**Documentación Detallada de "entrevista.py"**

**1. Introducción**

El script "entrevista.py" es una aplicación modular en Python diseñada para automatizar el procesamiento de datos de entregas, el envío de notificaciones por correo electrónico a clientes y la generación de reportes resumen. Este documento explica en detalle la estructura, funcionalidades y el flujo de ejecución del programa.

**2. Requisitos y Dependencias**

* **Librerías y módulos estándar:**
  + pandas: Para manipulación y análisis de datos, especialmente para leer y escribir archivos Excel.
  + os: Para operaciones relacionadas con el sistema de archivos.
  + re: Para procesamiento y transformación de cadenas mediante expresiones regulares.
  + smtplib y email.message: Para el envío de correos electrónicos usando el protocolo SMTP.
  + datetime: Para el manejo y formateo de fechas.
* **Archivo de entrada:**
  + Se espera un archivo Excel llamado entregas\_pendientes.xlsx que contenga la información de las entregas.
* **Configuración de correo:**
  + Se requiere una cuenta de Gmail válida. El script utiliza el servidor SMTP de Gmail para enviar notificaciones a los clientes.

**3. Estructura General del Código**

El código se organiza en tres clases principales y una función main que orquesta el flujo completo de procesamiento.

**3.1. Clase ProcesadorDatosEntregas**

**Propósito:**  
Esta clase se encarga de leer y transformar los datos de entregas desde un archivo Excel, así como de filtrar la información no deseada (por ejemplo, filas con estado "Devuelto").

**Componentes:**

* **Atributo de Clase:**
  + MESES: Un diccionario que mapea nombres de meses en español a su representación numérica en dos dígitos (por ejemplo, "Enero" → "01").
* **Métodos:**
  + **\_\_init\_\_(self, archivo\_entrada)**  
    Inicializa la instancia con el archivo de entrada y prepara atributos para almacenar datos originales y procesados.
  + **convertir\_fecha(self, fecha)**  
    Convierte una fecha que puede estar en varios formatos (cadena o datetime) al formato estándar YYYY-MM-DD.
    - Utiliza expresiones regulares para dividir la cadena y reemplaza el nombre del mes por su correspondiente numérico.
    - Prueba distintos formatos ("%Y/%d/%m", "%Y/%m/%d") hasta lograr una conversión exitosa.
  + **limpiar\_valor(self, valor)**  
    Limpia y convierte un valor numérico representado como cadena (posiblemente con puntos y comas) a un número flotante, redondeado a dos decimales.
  + **procesar\_datos(self)**  
    Lee el archivo Excel, aplica la conversión de fechas, limpia los datos (por ejemplo, eliminando espacios innecesarios en el campo "Cliente") y formatea los valores numéricos.
    - Filtra las filas en las que el campo "Estado\_Entrega" no es "Devuelto", generando un DataFrame de datos procesados.
  + **guardar\_archivo\_procesado(self, archivo\_salida)**  
    Guarda el DataFrame resultante en un archivo Excel, mostrando un mensaje en la consola que indica la ruta de salida.

**3.2. Clase EnviarCorreo**

**Propósito:**  
Encargada de gestionar el envío de correos electrónicos a los clientes según el estado de la entrega de sus pedidos.

**Componentes:**

* **Métodos:**
  + **\_\_init\_\_(self, sender\_email, password=None)**  
    Inicializa la clase con las credenciales del remitente.
    - Si no se suministra la contraseña, se solicita interactivamente mediante input().
  + **enviar\_correo(self, destinatario, asunto, cuerpo)**  
    Construye y envía un correo electrónico utilizando EmailMessage y el servidor SMTP de Gmail.
    - Se establece una conexión segura (TLS) y se maneja el inicio de sesión.
    - Se envía el mensaje y se imprime un mensaje de éxito o error según el resultado.
  + **enviar\_correos\_desde\_dataframe(self, df)**  
    Recorre cada fila de un DataFrame y envía correos a los clientes basándose en el estado de la entrega:
    - Para entregas **"Pendiente"**: Se envía un mensaje notificando que el pedido está en camino.
    - Para entregas **"Entregado"**: Se notifica que el pedido ha sido entregado exitosamente.
    - Para otros estados (por ejemplo, "Devuelto"): Se imprime un mensaje indicando que no se envió correo.

**3.3. Clase GeneradorReporte**

**Propósito:**  
Genera un reporte resumen basado en los datos procesados de entregas, ofreciendo información consolidada para la toma de decisiones.

**Componentes:**

* **Métodos:**
  + **\_\_init\_\_(self, datos)**  
    Inicializa la clase con el DataFrame de datos procesados.
  + **generar\_reporte(self)**  
    Crea un resumen con los siguientes elementos:
    - **Total de entregas procesadas:** Cuenta total de registros.
    - **Ciudades con más entregas pendientes:** Se obtienen las ciudades con pedidos en estado "Pendiente" y se cuentan.
    - **Monto total de entregas realizadas:** Suma de los valores de pedidos con estado "Entregado".

El reporte se genera como una cadena de texto con formato.

* + **guardar\_reporte(self, archivo\_reporte)**  
    Escribe el reporte generado en un archivo de texto y lo muestra en la consola.

**3.4. Función main**

**Propósito:**  
Orquestar la ejecución de todas las operaciones del script, desde el procesamiento de datos hasta el envío de correos y la generación de reportes.

**Flujo de Ejecución:**

1. **Procesamiento de Datos:**
   * Se instancia ProcesadorDatosEntregas utilizando el archivo de entrada entregas\_pendientes.xlsx.
   * Se llama a procesar\_datos() para realizar las transformaciones y filtrado.
   * Se guarda el DataFrame procesado en entregas\_procesadas.xlsx mediante guardar\_archivo\_procesado().
2. **Envío de Correos:**
   * Se configura el correo del remitente y la contraseña (la contraseña puede ser pasada directamente o ingresada manualmente).
   * Se instancia la clase EnviarCorreo y se llama a enviar\_correos\_desde\_dataframe(), la cual recorre el DataFrame filtrado y envía notificaciones según el estado de cada entrega.
3. **Generación del Reporte:**
   * Se crea una instancia de GeneradorReporte usando el DataFrame de datos procesados.
   * Se genera un reporte resumen y se imprime en la consola.
   * El reporte se guarda en un archivo de texto llamado reporte.txt.
4. **Ejecución del Script:**
   * La función main() se ejecuta al iniciar el script, coordinando todas las operaciones anteriores.

**4. Uso y Ejecución**

Para utilizar el script, siga estos pasos:

1. **Preparación del entorno:**
   * Instale las dependencias necesarias (por ejemplo, mediante pip install pandas).
   * Asegúrese de tener un archivo Excel entregas\_pendientes.xlsx con la estructura de datos esperada (incluyendo campos como Fecha\_Pedido, Cliente, Valor, Estado\_Entrega, Ciudad, Correo\_Cliente, ID\_Entrega).
2. **Configuración del correo:**
   * Modifique la variable sender\_email con su dirección de Gmail.
   * Proporcione la contraseña correspondiente (puede ser ingresada manualmente o directamente en el código, aunque esta práctica no es recomendable en producción).
3. **Ejecución:**
   * Ejecute el script desde la línea de comandos:

nginx

Copiar

python entrevista.py

* + El script procesará los datos, enviará los correos según corresponda y generará un reporte que se imprimirá en la consola y se guardará en reporte.txt.

**5. Consideraciones y Posibles Mejoras**

* **Seguridad:**
  + Evitar almacenar contraseñas en texto plano. Se recomienda utilizar variables de entorno o métodos seguros de gestión de credenciales.
* **Manejo de errores:**
  + Se implementa manejo básico de excepciones en la conversión de valores y el envío de correos. Sin embargo, se podría mejorar con registros (logging) para mayor trazabilidad.
* **Validación de datos:**
  + Verificar que el archivo de entrada cumple con la estructura esperada y que los campos tienen el formato correcto.
* **Extensibilidad:**
  + Agregar soporte para otros proveedores de correo o incluir configuraciones personalizadas mediante archivos de configuración.
  + Implementar interfaces gráficas o web para facilitar la interacción con el usuario.

**6. Conclusión**

El archivo "entrevista.py" es un ejemplo de aplicación modular en Python que integra procesamiento de datos, automatización del envío de notificaciones y generación de reportes. Gracias a su estructura basada en clases, el código es organizado y escalable, lo que facilita su mantenimiento y futuras ampliaciones.

Esta guía pretende servir como documento de referencia para entender cada componente y el flujo general del programa, permitiendo a desarrolladores y administradores adaptar y mejorar la aplicación según sus necesidades.