Unidad 1. Programación orientada a objetos

Taller entregable 1

2024-2

Es requerido que todo sea realizado utilizando programación orientada a objetos de python3, todo concepto de POO que se utilice será válido.

El trabajo podrá ser enviado en parejas, al enviar el trabajo no olviden mencionar a su compañero, en caso contrario se le calificará como si hubiera sido entregado por la persona que subió el archivo únicamente.

Se recomienda hacer entrega del informe en un archivo formato ipynb utilizando el markdown como textos por facilidad de calificación, u opcionalmente mandar el informe en pdf y el código en .py.

Entregable

Objetivo: Desarrollar un sistema de gestión de implantes médicos utilizando programación orientada a objetos (POO) que permita administrar información relevante sobre diversos tipos de implantes biomédicos.

Procedimiento:

Los puntos 1-3 son preguntas/análisis, el 4 es un código

1) Responder las siguientes preguntas (20%, 5% cada uno):
   - ¿Qué es una clase y que es un objeto en Python? Explique las diferencias y muestre como se define una clase.
   - Explique la herencia en la programación orientada a objetos.
   - ¿Qué es la encapsulación en programación orientada a objetos? ¿Por qué utilizar esto?
   - Defina la abstracción en la programación orientada a objetos en Python, ¿cómo se implementa en Python?

2) Verdadero o falso y justificar (No tiene que ser una justificación larga) (10%, 2.5% cada uno):
   - Las clases en Python no pueden heredar de más de una clase a la vez (No soportan herencia múltiple).
   - La herencia en Python hereda métodos, pero no las propiedades.

- Al heredar una clase en Python, la clase hija se puede modificar los métodos que fueron definidos en la clase padre para funcionar diferente en la clase hija.
- La función __init__() no es necesario ejecutarla en todos los casos cuando se define una clase hija.

3) Observe el siguiente código y respondas las preguntas (30%, 6% por pregunta):

```
class Persona:
    def __init__(self, nombre: str, apellido: str):
        self.nombre = nombre
        self.apellido = apellido

    def __str__(self):
        return f'{self.nombre} {self.apellido}'

class Profesor(Persona):
    def __init__(self, nombre: str, apellido: str, materia: str):
        super().__init__(nombre, apellido)
        self.__materia = materia

    @property
    def materia(self):
        return self.__materia

    @materia.setter
    def materia(self, materia):
        self.__materia = materia

    def __str__(self):
        return f'{super().__str__()} es profesor de {self.materia}'

class Estudiante(Persona):
    def __init__(self, nombre: str, apellido: str, nivel: str):
        super().__init__(nombre, apellido)
        self.nivel = nivel

    def __str__(self):
        return f'{super().__str__()} está en el nivel {self.nivel}'

estudiante = Estudiante("Andres", "Banquez", 10)
profesor = Profesor("Jose", "Alberto", "Matematicas")
print(estudiante)
```

```
print(profesor)
profesor.materia = "Ciencias"
print(profesor)

El resultado es:
Andres Banquez está en el nivel 10
Jose Alberto es profesor de Matematicas
Jose Alberto es profesor de Ciencias
```

- ¿Qué clases son padres y que clases son hijas? ¿Qué es lo que se hereda de las clases padres a las hijas en este caso?
- ¿Por qué al utilizar la función print en los objetos estudiante y profesor, se imprime la información ingresada y no algo diferente?
- ¿Por qué a pesar de que "materia" tiene un encapsulamiento de tipo privado se puede hacer el cambio de la materia con profesor.materia = ""?.
- Explique la función del decorador @property en el código.
- Explique la función del decorador @materia.setter en el código.

4) Realice un código que pueda realizar lo siguiente (40%):

El .txt adjunto contiene información de reportes de monitores multiparamétricos, estos reportes tendrán la siguiente información:

1. Zona de uso del equipo
2. Hora del reporte
3. Frecuencia Cardiaca
4. Presión Arterial
5. SpO2
6. Temperatura Corporal
7. Frecuencia respiratoria

Se repartirá de la siguiente manera:

1|2|DatosRecolectados^3^4^5^6^7

Se deberá crear una clase tipo reporte que deberá de utilizarse para guardar la información de cada línea del .txt y posteriormente utilizando el método __str__() se imprimirá toda la información de cada uno de los reportes.

Toda la información debe de estar encapsulada de manera privada y esta debe tener la opción de obtener la información, pero no de modificarla.

Cualquier duda por favor escribir al monitor andres.banquez@udea.edu.co