**Challenge de Fin de Curso: "Python Data Journey"**

Introducción

En el viaje que es la programación en Python, hemos atravesado desde las praderas de las variables y los bucles hasta las montañas de las estructuras de datos complejas y la interacción con archivos. Ahora, al final de este curso de Programación I en Python, nos enfrentamos a un reto integral que pone a prueba todo lo aprendido y un poco más. ¡Es hora de aplicar tus habilidades en un proyecto que simule un escenario del mundo real!

Objetivos

* Consolidar los conocimientos adquiridos durante el curso de Python en un proyecto aplicado.
* Desarrollar un programa que procese datos de archivos y presente resultados analíticos.
* Familiarizarse con el uso de control de versiones utilizando GitHub.
* Demostrar buenas prácticas de codificación y documentación de código.

Metodología

1. **Planificación:** Antes de escribir una línea de código, planifica tu enfoque. Define qué estructuras de datos utilizarás, cómo manejarás la entrada/salida de archivos y qué lógica aplicarás para el procesamiento de datos.
2. **Desarrollo Iterativo:** Comienza con las funcionalidades básicas y avanza incrementando la complejidad. Después de cada función o módulo completado, realiza pruebas para garantizar la funcionalidad.
3. **Refactorización:** Una vez que tengas un prototipo funcional, revisa tu código y mejóralo. Elimina redundancias y asegúrate de que tu código es limpio y legible.
4. **Control de Versiones:** Utiliza GitHub para mantener un registro de tus cambios y versiones. Asegúrate de hacer commits claros y concisos que reflejen cada etapa importante del desarrollo de tu software.

Uso de GitHub

* Inicializa un repositorio en GitHub al comienzo de tu proyecto.
* Realiza commits pequeños y frecuentes con mensajes descriptivos.
* Mantén una estructura de directorios limpia y profesional en tu repositorio.
* Incluye un archivo **README.md** que proporcione una descripción clara del proyecto, cómo ejecutarlo y ejemplos de uso.

Puntos a Evaluar

* **Funcionalidad (40%):** El script debe funcionar según lo requerido, procesando y analizando datos de un archivo de texto de manera eficiente y efectiva.
* **Calidad del Código (20%):** El código debe ser limpio, bien organizado y seguir las convenciones de Python (PEP 8).
* **Manejo de Excepciones (10%):** El script debe manejar adecuadamente los errores y excepciones, proporcionando mensajes de error informativos.
* **Documentación (10%):** El código debe estar bien comentado y el repositorio de GitHub debe tener un **README.md** detallado.
* **Uso de Git (10%):** Deben realizarse commits regulares y descriptivos a lo largo del desarrollo del proyecto.
* **Presentación de Resultados (10%):** El script debe presentar los resultados de manera clara y legible, preferiblemente en un archivo de salida separado.

Descripción del Challenge

**El Gran Analizador de Texto Python (GATPy):**

Crearás un script de Python denominado **GATPy**. Este script leerá un archivo de texto provisto por el usuario (por ejemplo, un libro de dominio público de Project Gutenberg). Luego, calculará y presentará lo siguiente:

* La frecuencia de cada palabra en el texto.
* Las 5 palabras más comunes y su conteo.
* La cantidad de palabras únicas en el documento.
* Opcional: Una nube de palabras generada a partir del texto.

Además, el script generará un archivo de salida **.txt** con los resultados del análisis y una visualización simple si optas por la funcionalidad opcional.

Entrega

El plazo de entrega será de dos semanas a partir de la fecha de lanzamiento del desafío. Todo el trabajo debe ser subido a tu repositorio de GitHub antes de la fecha y hora límite. Asegúrate de seguir las mejores prácticas de codificación y mantener tu repositorio público para que pueda ser revisado por los instructores.

¡Buena suerte y que comience la codificación!

**Tópicos Adicionales para el GATPy**

1. **Análisis de Sentimiento:**
   * Implementar una función que determine el sentimiento general del texto (positivo, negativo, neutral) utilizando un paquete de Python como **TextBlob** o **nltk**.
2. **Resumen de Texto:**
   * Crear un resumen automático del texto proporcionado, destacando los puntos clave o las frases más frecuentes.
3. **Indexación de Capítulos o Secciones:**
   * Si el texto tiene capítulos o secciones claras, escribir una función que automáticamente los indexe y proporcione un conteo de palabra por capítulo o sección.
4. **Análisis de Complejidad de Lectura:**
   * Añadir una funcionalidad que calcule el índice de legibilidad de Flesch-Kincaid o cualquier otra métrica de complejidad de lectura.
5. **Estadísticas de Gramática:**
   * Contar y categorizar las partes del habla (sustantivos, verbos, adjetivos, etc.) utilizando bibliotecas como **spaCy** o **nltk**.
6. **Reconocimiento de Entidades Nombradas:**
   * Utilizar una biblioteca de procesamiento de lenguaje natural para reconocer y contar entidades nombradas (personas, lugares, organizaciones).
7. **Generación de un Índice de Palabras:**
   * Generar un índice alfabético de todas las palabras utilizadas en el texto con el número de página o la posición del texto donde aparecen.
8. **Comparación con Otros Textos:**
   * Permitir al usuario introducir múltiples textos y proporcionar una comparación estadística entre ellos (uso de palabras comunes, estilo de escritura, etc.).
9. **Visualizaciones Avanzadas:**
   * Crear gráficos interactivos para las estadísticas recolectadas, utilizando bibliotecas como **matplotlib**, **seaborn**, o **plotly**.
10. **Interfaz de Usuario:**
    * Desarrollar una interfaz gráfica sencilla para permitir a los usuarios seleccionar archivos y visualizar los análisis directamente, usando por ejemplo **tkinter** o **PyQt**.
11. **Empaquetado y Distribución:**
    * Preparar el script para que pueda ser instalado y utilizado como un paquete de Python independiente, incluyendo todas las dependencias necesarias.
12. **Integración Continua:**
    * Configurar un pipeline de integración continua con GitHub Actions para automatizar las pruebas del script.

**Entregables**

Además del código fuente, los siguientes entregables serán parte del challenge:

* **Informe de Análisis:** Un documento que resume los hallazgos y estadísticas del texto analizado, incluyendo gráficos y visualizaciones.
* **Guía de Usuario:** Un manual o documento de ayuda que guíe a los usuarios en la utilización del GATPy, con ejemplos de uso.
* **Reflexión Personal:** Una breve reflexión escrita sobre el proceso de aprendizaje, los desafíos enfrentados y cómo se resolvieron.
* **Pruebas Unitarias:** Un conjunto de pruebas unitarias que aseguren la correcta funcionalidad de cada componente del script.
* **Vídeo Demostrativo:** Una grabación de pantalla mostrando cómo funciona el GATPy, explicando las características implementadas.

Estos tópicos adicionales pretenden que los estudiantes apliquen una variedad más amplia de las habilidades aprendidas durante el curso, al tiempo que ofrecen un enfoque holístico e integrado para el desarrollo de un proyecto de software.

**Challenge “Python Enterprise Simulator"**

Introducción

Al culminar el curso de Programación I con Python en el nivel de maestría, te enfrentas a un desafío que encapsula la esencia de un entorno empresarial complejo. Este reto no solo pondrá a prueba tus habilidades técnicas, sino también tu capacidad para gestionar un proyecto de programación escalable y modular, que podría integrarse en sistemas de mayor envergadura.

Objetivos

* Desarrollar una aplicación Python que simule un entorno empresarial con varios módulos, como gestión de inventarios, procesamiento de transacciones y análisis de datos.
* Demostrar un diseño de software eficiente que permita la fácil expansión y mantenimiento del código.
* Manejar de forma efectiva un repositorio de GitHub para control de versiones y colaboración.
* Aplicar las mejores prácticas en programación, pruebas, y documentación.

Metodología

1. **Diseño Modular:** Tu aplicación debe estar diseñada en módulos independientes que interactúan entre sí. Cada módulo debería ser responsable de una parte del simulador empresarial.
2. **Desarrollo Ágil:** Implementa tu proyecto utilizando sprints de una semana, cada uno con sus propios objetivos y tareas. Al final de cada sprint, evalúa tu progreso y reajusta los planes según sea necesario.
3. **Revisión de Pares:** Realiza revisiones de código con tus compañeros para asegurarte de que tu código sigue las mejores prácticas y es claro para otros desarrolladores.
4. **Testing:** Desarrolla tests para cada módulo utilizando frameworks como **unittest** o **pytest**. Asegúrate de que tu código pasa todas las pruebas antes de fusionarlo con la rama principal.

Uso de GitHub

* Configura un repositorio en GitHub con una estructura de ramas adecuada para gestionar características, desarrollos y lanzamientos.
* Usa issues y pull requests para gestionar tareas y la colaboración.
* Integra GitHub Actions para automatizar tus pruebas y asegurar la calidad del código.

Puntos a Evaluar

* **Arquitectura de Software (25%):** Diseño modular y uso de patrones de diseño.
* **Calidad del Código (20%):** Legibilidad, uso de convenciones y documentación interna del código.
* **Control de Versiones (15%):** Uso efectivo de GitHub, incluyendo branches, commits, issues, y pull requests.
* **Funcionalidad (15%):** El simulador debe funcionar como se espera y manejar correctamente las simulaciones empresariales.
* **Pruebas (10%):** Cobertura de pruebas completa y pasaje de todas las pruebas automatizadas.
* **Documentación (10%):** README detallado, documentación de la API y comentarios en el código.
* **Presentación (5%):** Entrega final con una presentación clara y profesional del proyecto.

Descripción del Challenge

**Python Enterprise Simulator (PyESim):**

Tu simulador empresarial, denominado **PyESim**, constará de los siguientes módulos:

1. **Gestión de Inventario:** Control de stock, seguimiento de productos y alertas de bajo inventario.
2. **Procesamiento de Transacciones:** Simulación de ventas y compras, con registro de transacciones.
3. **Análisis de Datos:** Generación de informes sobre ventas, predicciones de demanda y visualizaciones de datos de inventario.
4. **Interfaz de Usuario:** Una interfaz de línea de comandos o gráfica que permita interactuar con el simulador.
5. **Integración y Automatización:** Scripts para integrar los módulos y automatizar tareas comunes.
6. **Seguridad:** Implementación de medidas básicas de seguridad para proteger los datos simulados.

Entrega

El proyecto deberá ser entregado en un periodo de 8 semanas. La entrega final incluirá:

* **Código Fuente:** Todo el código fuente del proyecto, alojado en el repositorio de GitHub.
* **Documentación Técnica:** Instrucciones detalladas sobre cómo instalar y ejecutar el simulador, incluyendo requisitos y dependencias.
* **Informe de Proyecto:** Un informe que detalle la arquitectura del software, las decisiones de diseño importantes, y una reflexión sobre los desafíos enfrentados.
* **Presentación Final:** Una presentación del simulador, demostrando sus capacidades y cómo