**💻 Presentación del Programa: "Control de Gastos con Python"**

**🔹 Introducción**

El siguiente programa fue desarrollado en **Python** con el objetivo de permitir al usuario **gestionar sus gastos personales** de forma interactiva desde la consola.  
A través de un menú dinámico, el usuario puede **agregar, visualizar, actualizar y eliminar gastos**, además de **analizar sus datos mediante gráficos** generados con la biblioteca **Matplotlib**.

El sistema guarda la información en un archivo **JSON**, lo que permite mantener los registros entre ejecuciones.

**🔹 Objetivos del Proyecto**

* Registrar gastos clasificados por **categorías predefinidas**.
* Permitir la **edición y eliminación** de registros.
* **Guardar y cargar** los datos de manera persistente.
* **Visualizar estadísticas** de los gastos mediante gráficos:
  + Histograma
  + Gráfico de barras
  + Boxplot
  + Gráfico circular (Pie Chart)

**🔹 Bibliotecas Utilizadas**

| **Biblioteca** | **Función principal** |
| --- | --- |
| **os** | Limpieza de pantalla según el sistema operativo. |
| **json** | Guardar y recuperar los datos en formato JSON. |
| **matplotlib.pyplot** | Creación de gráficos estadísticos. |
| **numpy** | Soporte para operaciones numéricas eficientes. |
| **random** | Generación de datos de prueba aleatorios. |
| **datetime** | Manejo de fechas para registrar gastos. |

Estas bibliotecas hacen posible una aplicación interactiva, persistente y con análisis visual.

**🔹 Estructura del Código**

El código está organizado de forma modular mediante **funciones** que cumplen tareas específicas:

**1. 📂 Funciones de gestión de archivos**

* cargar\_gastos() → Carga los datos desde un archivo JSON.
* guardar\_gastos() → Guarda los registros actualizados.

**2. 🧮 Funciones principales del menú**

* agregar\_gasto() → Permite al usuario registrar un nuevo gasto.
* ver\_gastos() → Muestra todos los gastos registrados.
* actualizar\_gasto() → Edita un gasto existente.
* eliminar\_gasto() → Elimina un registro con confirmación.

**3. 🧰 Funciones auxiliares**

* limpiar\_pantalla() → Limpia la consola.
* esperar\_enter() → Pausa la ejecución hasta que el usuario presione Enter.
* generar\_gastos\_falsos() → Crea datos de ejemplo para probar el sistema.

**4. 📊 Funciones de análisis y visualización**

* ver\_histograma() → Cantidad de gastos por categoría.
* ver\_barras() → Monto total por categoría.
* ver\_boxplot() → Distribución de montos.
* ver\_pie\_chart() → Proporción de gastos por categoría.
* mostrar\_todos\_los\_graficos() → Visualiza los cuatro gráficos juntos.

**🔹 Funcionalidad Principal**

El programa inicia mostrando un **menú interactivo**, desde donde el usuario puede seleccionar acciones numéricas.  
Cada opción invoca una función específica, y al finalizar una acción, el sistema **pausa y espera la interacción del usuario**.

Además, el menú incluye una sección especial de **análisis de datos**, donde se pueden generar gráficos individuales o ver todos simultáneamente.

**🔹 Características Destacadas**

✅ **Interfaz simple y clara:** todas las acciones se realizan desde la consola.  
✅ **Persistencia de datos:** los registros se guardan en formato JSON.  
✅ **Visualización estadística:** los gráficos permiten analizar gastos de forma intuitiva.  
✅ **Datos de prueba automáticos:** útil para probar la aplicación sin ingresar información manualmente.

**🔹 Ejemplo de uso**

1️⃣ El usuario ejecuta el programa y elige “Agregar gasto”.  
2️⃣ Selecciona la categoría, introduce monto, fecha y descripción.  
3️⃣ Puede luego ver todos los gastos registrados.  
4️⃣ Desde la sección de análisis, observa gráficos que muestran sus hábitos de consumo.

**🔹 Conclusión**

Este programa demuestra cómo **Python puede integrarse con herramientas de análisis de datos** para resolver problemas cotidianos.  
Su diseño modular, el uso de archivos JSON y la implementación de visualizaciones lo convierten en una base sólida para futuros proyectos más complejos, como una aplicación con interfaz gráfica o conexión a bases de datos.