TRABAJO INTEGRADOR 2DO PARCIAL PROGRAMACIÓN Y LABORATORIO 2

El trabajo debe de poseer carátula y datos identificatorios del grupo:
Apellido y nombre de los integrantes, fecha de realización
Se solicita que no hayan dos temas iguales o similares, cada grupo deberá de tener un tema definido
El grupo deberá de encontrar un problema a resolver mediante la realización de un programa realizado en Python. El problema puede o debe salir de la solución a una necesidad detectada. Se valorará la creatividad y la originalidad del tema elegido, es decir, el problema a solucionar.

Se deberán entregar los documentos del TP con la siguiente información:
1- Enunciado del tema o problema,
2-análisis del problema y solución,Incluir perfil del usuario,programas similares existentes, etc
3- Prueba de escritorio - Ejecutar y mostrar resultados
4-Diagrama de resolución
5- código con comentarios
6- El programa deberá de resolver el enunciado dado y constará de usuario y contraseña
7- En el programa deben utilizarse los conceptos trabajados, entrada y salida de datos, condicionales, listas,bucles, etc. (Listar y nombrar todos los contenidos trabajados en el desarrollo del software)
8-Carpeta de proyecto- Donde irán documentando día a día los avances, los errores, los cambios realizados en el mismo.
9-Carpeta Resumen del proyecto(Explicando lo indispensable para realizar una venta del mismo) y código
10- Presentación en power point o similar que servirá como guía para la defensa del mismo.

Expectativas de Logro:

Aplicación de Conceptos: Los estudiantes deben demostrar la capacidad de aplicar todos los conceptos de Python aprendidos durante el curso en la creación de su proyecto.

Originalidad: Se espera que los proyectos sean originales y no simplemente copias de ejemplos existentes en línea o trabajos de otros estudiantes.

Estructuración del Problema: Los estudiantes deben ser capaces de definir claramente el problema que están resolviendo y comunicar de manera efectiva por qué es relevante.

Estructuración del Programa: Se espera que los proyectos estén bien organizados en términos de estructura de código y uso de funciones y módulos apropiados.

Funcionalidad: Los proyectos deben cumplir con los requisitos establecidos y realizar las tareas o resolver los problemas previstos de manera efectiva.

Documentación del Código: Los estudiantes deben proporcionar una documentación adecuada que explique cómo funciona su código, incluyendo comentarios y explicaciones claras.

Defensa del Proyecto: Los estudiantes deben ser capaces de defender su proyecto ante preguntas y desafíos, explicando sus decisiones de diseño y elecciones de implementación.

Objetivos:

- Aplicar de manera efectiva todos los conceptos de Python aprendidos durante el curso.
- Desarrollar una idea original y relevante que requiera programación en Python.
- Diseñar y estructurar un programa de manera organizada y eficiente.
- Implementar todas las funcionalidades requeridas para resolver el problema.
- Proporcionar documentación clara y legible del código.
- Presentar y defender el proyecto de manera efectiva ante el profesor y/o sus compañeros.

Rúbrica de evaluación (General)

Criterio	Insatisfactorio (0-2)	Aceptable (3-4)	Sobresaliente (5)
Aplicación de conceptos de Python	No demuestra dominio de los conceptos de Python.	Aplica algunos conceptos de Python, pero con errores.	Aplicación de manera efectiva todos los conceptos de Python aprendidos.
Originalidad	El proyecto carece de originalidad o es una copia evidente de trabajos existentes.	El proyecto es original, pero no necesariamente innovador.	El proyecto es altamente original e innovador.

Estructuración del Problema	No defina claramente el problema o su relevancia.	Defina el problema de manera adecuada, pero la relevancia no está clara.	Defina el problema claramente y explique su relevancia de manera efectiva.
Estructuración del Programa	El código está desorganizado y no se utiliza una estructura apropiada.	El código está organizado, pero podría ser más eficiente.	El código está bien estructurado y optimizado.
Funcionalidad	El proyecto no cumple con los requisitos o no funciona correctamente.	Cumple con la mayoría de los requisitos, pero con algunos errores.	Cumple con todos los requisitos y funciona de manera eficiente.
Documentación del Código	La documentación es insuficiente o inexistente.	La documentación es adecuada, pero no siempre es clara.	La documentación es clara y completa.
Defensa del Proyecto	No puede defender ni explicar efectivamente el proyecto.	Puede defender el proyecto, pero con dificultades para explicar decisiones.	Defiende y explica el proyecto de manera efectiva, respondiendo preguntas con claridad.

RUBRICA ESPECÍFICA

Criterio	Insatisfactorio (0-3)	Aceptable (4-7)	Sobresaliente (8-10)
Aplicación de conceptos de Python	No demuestra dominio de los conceptos de Python.	Aplica algunos conceptos de Python, pero con errores.	Aplicación de manera efectiva todos los conceptos de Python aprendidos.
- Aplicación de conceptos fundamentales de Python (variables, estructuras de datos, bucles, condicionales, funciones, etc.)			
- Uso de conceptos avanzados (clases, herencia, módulos, manejo de excepciones, etc.)			
Originalidad y Relevancia	El proyecto carece de originalidad o es una copia evidente de trabajos existentes.	El proyecto es original, pero no necesariamente innovador.	El proyecto es altamente original e innovador.

- Originalidad de la idea y su relevancia en el contexto actual.			
- Solución innovadora o creativa para el problema planteado.			
Estructuración del Problema	No defina claramente el problema o su relevancia.	Defina el problema de manera adecuada, pero la relevancia no está clara.	Defina el problema claramente y explique su relevancia de manera efectiva.
- Definición clara del problema a resolver y su relevancia			
- Identificación de requisitos y especificaciones del proyecto.			
Estructuración del Programa	El código está desorganizado y no se utiliza una estructura apropiada.	El código está organizado, pero podría ser más eficiente.	El código está bien estructurado y optimizado.

- Organización general del código y estilo de programación			
- Uso de funciones y módulos para modularizar el código			
- Eficiencia y optimización del código			
Funcionalidad y Cumplimiento de Requisitos	El proyecto no cumple con los requisitos o no funciona correctamente.	Cumple con la mayoría de los requisitos, pero con algunos errores.	Cumple con todos los requisitos y funciona de manera eficiente.
- Implementación correcta de todas las funcionalidades requeridas.			
- Manejo de casos límite y errores de manera adecuada			

Documentación y comentarios	La documentación es insuficiente o inexistente.	La documentación es adecuada, pero no siempre es clara.	La documentación es clara y completa.
- Documentación clara y detallada del código (docstrings, comentarios)			
- Comentarios explicativos que facilitan la comprensión del código.			
Interfaz de usuario (si aplica)	Diseño y usabilidad de la interfaz de usuario (si es relevante)	Diseño y usabilidad de la interfaz de usuario (si es relevante)	Diseño y usabilidad de la interfaz de usuario (si es relevante)
- Retroalimentación al usuario y manejo de eventos (si es relevante)			
Pruebas y Validación	Implementación de pruebas unitarias y/o funcionales.	Implementación de pruebas unitarias y/o funcionales.	Implementación de pruebas unitarias y/o funcionales.
- Resultados de pruebas y validación correctas.			

Defensa del Proyecto			
- Habilidad para explicar y justificar decisiones de diseño y elecciones de implementación.	No puede defender ni explicar efectivamente el proyecto.	Puede defender el proyecto, pero con dificultades para explicar decisiones.	Defiende y explica el proyecto de manera efectiva, respondiendo preguntas con claridad.
- Capacidad para responder preguntas y desafíos de manera efectiva			
Presentación y Calidad Visual (si aplica)			
- Uso de gráficos, diagramas o elementos visuales (si es relevante)	Calidad general de la presentación	Calidad general de la presentación	Calidad general de la presentación