Descripción del Proyecto 1 y Rúbrica de Evaluación

Programación – Santiago Echeverri Arteaga

Descripción del Proyecto

Este proyecto tiene como objetivo que los estudiantes apliquen los conocimientos adquiridos durante el curso de Programación en Python en un problema de física. Se espera que implementen un modelo o simulación numérica que resuelva dicho problema, haciendo uso de conceptos avanzados de programación orientada a objetos (POO), visualización y gestión de proyectos con GitHub.

Requisitos del Proyecto:

1. Implementación de POO: Utilizar herencia múltiple, abstracción y polimorfismo para modelar el problema de física. Se espera que las clases creadas tengan una estructura clara y que se aproveche la reutilización de código mediante la herencia.
2. Encapsulación: Los atributos importantes deben ser privados, accedidos y modificados mediante decoradores @property para asegurar un buen control de acceso.
3. Decoradores Adicionales: Definir e implementar decoradores adicionales como caching o timing para extender la funcionalidad de las funciones.
4. Método Numérico: Implementar un método numérico utilizando bucles complejos, recursividad o condiciones iterativas. No se permite el uso de librerías externas para el cálculo (más allá de math).
5. Visualización y Animación: Usar Matplotlib para realizar gráficas y animaciones que muestren la evolución de los resultados de la simulación.
6. Modularización y Generalización: El código debe estar modularizado y debe ser generalizable a otros problemas físicos sin modificar gran parte del código.
7. Gestión del Proyecto en GitHub: Se espera una adecuada gestión del proyecto mediante GitHub, con commits frecuentes y detallados, así como una clara documentación en el archivo README.md.
8. Calidad General del Código: El código debe ser claro, bien comentado y seguir las mejores prácticas de programación.

Los estudiantes deben entregar el código en un repositorio de GitHub, con un archivo README.md explicando cómo utilizar el proyecto. El código debe estar bien estructurado, modularizado y ser fácilmente entendible.

Rúbrica de Evaluación del Proyecto

Criterio	Descripción Detallada	Excelente (90-100%)	Bueno (75-89%)	Aceptable (60-74%)	Insuficiente (0-59%)	Puntaje Máximo
1. Implementación de POO (Herencia, Abstracción, Polimorfismo)	Aplicación correcta de la herencia múltiple, abstracción mediante clases abstractas, y polimorfismo para definir y extender comportamientos en diferentes clases.	Uso completo y preciso de todos los conceptos de POO. Todas las clases están estructuradas eficientemente.	Cumple con la mayoría de los conceptos de POO, con algunos errores menores.	Implementación parcial o con errores significativos.	Falla en la implementación de herencia múltiple, abstracción o polimorfismo.	20
2. Encapsulación y Atributos Privados	Implementar atributos privados con acceso controlado mediante el decorador @property.	Encapsulación correcta y uso óptimo de @property.	Encapsulación funcional con algunos errores menores.	Algunos atributos encapsulados, pero falta uso adecuado de @property.	No hay encapsulación o uso correcto de @property.	10
3. Decoradores Adicionales	Definir e implementar decoradores adicionales, como caching o timing, para extender la funcionalidad de las funciones.	Decoradores bien diseñados y extendiendo efectivamente la funcionalidad.	Decoradores presentes, pero su implementación no es óptima.	Decoradores implementados con errores o limitaciones.	No hay decoradores adicionales o están mal implementados.	10
4. Método Numérico (bucles complejos,	Implementar un método numérico utilizando	Método numérico eficiente, con bucles	Método funcional, pero puede mejorar en	Método numérico con errores	No se implementa un método	20

recursividad o condiciones iterativas)	bucles complejos, recursividad o condiciones iterativas. No usar librerías externas.	complejos o recursividad adecuada.	eficiencia o claridad.	significativos.	numérico adecuado.	
5. Visualización y Animación con Matplotlib	Uso de Matplotlib para visualizar y animar la evolución de los resultados.	Gráficas y animaciones de alta calidad y personalizables.	Gráficas y animaciones correctas pero podrían ser más detalladas.	Visualización básica y animación limitada.	No hay visualización o es confusa.	20
6. Modularización y Generalización del Código	Código modular, organizado en diferentes archivos y reutilizable en otros problemas.	Código completamente modular y generalizable.	Código modularizado pero con áreas de mejora.	Código parcialmente modular, pero poco flexible.	No está modularizado o no es generalizable.	10
7. Gestión del Proyecto en GitHub y Documentación	Uso adecuado de GitHub con commits frecuentes y documentación clara.	Repositorio bien gestionado y documentado.	Repositorio organizado pero documentación o commits podrían mejorar.	Gestión básica, con documentación y commits limitados.	Gestión insuficiente o sin documentación.	10
8. Calidad General del Código	Código bien estructurado, comentado y fácil de entender. Sigue las mejores prácticas.	Código limpio y bien comentado, con estructura clara.	Código bien estructurado pero con áreas de mejora en comentarios o claridad.	Código funcional pero desorganizado o poco claro.	Código mal estructurado o sin comentarios.	10