1. Datos Generales

Carrera:	**Tecnología Superior en Desarrollo de Software**
Período académico:	**Marzo 2025 – Julio 2025**
Asignatura:	**Inteligencia Artificial**
Unidad Nº:	**1**
Tema:	**Guía práctica Unidad 1**
Ciclo-Paralelo:	**N6A**
Estudiante:	**José Marcelo Morocho Chimbo**
Docente:	**Ing. Marcelo Monteros Guerrero**

2. Contenido

2.1 Introducción

El presente trabajo práctico tiene como finalidad aplicar conceptos fundamentales de programación en Python, enfocados en el manejo de archivos, análisis de datos con Pandas y visualización de información con Matplotlib. A través del desarrollo de distintos módulos, se busca fortalecer habilidades en automatización de tareas, limpieza y procesamiento de datos reales, y representación gráfica de resultados. Para ello, se emplearon dos conjuntos de datos: uno referente a la recaudación municipal por provincias y otro relacionado con el desempeño de selecciones en Copas Mundiales de Fútbol.

2.2 Objetivo

El objetivo principal del trabajo es desarrollar un sistema funcional en Python que permita:

1. Realizar respaldos automáticos y manuales de archivos en un directorio especificado.
2. Recuperar archivos eliminados desde copias de seguridad.
3. Analizar un conjunto de datos con información demográfica y financiera, aplicando técnicas de depuración y agrupación mediante la biblioteca Pandas.
4. Representar gráficamente los promedios de recaudación por provincia y los resultados históricos de los equipos participantes en los mundiales.

5. Documentar adecuadamente cada parte del proceso con código funcional y explicaciones claras.

2.3 Materiales, herramientas, equipos y software

- Computadora
- Python
- Internet

2.4 Desarrollo

EJERCICIO1
Paso 1: Importación de librerías

```
main.py > ...
1   import os
2   import shutil
3   import time
4   import threading
5   import filecmp
6
```

Estas librerías permiten:

- os: manipular carpetas y rutas
- shutil: copiar archivos
- time: usar temporizadores
- threading: ejecutar tareas en segundo plano (como el backup automático)
- filecmp: comparar archivos para validar si son idénticos

Paso 2: Definición de carpetas por defecto

```
BACKUP_FOLDER = "backups/"
ORIGINAL_FOLDER = "original_files/"
```

Aquí se definen las rutas por defecto para los archivos originales y la carpeta de respaldo.

Paso 3: Función escoger_directorio()

```
def escoger_directorio():
    directorio = input("Introduce la ruta del directorio a respaldar (por defecto 'original_files/'): ").strip()
    return directorio if directorio else ORIGINAL_FOLDER
```

Permite al usuario escribir una ruta personalizada o usar el valor por defecto original_files/.

Paso 4: Función hacer_backup()

```
def hacer_backup(directorio):
    if not os.path.exists(BACKUP_FOLDER):
        os.makedirs(BACKUP_FOLDER)
    for archivo in os.listdir(directorio):
        origen = os.path.join(directorio, archivo)
        destino = os.path.join(BACKUP_FOLDER, archivo)
        if os.path.isfile(origen):
            shutil.copy2(origen, destino)
            print(f"[✓] Backup creado para: {archivo}")
```

Esta función:

- Crea la carpeta de backup si no existe.
- Copia todos los archivos del directorio de origen a la carpeta backup.

```
Selecciona una opción: 2
[✓] Backup creado para: Ejercicios de Aplicaciones Seguras.zip
```

Paso 5: Backup automático

```
def backup_periodico(directorio):
    while True:
        hacer_backup(directorio)
        print("Esperando 2 minutos para el siguiente backup...")
        time.sleep(120)
```

Este proceso corre en segundo plano cada 2 minutos y hace backup de todo automáticamente.

Paso 6: Restaurar archivo

```python
def restaurar_archivo(nombre_archivo):
    backup_path = os.path.join(BACKUP_FOLDER, nombre_archivo)
    original_path = os.path.join(ORIGINAL_FOLDER, nombre_archivo)
    if os.path.exists(backup_path):
        shutil.copy2(backup_path, original_path)
        print(f"[✓] Archivo restaurado: {nombre_archivo}")
        if filecmp.cmp(backup_path, original_path, shallow=False):
            print("[✓] Validación completa: el archivo restaurado es idéntico al original.")
        else:
            print("[✗] El archivo restaurado es diferente al original.")
    else:
        print("[!] No se encontró un respaldo de ese archivo.")
```

Esta función:

- Copia un archivo desde la carpeta de respaldo a original_files/
- Verifica que el archivo restaurado sea igual al respaldado

Paso 7: Menú principal

Este bloque gestiona el flujo del programa con un menú que permite:

```
--- MENÚ DE BACKUP ---
1. Escoger directorio a respaldar
2. Hacer backup manual
3. Activar backup automático (cada 2 min)
4. Restaurar archivo
5. Salir
Selecciona una opción:
```

EJERCICIOS 2

1.Importación de librerías

```python
import pandas as pd
import matplotlib.pyplot as plt

```

Estas dos librerías son esenciales para:

- pandas (pd): manipulación y análisis de datos en tablas (como Excel o CSV).
- matplotlib.pyplot (plt): para generar gráficos y visualizaciones.

2.Función opcion_1(): Análisis de recaudación por provincia

2.1. Lectura del archivo CSV

```
def opcion_1():

    try:
        df = pd.read_csv("poblacionMunicipios.csv", sep=";", encoding="utf-8", on_bad_lines='skip')
    except Exception as e:
        print("Error al leer el archivo:", e)
        return
```

- Lee el archivo CSV con separador ;.
- Si hay líneas con errores de formato, se saltan con on_bad_lines='skip'.
- En caso de error, se imprime un mensaje y se sale de la función.

2.2. Limpieza de datos

```
if 'Recaudacion' in df.columns and 'Provincia' in df.columns:
    df['Recaudacion'] = pd.to_numeric(df['Recaudacion'], errors='coerce')
    minimo = df["Recaudacion"].min()
    df["Recaudacion"].fillna(minimo, inplace=True)
```

- Convierte la columna Recaudacion en numérica.
- Reemplaza los valores faltantes con el mínimo de la columna.

2.3. Agrupación y promedio

```
    promedios = df.groupby("Provincia")["Recaudacion"].mean().reset_index()
    promedios = promedios.sort_values(by="Recaudacion", ascending=False)
```

- Agrupa los datos por provincia y calcula el promedio de recaudación.
- Luego ordena de mayor a menor.

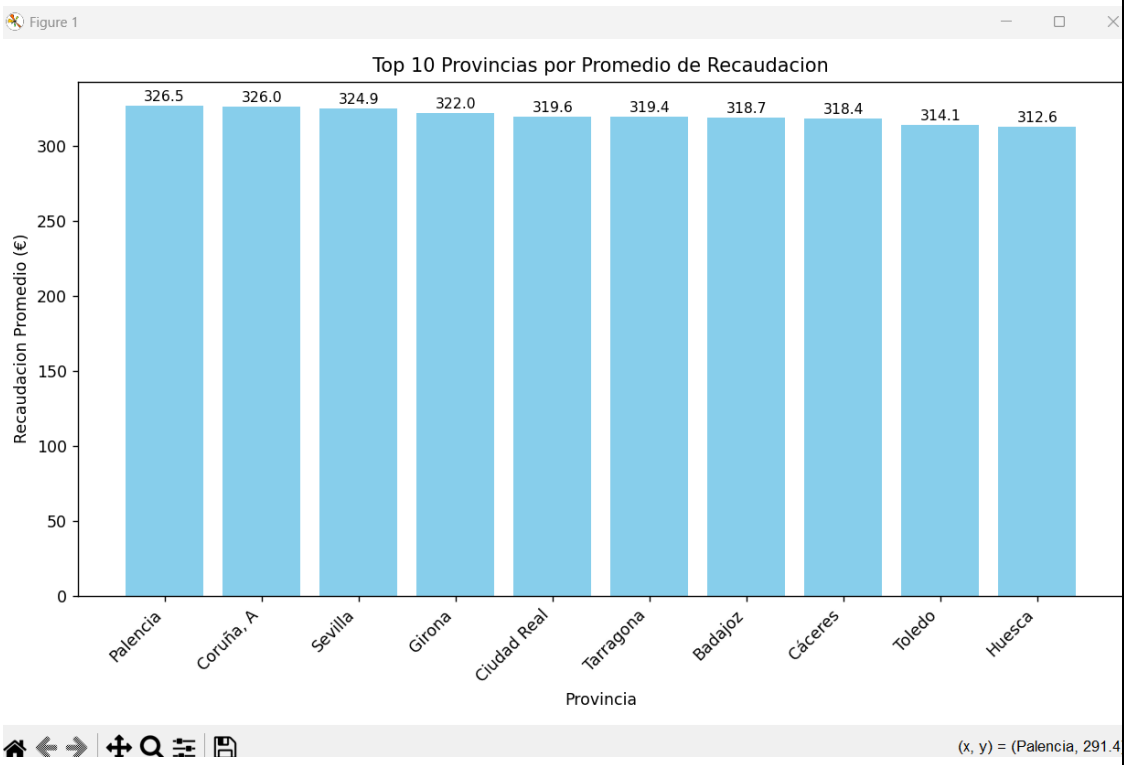
2.4. Exportación a CSV

```
    promedios.to_csv("promedio_recaudacion_provincia.csv", index=False)
```

- Guarda los promedios en un nuevo archivo CSV.

2.5. Gráfico de barras (Top 10 provincias)

- Se crea una figura de tamaño 10x6.
- Se grafica el top 10 de provincias con mayor recaudación promedio.
- Se añaden etiquetas numéricas sobre las barras.



.

EJERCICIO 3

3.Función opcion_2: Análisis de partidos de mundiales

3.1. Cargar archivo mundiales.csv

```
df = pd.read_csv("mundiales.csv")
```

- Lee el archivo con datos de partidos jugados por país.

3.2. Limpieza de caracteres extraños

```
df['pais'] = df['pais'].str.encode('latin1').str.decode('utf-8', errors='ignore')
max_partidos = df['partidos_totales'].max()
```

- Soluciona posibles errores de codificación en los nombres de países.

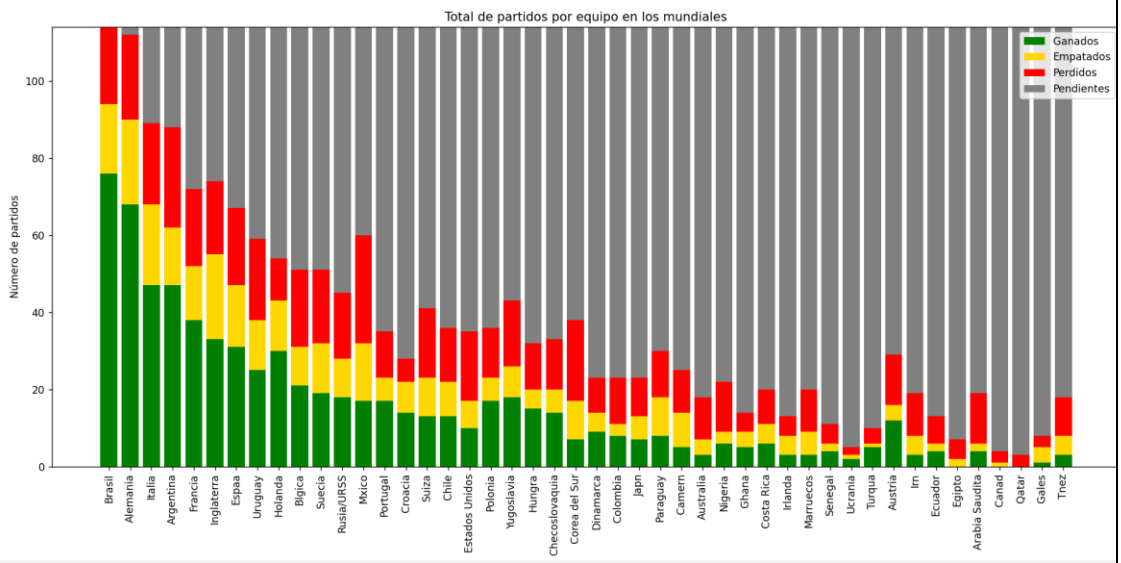
3.3. Cálculo de partidos pendientes

```
df['partidos_pendientes'] = max_partidos - df['partidos_totales']
```

- Calcula cuántos partidos le faltan jugar a cada país tomando como referencia el que más ha jugado.

3.4. Gráfico de barras apiladas

- Crea un gráfico de barras con diferentes colores para ganados, empatados, perdidos y pendientes.
- Las barras están apiladas para cada país.

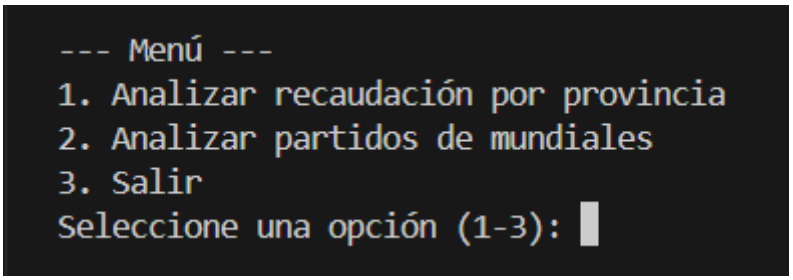


4.Menú principal

- Muestra un menú con 3 opciones: ejecutar el análisis 1, el análisis 2 o salir.
- Según la opción elegida, llama a la función correspondiente.

```python
while True:
    print("\n--- Menú ---")
    print("1. Analizar recaudación por provincia")
    print("2. Analizar partidos de mundiales")
    print("3. Salir")
    opcion = input("Seleccione una opción (1-3): ")

    if opcion == '1':
        opcion_1()
    elif opcion == '2':
        opcion_2()
    elif opcion == '3':
        print("Saliendo del programa...")
        break
    else:
        print("Opción no válida. Intente de nuevo.")
```



2.5 Conclusiones

La realización de este trabajo práctico permitió integrar distintas habilidades clave en el desarrollo de soluciones en Python. Se logró automatizar el respaldo y recuperación de archivos, implementar correctamente la lectura y procesamiento de datos con Pandas, y visualizar la información de manera efectiva con Matplotlib. Además, se evidenció la importancia de la limpieza y verificación de los datos antes de realizar cualquier análisis. Este proyecto refuerza la comprensión del flujo de trabajo en ciencia de datos y demuestra cómo Python puede ser utilizado como una herramienta poderosa para resolver problemas reales de manera eficiente y organizada.