactuando con él.
De esta forma el objeto se activa y responde al evento según lo programado en su código.

Atributos.- Características que aplican al objeto solo en el caso en que el sea visible en pantalla por el usuario; entonces sus atributos son el aspecto que refleja, tanto en color, tamaño, posición, si está o no habilitado.

Ejemplo de objeto: una coche.

Se considera un coche un objeto, totalmente diferenciado del alquitrán donde está aparcado.

Sus atributos son su color, marca, modelo, número de matrícula, número de puertas.

Sus eventos todos aquellas acciones por las cuales si el coche tuviera vida propia reconocería que le estamos dando un uso, como abrir la puerta, girar el volante, embragar, abrir el capot,

Los métodos son todo aquello que nos ofrece el coche como hacer sonar una bocina cuando tocamos el claxon (evento), llevarnos por la carretera, reducir la velocidad al pisar el freno (evento),

Mensajes.- Aunque los objetos se han de diferenciar claramente en una aplicación, estos se han de poder comunicar para poder trabajar en conjunto y construir así aplicaciones. Esto se hace posible a través de lo que se denomina paso de mensajes. Cuando un objeto quiere comunicarse con otro lo que hace es enviarle un mensaje con los datos que desea transmitir.

En el símil del coche, al apretar el claxon, el objeto claxon envía un mensaje a la bocina indicándole que haga sonar cierto sonido.

La potencia de este sistema radica en que el objeto emisor no necesita saber la forma en que el objeto receptor va a realizar la acción. Simplemente este la ejecuta y el emisor se desentiende del como; de hecho ni le importa, solo tiene conocimiento de que se está realizando.

Para que todo esto sea posible es necesario una buena programación de los eventos y de los métodos de cada objeto.

El conjunto de mensajes a los que un objeto puede responder se denomina protocolo del objeto.

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. Definir clases permite trabajar con código reutilizable. Puesto que desde una clase se puede crear una instancia y así reutilizar el código escrito para esta si tener que volver a escribir el código para la instancia. La instancia toma el patrón de la clase padre. Sin embargo, las variables son independientes.

Herencia.- Mecanismo para compartir automáticamente métodos y datos entre clases, subclases y objetos.

Permite crear nuevas clases introduciendo las variaciones con respecto a su clase padre.

-*Herencia simple*: una subclase puede heredar datos y métodos de una clase simple así como añadir o sustraer ciertos comportamientos.

-*Herencia múltiple: posibilidad* de adquirir métodos y datos de varias clases simultáneamente.

Encapsulación.- Define el comportamiento de una clase u objeto que tiene dentro de él todo tipo de métodos y datos pero que solo es accesible mediante el paso de mensajes. y los datos a través de los métodos del objeto/clase.

Polimorfismo.- Los objetos responden a los mensajes que se les envían. Un mismo mensaje puede ser interpretado o dar paso a distintas acciones según que objeto es el destinatario.

Con este sistema el emisor se desentiende de los detalles de la ejecución (aunque el programador ha de saber en todo momento cuales son las consecuencias de ese mensaje).

7. Defina los siguientes:

Multiplataforma.- 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. El software multiplataforma puede dividirse en dos grandes tipos o clases: uno requiere una compilación individual para cada plataforma que le da soporte, y el otro se puede ejecutar directamente en cualquier plataforma, sin preparación especial, por ejemplo, el software escrito en un lenguaje interpretado o bytecode pre compilado portable, para los cuales los intérpretes o paquetes en tiempo de ejecución son componentes comunes o estándar de todas las plataformas.

Multiparadigma.- Un lenguaje de programación Multiparadigma es el cual soporta más de un paradigma de programación. Según lo describe Bjarne Stroustrup, permiten crear "programas usando más de un estilo de programación".

El objetivo en el diseño de estos lenguajes es permitir a los programadores utilizar el mejor paradigma para cada trabajo, admitiendo que ninguno resuelve todos los problemas de la forma más fácil y eficiente posible. Por ejemplo, lenguajes de programación como C++, Genie, Delphi, Visual Basic o PHP, combinan el paradigma imperativo con la orientación a objetos.

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).

En informática, se define al concepto de Encapsulamiento como el proceso que interviene en el momento en que se envían los datos a través de una determinada Red, de modo que se pueden ordenar, administrar y hasta verificar si han llegado a destino, en qué estado, o si ha sido eficiente la operación, referida comúnmente como Encapsulamiento de Datos.

La acción de Encapsulado, entonces, tiene dos etapas, una en la cual se asigna una Interfaz Pública al dato que está siendo enviado y a las operaciones que se pueden realizar con el mismo, y otra en la que se implementa un objeto, asignándole la propiedad de Accesos a través de la misma, o bien mediante una interrelación con otros Encapsulamientos que hayan sido transformados con la implementación de otro Objeto.

Ejemplo:

```python
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('Juan', 'Condori', 20)
print(per1)
```

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

Cuando hablamos de herencia en programación no nos referimos precisamente a que algún familiar lejano nos ha podido dejar una fortuna, ya nos gustaría. En realidad se trata de uno de los pilares fundamentales de la programación orientada a objetos. Es el mecanismo por el cual una clase permite heredar las características (atributos y métodos) de otra clase.

La herencia permite que se puedan definir nuevas clases basadas de unas ya existentes a fin de reutilizar el código, generando así una jerarquía de clases dentro de una aplicación. Si una clase deriva de otra, esta hereda sus atributos y métodos y puede añadir nuevos atributos, métodos o redefinir los heredados.

```python
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('Juan', 'Condori', 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('Juan', 'Condori', 20, '214-Lab comp 2', 'juanmauricio111222@gmail.com')
print(est1)
est1.modifica_edad(50)
print(est1)
```

10.Defina los siguientes:

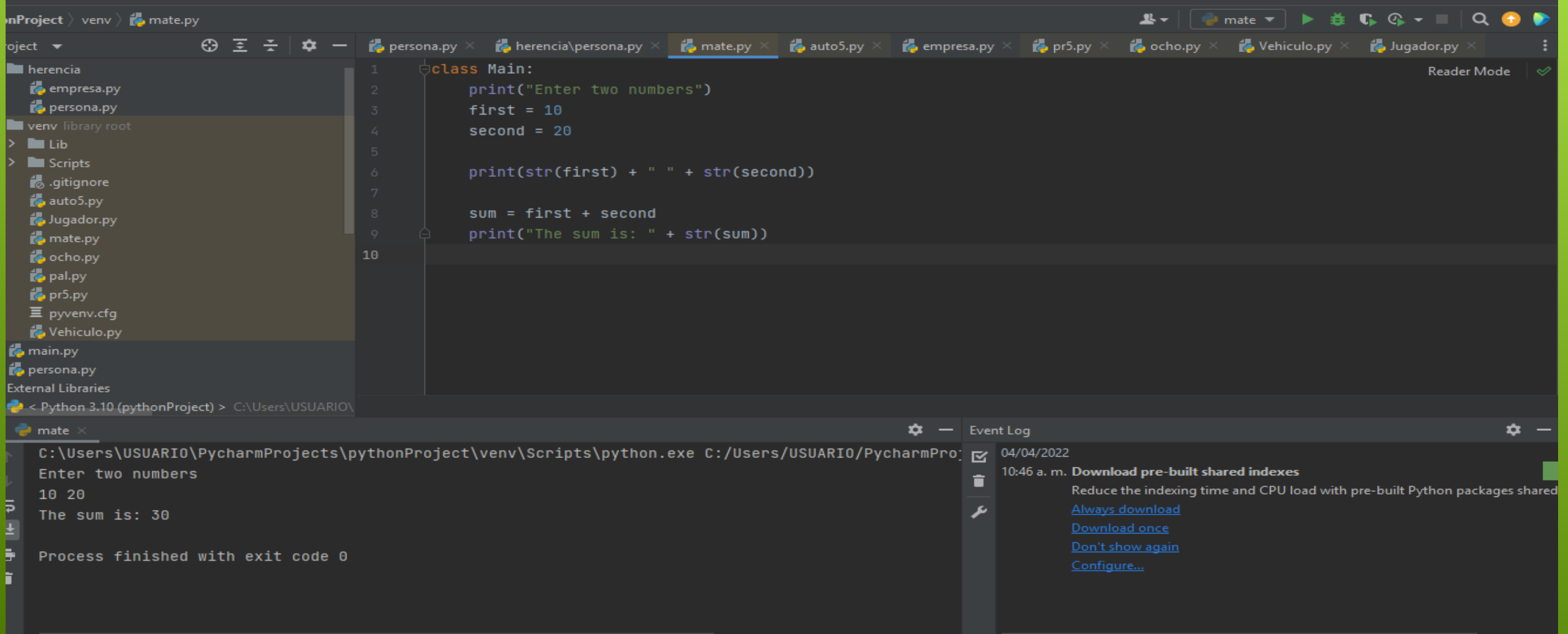
Que es una Clase: Una clase es una plantilla para la creación de objetos de datos según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos, como los sustantivos en el lenguaje. Cada clase es un modelo que define un conjunto de variables y métodos apropiados para operar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

```python
class Main:
    print("Enter two numbers")
    first = 10
    second = 20

    print(str(first) + " " + str(second))

    sum = first + second
    print("The sum is: " + str(sum))
```



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

```
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('Juan', 'juanmauricio111222@gmail.com', 'Masculino', 'Boliviana')
print(lec1)
print('book:', lec1.write_book())
print('movie:', lec1.write_movie())
print('edit email')
lec1.edit_email('fare@gmail.com')
print(lec1)
print('edit nacionalidad')
lec1.edit_nacionalidad('chile')
print(lec1)
```

```
Nombre: Juan,
Email: juanmauricio111222@gmail.com,
Genero: Masculino,
Nacionalidad: Boliviana
book: new book
movie: new movie
edit email

Nombre: Juan,
Email: fare@gmail.com,
Genero: Masculino,
Nacionalidad: chile

Process finished with exit code 0
```

```
class Libreria
{
  string name;
  string email;
  string genero;
  string nacionalidad;

  public Libreria(String name, String email, String genero, String nacionalidad)
    {
      this.name = name;
      this.email = email;
      this.genero = genero;
      this.nacionalidad = nacionalidad;
    }

  public String escribirBook()
    {
      return this.name + "escribir un libro.";
    }

  public String escribirMovie()
    {
      return this.name + "escribiendo una pelicula.";
    }

  public void changeNacionalidad(String nacionalidad)
    {
      this.nacionalidad = nacionalidad;
    }

  public void changeEmail(String email)
    {
      this.email = email;
    }
}
```

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: ")))
```


Run:
fibonacci
C:\Users\USUARIO\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/USUARIO/PycharmProjects/pythonProject/venv/fibonacci.py
Ingrese el valor de N: 8
[0, 1, 1, 2, 3, 5, 8, 13]
Process finished with exit code 0

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_(),
                                                 self._accelerate_())


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

```python
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_
```

Run: Vehiculo

```
Encendiendo Vehiculo
Acelerando Vehiculo
Vehiculo Color: negro, Cantidad ruedas  6 , 200 km/h, 300 cc
Iniciando Bicicleta
Acelerando Bicicleta
Vehiculo Color: Amarillo, Cantidad ruedas  3 , 2 sillas, 50 cond.

Process finished with exit code 0
```

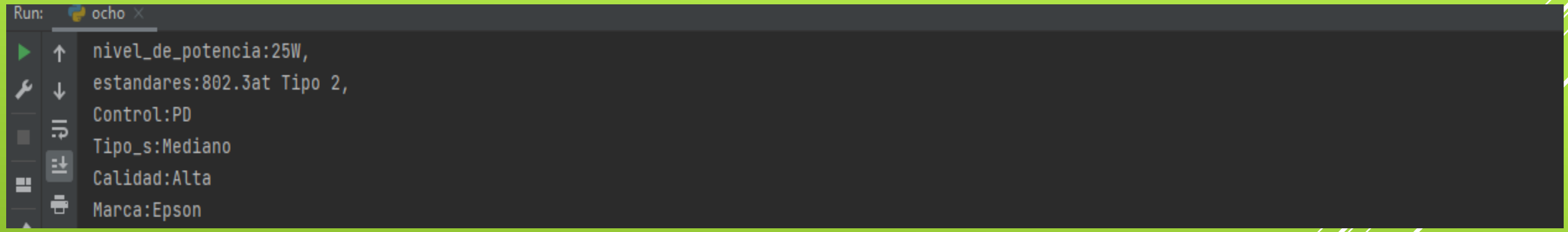
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Dow... (toda

Escribe aquí para buscar

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('25W', '802.3at Tipo 2', 'PD', 'Mediano', 'Alta', 'Epson')
print(est1)
```

```python
class Printer:

    Capacidad_hojas = None
    Mod = None
    Tipo_P = None

    def __init__(self, potencia, estandares, control, Capacidad_hojas, Mod, Tipo_P):
        Cannon.__init__(self, potencia, estandares, control)
        self.Capacidad_hojas = Capacidad_hojas
        self.Mod = Mod
        self.Tipo_P = Tipo_P

    def __str__(self):
        return poweredDevice.__str__(self) + f' \nCapacidad_hojas:{self.Capacidad_hojas} \nModelo:{self.Mod} \nTipo_P:{self.Tipo_P}'


pri1 = Printer('25W', '802.3at Tipo 2', 'PD', 200, 'epson', 'grande')
print(pri1)


class Copier:

    Color_C = None
    Marca_C = None
    Calidad_C = None

    def __init__(self, tipo_s, calidad, marca, Color_C, Marca_C, Calidad_C):
        Scanner.__init__(self, tipo_s, calidad, marca)
        self.Color_C = Color_C
        self.Marca_C = Marca_C
        self.Calidad_C = Calidad_C

    def __str__(self):
        return Scanner.__str__(self) + f' \nColor_C:{self.Color_C} \nMarca_C:{self.Marca_C} \nCalidad_C:{self.Calidad_C}'


cop1 = Copier('Mediano', 'Alta', 'Epson', 'Negro', 'epson', 'Baja')
print(cop1)
```



16.Ejercicio de planteamiento.

```
class Identidad:
    __cedula = None
    __nombre = None
    __apellido = None
    __sexo = None

    def __init__(self, __cedula, __nombre, __apellido, __sexo):
        self.__cedula = __cedula
        self.__nombre = __nombre
        self.__apellido = __apellido
        self.__sexo = __sexo

    def __str__(self):
        return f'\nCedula: {self.__cedula}, \nNombre: {self.__nombre}, \nApellido: {self.__apellido}, \nSexo: {self.__sexo} '
```

```
 def hablar(selfself, mensaje):
        return mensaje

    def getGenero(self, sexo):
        genero = ('Masculino', 'Femenino')
        if sexo == “M":
            return genero[0]
        elif sexo == "F":
            return genero[1]
        else:
            return "Desconocido"


iden1 = Identidad('6064980 LP', 'Juan', 'Condori', 'Maculino')
iden2 = Identidad('3456756 SCR', 'Maria', 'Mercedes', 'Femenino')
print(iden1)
print(iden2)


class Supervisor(Identidad):
    __rol = None
    __salario = None

    def __init__(self, __cedula, __nombre, __apellido, __sexo, __rol):
        Identidad.__init__(self, __cedula, __nombre, __apellido, __sexo)
        self.__rol = __rol

    def __str__(self):
        return Identidad.__str__(self) + f'\nCargo: {self.__rol}‘


sup1 = Supervisor('6064980 LP', 'Juan', 'Condori', 'Masculino', 'Presidente')
print(sup1)
```

Run: identidad

```
C:\Users\USUARIO\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/USUARIO/PycharmProjects/pythonProject/venv/identidad.py

Cedula: 6064980 LP,
Nombre: Juan,
Apellido: Condori,
Sexo: Maculino


Cedula: 3456756 SCR,
Nombre: Maria,
Apellido: Mercedes,
Sexo: Femenino


Cedula: 6064980 LP,
Nombre: Juan,
Apellido: Condori,
Sexo: Masculino
Cargo: Presidente

Process finished with exit code 0
```

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Do... (today 08:44 p. m.) 47:76 CRLF UTF-8 4 spaces Python 3.10 (pythonPr

Link del Video.-

https://drive.google.com/file/d/1wQbnEYle-UxtSPXrjGE4d58xqwkYJ4LM/view?usp=sharing