

**UNIVERSIDAD PRIVADA FRANZ TAMAYO**

**FACULTAD DE INGENIERÍA**

**CARRERA DE ING. DE SISTEMAS**



## **Tarea actividad final hito2**

**ESTUDIANTE:**

- **BLANCO CHAMBILLA BRAYAN ALVARO**

**DOCENTE: ING. WILLIAM RODDY BARRA PAREDES**

**ASIGNATURA: PROGRAMACIÓN DE SISTEMAS EMBEBIDOS**

## **Manejo de conceptos.**

### **1. ¿Qué es un sistema embebido?**

**R.** Es un sistema de computación basado en un microprocesador o un microcontrolador diseñado para realizar una o algunas pocas funciones dedicadas.

### **2. ¿Mencione 5 ejemplos de sistemas embebidos?**

**R.** Router, Lavadora, Cámara, Teléfono, Xbox.

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

**R.** Un sistema operativo es crucial en todo lo que tenga que ver con tecnología en la actualidad, un sistema móvil es un sistema operativo pero diseñado específicamente para operar en móviles y los sistemas embebidos son sistemas operativos de bajo nivel que acciona funciones necesarias para el funcionamiento específicos.

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

**R.**

- Los MCU(Microcontroladores) son un circuito integrado que contiene todos los componentes de un computador: CPU, ROM.
- Los MPU(Microprocesadores) son un chip que se encuentra integrado en la placa base y que se encarga de ejecutar las instrucciones que ordena el usuario: CPU, NÚCLEOS.

### **5. ¿Cuáles son los pilares de POO?**

**R.**

- **Encapsulación:** La encapsulación permite que todo lo referente a un objeto quede aislado dentro de éste.
- **Abstracción:** La clase debe representar las características de la entidad hacia el mundo exterior, pero ocultando la complejidad que llevan aparejada.
- **Herencia:** Es cuando una clase hereda de otra obtiene todos los rasgos que tuviese la primera.
- **Polimorfismo:** Varios objetos de diferentes clases, pero con una base común, se pueden usar de manera indistinta.

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

**R.**

- **Clase:** Una clase es la descripción de un conjunto de objetos similares.
- **Objeto:** Los objetos son las cosas que nos rodean, todos tienen características que pueden compartir o que los hacen diferentes unos de otros. Un objeto puede ser creado instanciando una clase.
- **Métodos:** Estas son las acciones que puede tener el objeto o los procesos que realizará.

## 7. Defina los siguientes:

- Multiplataforma: Hace referencia a los programas informáticos o métodos y conceptos de cómputo que son implementados, y operan internamente en múltiples plataformas informáticas.
- Multiparadigma: Es una práctica que emerge como resultado de los paradigmas orientados a objetos, procedural, declarativo y funcional buscando mejorar la producción en el desarrollo de proyectos.
- Multipropósito: Que se realiza más de una función en un solo dispositivo.
- 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).

**R.** La encapsulación consiste en negar el acceso a los atributos y métodos internos de la clase desde el exterior.

```
class Ejemplo:

    __atributo_privado = "Soy un atributo inalcanzable desde
fuera."

    atributo_publico = "Soy un atributo alcanzable desde fuera."

    def __metodo_privado(self):

        print("Soy un método inalcanzable desde fuera.")

    def metodo_publico(self):

        print("Soy un método alcanzable desde fuera.")

e = Ejemplo()

print(e.__atributo_privado)

print(e.atributo_publico)

e.__metodo_privado()

e.metodo_publico()
```

En caso de querer acceder al atributo y método privado saltara error:

```
Soy un atributo alcanzable desde fuera.
Soy un método alcanzable desde fuera.
Traceback (most recent call last):
  File "C:\Users\BRAYAN\PycharmProjects\pythonProject\persona.py", line 16, in <module>
    e.__metodo_privado()
AttributeError: 'Ejemplo' object has no attribute '__metodo_privado'

Process finished with exit code 1
```

```
Soy un atributo alcanzable desde fuera.
Soy un método alcanzable desde fuera.
Traceback (most recent call last):
  File "C:\Users\BRAYAN\PycharmProjects\pythonProject\persona.py", line 15, in <module>
    print(e.__atributo_privado)
AttributeError: 'Ejemplo' object has no attribute '__atributo_privado'

Process finished with exit code 1
```

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

**R.** La herencia es la capacidad que tiene una clase de heredar los atributos y métodos de otra, algo que nos permite reutilizar código y hacer programar mucho más óptimos.

```
class Producto:

    def __init__(self, referencia, nombre, descripcion):

        self.referencia = referencia

        self.nombre = nombre

        self.descripcion = descripcion

    def __str__(self):

        return f"REFERENCIA\t {self.referencia}\n" \

            f"NOMBRE\t\t {self.nombre}\n" \

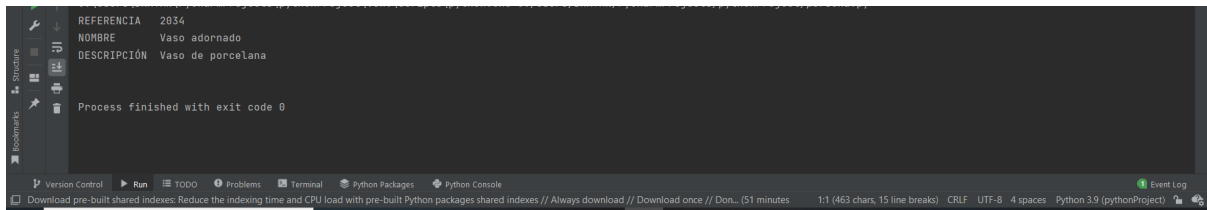
            f"DESCRIPCIÓN\t {self.descripcion}\n"

class Adorno(Producto):

    pass

adorno = Adorno(2034, "Vaso adornado", "Vaso de porcelana")

print(adorno)
```



## 10. Defina los siguientes:

- **Clase:** Una clase es la descripción de un conjunto de objetos similares.
- **Objeto:** Los objetos son las cosas que nos rodean, todos tienen características que pueden compartir o que los hacen diferentes unos de otros. Un objeto puede ser creado instanciando una clase.
- **Instancia:** Se llama instancia a todo objeto que derive de algún otro.

## 11. Llevar el siguiente código JAVA a Python.

```
print('Introduce 2 numeros: ')

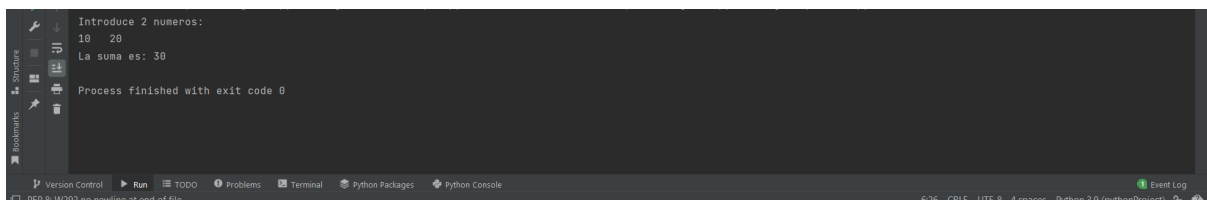
first = 10

second = 20

print(first, ' ', second)

sum = first + second

print('La suma es:', sum)
```



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

```
class Escritor:

    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

def Write_book(self):

    print('Escribio un libro exitosamente\n')

def Write_a_movie(self):

    print('Escribio una pelicula exitosamente\n')

def change_nationality(self, newNationality):

    self.nationality = newNationality

    print('Cambio de nacionalidad exitoso\n')

def change_email(self, newEmail):

    self.email = newEmail

    print('Cambio de gmail exitoso\n')

def __str__(self):

    return f'Nombres: {self.name}\nEmail: {self.email}\nGender: {self.gender}\nNationality: {self.nationality}\n'

escritor1 = Escritor('Brayan Blanco', 'braybe88@gmail.com',
'Masculino', 'Boliviano')

print(escritor1)

escritor1.Write_book()

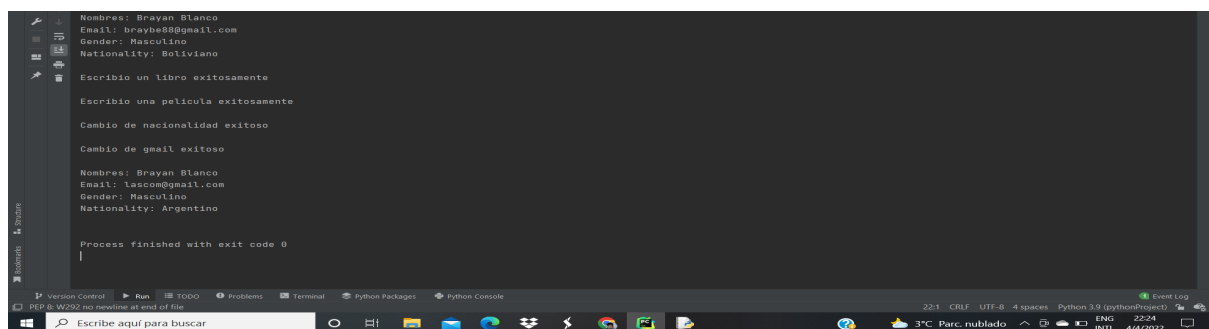
```

```
escritor1.Write_a_movie()
```

```
escritor1.change_nationality('Argentino')
```

```
escritor1.change_email('lascom@gmail.com')
```

```
print(escritor1)
```



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

```
a = 1
b = 0

imp = ''
total = 0

limit = int(input("Ingrese el valor de N: "))

if limit == 0:

    print('No hay nada en la serie fibonacci')

else:

    if limit == 1:

        print('0')

    else:

        imp = '0 '

        for i in range(limit - 1):
```

```

    total = a + b

    a = b

    b = total

    imp = imp + str(total) + ' '

print(imp)

```



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 travel(self):

        pass

    def __str__(self):

        return f'Color: {self.color}\nWheels: {self.wheels}\n'

veh1 = Vehicle('Rojo', 'Sport')

print(veh1)

```



```
class Bicycle(Vehicle):

    saddles = None

    chain_drive = None


    def __init__(self, color, wheels, saddles, chain_drive):

        Vehicle.__init__(self, color, wheels)

        self.saddles = saddles

        self.chain_drive = chain_drive


    def __str__(self):

        return Vehicle.__str__(self) + f'Saddles:
{self.saddles}\nChain drive: {self.chain_drive}\n'


    def start(self):

        print('go start\n')


    def accelerate(self):

        print('go accelerate\n')


bic1 = Bicycle('Amarillo', 'Titan', 'Cuero', 'Pedal')

print(bic1)

bic1.start()

bic1.accelerate()


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):
    return Vehicle.__str__(self) + f'Seats: {self.seats}\nEngine: {self.engine}\n'

def start(self):
    print('go start\n')

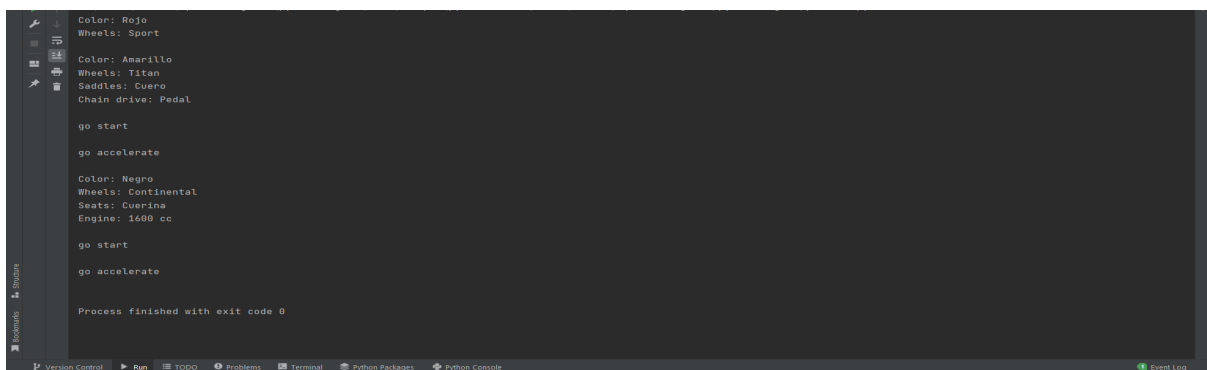
def accelerate(self):
    print('go accelerate\n')

car1 = Car('Negro', 'Continental', 'Cuerina', '1600 cc', )
print(car1)

car1.start()

car1.accelerate()

```



```

Color: Rojo
Wheels: Sport

Color: Amarillo
Wheels: Titan
Saddles: Cuero
Chain drive: Pedal

go start

go accelerate

Color: Negro
Wheels: Continental
Seats: Cuerina
Engine: 1600 cc

go start

go accelerate

Process finished with exit code 0

```

15. Realizar un análisis para el siguiente escenario.

- En la actualidad tenemos equipos electrónicos como impresoras, scanners, fotocopadoras, etc. Sin embargo también existen equipos electrónicos multifunción, como por ejemplo es posible tener una impresora y un scanner al mismo tiempo. (Véase imagen siguiente)

```
class Equipo:

    id_equipo = None

    consumo = None

    puerto = None

    def __init__(self, id_equipo, consumo, puerto):

        self.id_equipo = id_equipo

        self.consumo = consumo

        self.puerto = puerto

    def __str__(self):

        return f'ID_EQUIPO: {self.idEquipo}\nConsumo: {self.consumo}\nPuerto: {self.puerto}\n'


class Copier(Equipo):

    def copiar_hojas(self):

        print('se esta copiando')


class Scanner(Copier):

    def escanear_hojas(self):

        print('se esta escaneando')


class Printer(Copier):

    def imprimir_hojas(self):
```

```

        print('se esta imprimiendo')

class PoweredDevice(Scanner, Printer):

    def total_funcion(self):

        print('tengo todas las funciones')

pw1 = PoweredDevice('pw1', '300 V', 'USB')

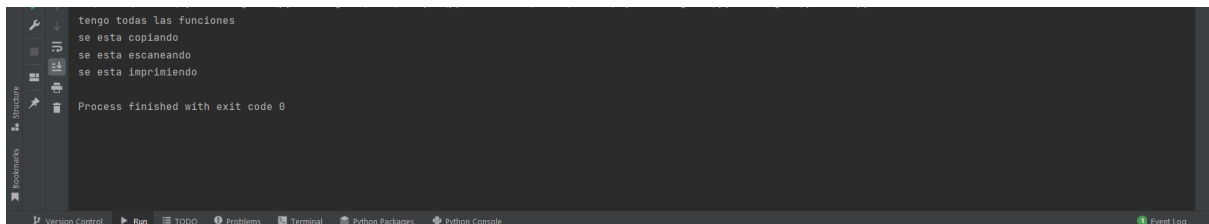
pw1.total_funcion()

pw1.copiar_hojas()

pw1.escanear_hojas()

pw1.imprimir_hojas()

```



## 16.Ejercicio de planteamiento.

```

class Persona:
    def __init__(self, cedula, nombre, apellido, sexo):
        self.__cedula = cedula
        self.__nombre = nombre
        self.__apellido = apellido
        self.__sexo = sexo

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

personal = Persona("13458796", "Leonardo", "Caballero", "M")
print(personal)

```

```

class Supervisor(Persona):
    def __init__(self, cedula, nombre, apellido, sexo, rol,
tareas):
        Persona.__init__(self, cedula, nombre, apellido, sexo)
        self.__rol = rol
        self.__tareas = tareas

    def __str__(self):
        return Persona.__str__(self) + f'Rol:
{self.__rol}\nTareas: {self.__tareas}\n'

```

```

sup1= Supervisor("13019753", "Brayan", "Blanco", "M",
"Gerente", 'Pendientes')
print(sup1)

```

```

class Destreza:
    def __init__(self, area, herramienta, experiencia):
        self.__area = area
        self.__herramienta = herramienta
        self.__experiencia = experiencia

    def __str__(self):
        return f'Area: {self.__area}\nHerramienta:
{self.__herramienta}\nExperiencia: {self.__experiencia}\n'

```

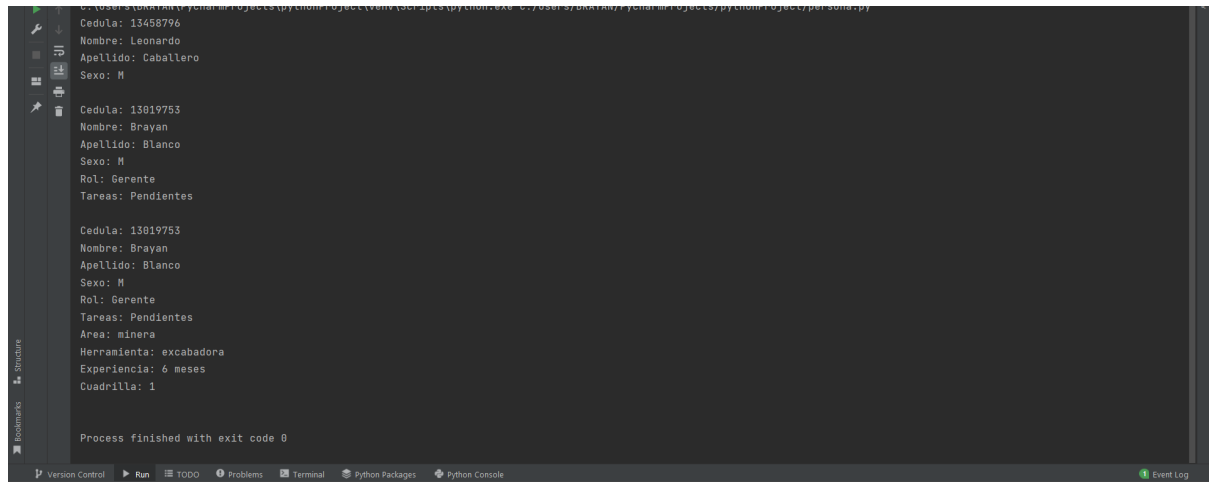
```

class JefeCuadrilla(Supervisor, Destreza):
    def __init__(self, cedula, nombre, apellido, sexo, rol,
tareas, area, herramienta, experiencia, cuadrilla):
        Supervisor.__init__(self, cedula, nombre, apellido,
sexo, rol, tareas)
        Destreza.__init__(self, area, herramienta, experiencia)
        self.__cuadrilla = cuadrilla

    def __str__(self):
        return Supervisor.__str__(self) +
Destreza.__str__(self) + f'Cuadrilla: {self.__cuadrilla}\n'

```

```
jcl = JefeCuadrilla("13019753", "Brayan", "Blanco", "M",  
"Gerente", 'Pendientes', 'minera', 'excabadora', '6 meses',  
'1')  
print(jcl)
```



The screenshot shows a Python IDE with a dark theme. The left sidebar contains icons for Explorer, Search, and Run and Debug. The main editor area displays the output of the code execution, showing the attributes of the JefeCuadrilla object. The output is organized into three sections, each starting with 'Cedula: 13019753'. The first section lists 'Nombre: Leonardo', 'Apellido: Caballero', and 'Sexo: M'. The second section lists 'Nombre: Brayan', 'Apellido: Blanco', 'Sexo: M', 'Rol: Gerente', and 'Tareas: Pendientes'. The third section lists 'Nombre: Brayan', 'Apellido: Blanco', 'Sexo: M', 'Rol: Gerente', 'Tareas: Pendientes', 'Area: minera', 'Herramienta: excabadora', 'Experiencia: 6 meses', and 'Cuadrilla: 1'. At the bottom, it states 'Process finished with exit code 0'. The bottom status bar shows 'Version Control', 'Run', 'TODO', 'Problems', 'Terminal', 'Python Packages', 'Python Console', and 'Event Log'.

```
Cedula: 13019753  
Nombre: Leonardo  
Apellido: Caballero  
Sexo: M  
  
Cedula: 13019753  
Nombre: Brayan  
Apellido: Blanco  
Sexo: M  
Rol: Gerente  
Tareas: Pendientes  
  
Cedula: 13019753  
Nombre: Brayan  
Apellido: Blanco  
Sexo: M  
Rol: Gerente  
Tareas: Pendientes  
Area: minera  
Herramienta: excabadora  
Experiencia: 6 meses  
Cuadrilla: 1  
  
Process finished with exit code 0
```