Coordinación de
Educación Abierta y a Distancia
VICERRECTORADO ACADÉMICO

Unach
UNIVERSIDAD NACIONAL DE CHIMBORAZO Online

PROGRAMACIÓN 2

Actividad Autónoma 1

Unidad 1: Fundamentos de la Programación Orientada a Objetos (POO)

Tema 1: Introducción a la POO, Clases y Objetos

FACULTAD DE
Ingeniería

Nombres:

Fecha:

Carrera: Ciencia de Datos

Periodo académico:

Semestre:

Objetivo de la actividad:

Diseñar e implementar una clase orientada a objetos que modele instancias del dataset de Iris, aplicando principios de encapsulamiento y representando la solución con un diagrama UML.

Recursos o temas que debe haber estudiado antes de hacer la actividad:

Conceptos fundamentales de Programación Orientada a Objetos (POO) en Python.

Definición y uso de clases y objetos.

Uso de atributos privados y encapsulamiento mediante métodos @property y @setter.

Implementación y utilidad del método mágico __init__().

Formato y propósito del método __repr__() para representar objetos.

Estructura básica de un diagrama UML de clases.

Sintaxis básica de Python en Jupyter Notebook.

Formato de entrega: PDF (máximo 5MB)

- Formato PDF, Generado en Jupyter Notebook (máximo 5MB).
- Además, subir Jupyer a la nube de GitHub Classroom.

Instrucciones:

1. Implementa la clase Sample en Python, que represente una flor del dataset Iris, con los siguientes atributos:
    - sepal_length (float)
    - sepal_width (float)
    - petal_length (float)
    - petal_width (float)
    - species (str) — este atributo debe estar **encapsulado**
2. Utiliza métodos @property y @setter para acceder y modificar el atributo species.
3. Agrega los siguientes métodos a la clase:
    - __init__() para inicializar todos los atributos.

- __repr__() para mostrar los datos del objeto de forma clara y concisa.
- describe() para imprimir las medidas de la flor en una oración comprensible.
- matches(species) que retorna True si la especie coincide con el valor dado.

4. Diseña un **diagrama UML** que incluya (Utiliza cualquier herramienta web como Diagram.io) :
   - Todos los atributos con su tipo y visibilidad (+ público, - privado)
   - Todos los métodos definidos en la clase
5. Responde brevemente las siguientes preguntas (Responde en el Notebook con Markdown):
   - ¿Qué significa **encapsular un atributo**?
   - ¿Por qué es importante encapsular species en lugar de dejarlo público?
   - ¿Qué ventajas ofrece el uso de @property sobre acceder directamente al atributo?

Bibliografía:

- Lott, S. F., & Phillips, D. (2023). *Python Object-Oriented Programming: Build robust and maintainable object-oriented Python applications and libraries* (4th Edition). Packt Publishing.
- Zelle, J. M. (2017). *Python Programming: An Introduction to Computer Science* (3rd Edition). Franklin, Beedle & Associates Inc.

Rúbrica de evaluación

Componente de aprendizaje:	**Autónomo**	X	**Contacto con el Docente**	
Nombre de la Unidad:	Unidad 1: Fundamentos de la Programación Orientada a Objetos (POO)			
Resultado(s) de aprendizaje:	• Comprender y aplicar el concepto de encapsulamiento para proteger los atributos de una clase en Python. • Implementar métodos getters y setters para el acceso controlado a los atributos privados. • Crear clases y objetos que representen datos reales, utilizando la Programación Orientada a Objetos (POO). • Documentar el código de manera clara, incluyendo comentarios que expliquen cada sección. • Representar gráficamente clases mediante un diagrama UML, indicando visibilidad de atributos y métodos. en POO.			
Nombre de la Actividad:	Encapsulamiento en el diseño de la clase Sample para el dataset de Iris			

	Escala de Valoración						
Criterios de Evaluación	**Excelente (10 - 9,1)**	**Bueno (9 - 8,1)**	**Satisfactorio (8 - 7)**	**Necesita mejorar (6,9 - 0,1)**	**No entrega (0)**	**Puntaje**	**Comentarios (SIGEA)**
1. Correctitud del código	Cumple completamente con las instrucciones y no presenta errores.	Cumple con la mayoría de las instrucciones, tiene pocos errores.	Cumple algunas instrucciones, pero presenta varios errores.	Cumple mínimamente con las instrucciones, tiene muchos errores.	No se presenta el código.		
2. Uso de encapsulamiento	Todos los atributos encapsulados correctamente y métodos getters/setters funcionan sin errores.	La mayoría de los atributos están encapsulados y los getters/setters funcionan bien.	Algunos atributos no están encapsulados adecuadamente o los getters/setters presentan problemas.	Encapsulamiento no implementado correctamente, getters/setters fallan.	No se presenta el encapsulamiento.		
3. Claridad y orden del código	Código claro, legible, bien organizado, con nombres descriptivos para	Código comprensible, cierta organización, áreas que podría	Código poco claro, carece de organización adecuada.	Código difícil de seguir, carece de estructura lógica.	No se presenta el código o es incomprensible.		

	variables y métodos.	n mejora r.					
4. Documentación / Comentarios	Todos los métodos y atributos están debidamente comentados, explicando su función.	La mayoría de los métodos y atributos tienen comentarios explicativos.	Algunos métodos o atributos tienen comentarios, pero no son suficientes o claros.	Código tiene muy pocos comentarios, lo que dificulta su comprensión.	No se presentan comentarios en el código.		
5. Cumplimiento de instrucciones	Se cumplen todas las instrucciones especificadas en la actividad.	Se cumplen la mayoría de las instrucciones especificadas.	Se cumplen parcialmente las instrucciones especificadas.	Se cumple con pocas instrucciones de la actividad.	No se cumple con ninguna instrucción.		

Puntaje total	

Los criterios 4 y 5 están alineados a los ejes de formación del Modelo Educativo UNACH "Introspección y Prospectiva" y responden principalmente a dos de los siguientes ejes:

1. Ambiente;
2. Autonomía y adaptabilidad;
3. Comunicación;
4. Desarrollo humano;
5. Ética y valores;
6. Emprendimiento;
7. Inter y multidisciplinariedad;
8. Innovación;
9. Inclusión e interculturalidad;
10. Investigación;
11. Impacto social;
12. Tecnologías.