

# Proyecto de clase de Python aplicado.

Fundamentos de Programación

## LaLiga Insights Pro

### 1. Información General

- **Nombre estudiantes:**

Andrés Felipe Fonseca Ardila – Andrés Felipe Mambuscay Reyes – Bayron Franco

- **Curso / Grupo:**

Fundamentos De Programación C 20252

- **Fecha de entrega:**

12/11/25

- **Profesor:**

Pablo Carreño

### 2. Título del Proyecto

LaLiga Insights Pro

### 3. Descripción del Proyecto

El proyecto LaLiga Insights Pro tiene como propósito analizar y visualizar datos reales de la liga española de fútbol (LaLiga) utilizando el lenguaje de programación Python. Su objetivo principal es facilitar la comprensión del rendimiento de los equipos y jugadores a lo largo de la temporada, a través de estadísticas como goles, victorias, empates y derrotas.

El proyecto está dirigido a aficionados al fútbol, analistas deportivos y estudiantes interesados en el análisis de datos, ya que permite obtener información actualizada directamente desde fuentes en línea mediante técnicas de web scraping. De esta forma, se convierte en una herramienta práctica para estudiar el desempeño de los clubes y futbolistas de una manera automatizada y ordenada.

Como resultado esperado, se busca generar tablas y visualizaciones claras que muestren el rendimiento de los equipos y jugadores, identificando los más destacados de la temporada. Además, el proyecto deja una base lista para incorporar en el futuro análisis más avanzados o modelos predictivos que permitan comparar temporadas o predecir resultados.

### 4. Objetivos

#### Objetivo general:

Analizar el rendimiento de los equipos y jugadores de **LaLiga** mediante el uso de **Python**, aplicando técnicas de recolección y análisis de datos que permitan obtener información relevante sobre goles, victorias, derrotas y desempeño general.

## Objetivos específicos:

- **Recolectar datos** de los equipos y jugadores de LaLiga en un formato adecuado para su análisis utilizando Python y sus librerías.
- **Comparar el rendimiento** de los equipos a partir de métricas clave como goles, victorias y derrotas.
- **Generar visualizaciones** que faciliten la interpretación de los datos y los hallazgos del análisis.

## 5. Requisitos

El proyecto fue desarrollado en el lenguaje de programación **Python**, aprovechando sus librerías más comunes para el análisis y procesamiento de datos.

Entre las principales herramientas utilizadas se encuentran:

- **Requests:** para realizar la conexión con las páginas web y obtener su contenido HTML.
- **BeautifulSoup (bs4):** para analizar y extraer información específica del HTML mediante técnicas de *web scraping*.
- **Pandas:** para organizar y manipular los datos en forma de tablas (DataFrames).
- **Matplotlib y Seaborn:** para la creación de gráficos y visualizaciones estadísticas.
- **OS:** para gestionar rutas y guardar los archivos generados en formato CSV.

### Requisitos de instalación o ejecución

Para ejecutar correctamente el programa, es necesario contar con los siguientes elementos instalados:

1. **Python 3.10 o superior.**
2. Las librerías utilizadas en el proyecto, que pueden instalarse desde la terminal con los siguientes comandos:

```
pip install requests
pip install beautifulsoup4
pip install pandas
pip install matplotlib
pip install seaborn
```

3. **Conexión a internet**, ya que el programa obtiene los datos directamente desde la página *Transfermarkt*.
4. Un entorno de ejecución compatible, como **VS Code**, **PyCharm** o la consola de **Python**, para correr el archivo principal (`main.py`).

## 6. Diseño Del Proyecto

El programa fue desarrollado bajo una estructura modular, lo que significa que está dividido en varias funciones independientes, cada una con una tarea específica. Esto facilita la lectura, el mantenimiento y la reutilización del código.

El proyecto se organiza de la siguiente manera:

**1. Importación de librerías:**

Se cargan las librerías necesarias (`requests`, `BeautifulSoup`, `pandas`, `os`) para realizar la conexión web, extraer los datos y manipular la información.

**2. Funciones principales de scraping:**

- a. `obtener_tabla_posiciones()`: se encarga de conectarse a la página de Transfermarkt y extraer la tabla de posiciones de LaLiga.
- b. `obtener_goleadores()`: obtiene los datos de los máximos goleadores de la temporada.  
Ambas funciones devuelven los datos organizados en un DataFrame de pandas.

**3. Funciones de utilidad:**

- a. `mostrar_tabla(df, titulo)`: muestra las tablas obtenidas de forma ordenada en consola.
- b. `guardar_csv(df, nombre_archivo)`: guarda los datos en archivos CSV dentro de la carpeta del proyecto.

**4. Función principal (`main()`):**

Contiene el menú interactivo que permite al usuario elegir qué desea hacer:

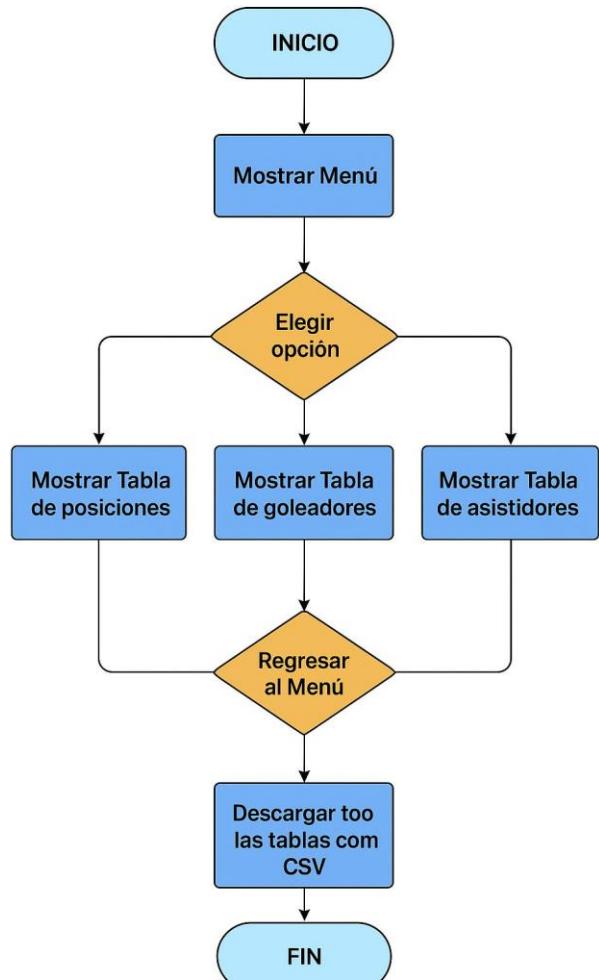
- a. Ver la tabla de posiciones.
  - b. Ver la tabla de goleadores.
  - c. Descargar las tablas.
  - d. Salir del programa.
- Esta función usa un ciclo `while` para mantener activo el menú hasta que el usuario decida salir.

**5. Bloque de ejecución:**

El código termina con

```
if __name__ == "__main__":  
    main()
```

Esto asegura que el menú solo se ejecute cuando el archivo se ejecute directamente, y no al importarlo desde otro script.



```

ons/ms-python.debugpy-2025.16.0/bundled/libs/debugpy/adapter/../../debugpy/launcher 53231 --
--- MENÚ LIGA ESPAÑOLA (TEMPORADA 2024/25) ---
1. Ver tabla de posiciones
2. Ver tabla de goleadores
3. Ver tabla de asistidores (no disponible)
4. Descargar tablas (sin asistidores)
5. Salir

Elige una opción: 4

Descargando tablas...
 Tabla de posiciones obtenida correctamente.
Archivo guardado como: /workspaces/Análisis-de-datos-de-LA-LIGA/tabla_posiciones.csv
 Tabla de goleadores obtenida correctamente.
Archivo guardado como: /workspaces/Análisis-de-datos-de-LA-LIGA/tabla_goleadores.csv
 Tablas descargadas correctamente.

--- MENÚ LIGA ESPAÑOLA (TEMPORADA 2024/25) ---
1. Ver tabla de posiciones
2. Ver tabla de goleadores
3. Ver tabla de asistidores (no disponible)
4. Descargar tablas (sin asistidores)
5. Salir

Elige una opción: 1
 Tabla de posiciones obtenida correctamente.

=====
                    TABLA DE POSICIONES
=====

Pos Equipo      PJ  G   E   P GF    GC DG Pts
 1     Barcelona 38 28  4  6 102:39 63  88
 2     Real Madrid 38 26  6  6 78:38 40  84
 3     Atlético 38 22 10  6 68:30 38  76
 4     Athletic Club 38 19 13  6 54:29 25  70
 5     Villarreal 38 20 10  8 71:51 20  70
 6     Real Betis 38 16 12 10 57:50 7  60
 7     Celta de Vigo 38 16  7 15 59:57 2  55
 8     Rayo Vallecano 38 13 13 12 41:45 -4  52
 9     CA Osasuna 38 12 16 10 48:52 -4  52
10     RCD Mallorca 38 13  9 16 35:44 -9  48

```

## 7. Desarrollo

### 1. Desarrollo paso a paso

#### 1. Definición del problema y objetivos

- a. Decidimos analizar datos de LaLiga (tabla de posiciones y goleadores) para practicar web scraping y análisis con Python.
- b. Se fijaron 3 objetivos: recolectar datos, comparar rendimientos y generar visualizaciones.

#### 2. Elección de fuentes y herramientas

- a. Elegimos **Transfermarkt** como fuente porque tiene tablas completas de la liga.
- b. Herramientas: `requests`, `BeautifulSoup`, `pandas`, `matplotlib/seaborn` y `os`.

#### 3. Planificación y división del trabajo

- a. Definir funciones independientes para cada tarea (modularización).
- b. Una función por tabla (posiciones, goleadores), utilidades para mostrar y guardar, y un `main()` con menú.

#### 4. Implementación básica

- a. Crear funciones de scraping que: hagan la petición HTTP → parseen HTML → extraigan columnas → armen listas → crear `DataFrame`.

#### 5. Pruebas y depuración

- a. Ejecutar el script en consola y revisar la salida.
- b. Comprobar casos en los que la página cambie la estructura (columnas faltantes) y manejar retornos vacíos con mensajes de error.

#### 6. Mejoras y robustez

- a. Incluir `User-Agent` en la petición para reducir bloqueos.
- b. Validar `res.status_code` y existencia de la tabla (`if not tabla:`) para evitar errores de ejecución.

#### 7. Documentación y entrega

- a. Redactar informe con objetivos, metodología, estructura y fragmentos de código comentados.
- b. Preparar presentación y repartir partes entre los integrantes para la exposición.

### 2. Fragmentos de código relevantes comentados

```
# Importaciones necesarias
import requests # para hacer peticiones HTTP a la web
from bs4 import BeautifulSoup # para parsear el HTML y extraer
datos
import pandas as pd # para manejar tablas (DataFrame)
import os # para manejar rutas y guardar
archivos
```

#### Función: obtener\_tabla\_posiciones

```
def obtener_tabla_posiciones():
    """Scrapea la tabla de posiciones de LaLiga 2024/25 desde
    Transfermarkt"""
    url =
"https://www.transfermarkt.com/laliga/tabelle/wettbewerb/ES1/saison\_id/2024"
    headers = {
        # User-Agent para simular un navegador y evitar bloqueos sencillos
        "User-Agent": (
            "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
            "AppleWebKit/537.36 (KHTML, like Gecko) "
            "Chrome/130.0.0.0 Safari/537.36"
    }
```

```

),
"Accept-Language": "es-ES,es;q=0.9,en;q=0.8"
}

# Hacemos la petición a la URL
res = requests.get(url, headers=headers)
if res.status_code != 200:
    # Si no responde bien, devolvemos un DataFrame vacío
    print(f"Error al conectar ({res.status_code}).")
    return pd.DataFrame()

# Parseamos el HTML con BeautifulSoup
soup = BeautifulSoup(res.text, "html.parser")
tabla = soup.find("table", {"class": "items"}) # localizamos la tabla
por su clase
if not tabla:
    print("No se encontró la tabla de posiciones.")
    return pd.DataFrame()

# Recorremos las filas y guardamos las columnas en listas
filas = tabla.find("tbody").find_all("tr", recursive=False)
posiciones, equipos, pj, g, e, p, gf, gc, dg, pts = ([] for _ in
range(10))

for fila in filas:
    columnas = fila.find_all("td", recursive=False)
    if len(columnas) >= 10: # comprobación mínima de columnas
esperadas
        posiciones.append(columnas[0].get_text(strip=True))
        equipo_tag = columnas[1].find("a")
        equipos.append(equipo_tag.get_text(strip=True) if equipo_tag
else columnas[1].get_text(strip=True))
        pj.append(columnas[2].get_text(strip=True))
        g.append(columnas[3].get_text(strip=True))
        e.append(columnas[4].get_text(strip=True))
        p.append(columnas[5].get_text(strip=True))
        gf.append(columnas[6].get_text(strip=True))
        gc.append(columnas[7].get_text(strip=True))
        dg.append(columnas[8].get_text(strip=True))
        pts.append(columnas[9].get_text(strip=True))

# Creamos un DataFrame con las listas recogidas
df = pd.DataFrame({
    "Pos": posiciones,
    "Equipo": equipos,
    "PJ": pj,
    "G": g,
    "E": e,
    "P": p,
    "GF": gf,
    "GC": gc,
    "DG": dg,
    "Pts": pts
})
print("✅ Tabla de posiciones obtenida correctamente.")

```

```
    return df
```

#### Función: obtener\_goleadores

```
def obtener_goleadores():
    """Scraapea los goleadores de LaLiga 2024/25 desde Transfermarkt"""
    url =
https://www.transfermarkt.com/laliga/torschuetzenliste/wettbewerb/ES1/saison\_id/2024
    headers = {
        "User-Agent": "Mozilla/5.0 ...", # similar al anterior
        "Accept-Language": "es-ES,es;q=0.9,en;q=0.8"
    }

    res = requests.get(url, headers=headers)
    if res.status_code != 200:
        print(f"Error al conectar ({res.status_code}).")
        return pd.DataFrame()

    soup = BeautifulSoup(res.text, "html.parser")
    tabla = soup.find("table", {"class": "items"})
    if not tabla:
        print("No se encontró la tabla de goleadores.")
        return pd.DataFrame()

    filas = tabla.find("tbody").find_all("tr", recursive=False)
    posiciones, jugadores, equipos, goles = [], [], [], []

    for fila in filas:
        columnas = fila.find_all("td", recursive=False)
        if len(columnas) < 5:
            continue

        posiciones.append(columnas[0].get_text(strip=True))

        # Localiza el primer enlace con '/spieler/' que normalmente
        # contiene el nombre
        jugador_tag = fila.find("a", href=lambda x: x and "/spieler/" in x)
        jugador = jugador_tag.get_text(strip=True) if jugador_tag else
        "Desconocido"
        jugadores.append(jugador)

        # Localiza el primer enlace con '/verein/' para el equipo
        equipo_tag = fila.find("a", href=lambda x: x and "/verein/" in x)
        equipo = equipo_tag.get_text(strip=True) if equipo_tag else "Sin
        equipo"
        equipos.append(equipo)

        # Busca la última celda que sea un número (los goles suelen ser
        # dígitos)
        goles_encontrado = None
        for td in reversed(columnas):
            texto = td.get_text(strip=True)
            if texto.isdigit():
```

```

        goles_encontrado = texto
        break
    goles.append(goles_encontrado if goles_encontrado else "0")

df = pd.DataFrame({
    "Pos": posiciones,
    "Jugador": jugadores,
    "Equipo": equipos,
    "Goles": goles
})

print("✅ Tabla de goleadores obtenida correctamente.")
return df

```

### Funciones utilitarias y main

```

def mostrar_tabla(df, titulo):
    print("\n" + "=" * 80)
    print(titulo.center(80))
    print("=" * 80)
    if df.empty:
        print("No hay datos para mostrar.\n")
    else:
        print(df.head(10).to_string(index=False)) # muestra los primeros
10 registros
        print("\n(Se muestran los primeros 10 registros)\n")

def guardar_csv(df, nombre_archivo):
    if df.empty:
        print(f"No se pudo guardar {nombre_archivo} porque está vacío.")
        return
    ruta = os.path.join(os.getcwd(), nombre_archivo)
    df.to_csv(ruta, index=False, encoding="utf-8-sig")
    print(f"Archivo guardado como: {ruta}")

def main():
    while True:
        print("== MENÚ LIGA ESPAÑOLA (TEMPORADA 2024/25) ==")
        print("1. Ver tabla de posiciones")
        print("2. Ver tabla de goleadores")
        print("3. Ver tabla de asistidores (no disponible)")
        print("4. Descargar tablas (sin asistidores)")
        print("5. Salir")

        opcion = input("\nElige una opción: ")

        if opcion == "1":
            df = obtener_tabla_posiciones()
            mostrar_tabla(df, "TABLA DE POSICIONES")

        elif opcion == "2":
            df = obtener_goleadores()
            mostrar_tabla(df, "TABLA DE GOLEADORES")

```

```

        elif opcion == "3":
            print("\n\tX Tabla de asistidores no disponible
actualmente.\n")

        elif opcion == "4":
            print("\nDescargando tablas...")
            guardar_csv(obtener_tabla_posiciones(),
"tabla_posiciones.csv")
            guardar_csv(obtener_goleadores(), "tabla_goleadores.csv")
            print("✓ Tablas descargadas correctamente.\n")

        elif opcion == "5":
            print("\nSaliendo del programa...")
            break

    else:
        print("\nOpción no válida, intenta de nuevo.\n")

if __name__ == "__main__":
    main()

```

### 3. Descripción de las funciones principales

- **obtener\_tabla\_posiciones()**  
Conecta con la página de Transfermarkt de la tabla de LaLiga, busca la tabla HTML, recorre sus filas y extrae las columnas relevantes (posición, equipo, PJ, G, E, P, GF, GC, DG, Pts). Devuelve un DataFrame con esos datos. Maneja errores de conexión y ausencia de la tabla.
- **obtener\_goleadores()**  
Conecta con la página de goleadores, extrae la posición, el nombre del jugador (buscando enlaces que contengan /spieler/), el equipo (enlaces con /verein/) y el número de goles (buscando la última celda con dígitos). Devuelve un DataFrame. Incluye comprobaciones para estructuras inconsistentes.
- **mostrar\_tabla(df, titulo)**  
Imprime en consola un encabezado estético y muestra los primeros 10 registros del DataFrame. Si el DataFrame está vacío, informa al usuario.
- **guardar\_csv(df, nombre\_archivo)**  
Guarda el DataFrame como un archivo CSV en el directorio actual. Verifica que el DataFrame no esté vacío y muestra la ruta del archivo guardado.
- **main()**  
Interfaz de usuario simple por consola que despliega un menú y permite ejecutar las funciones anteriores según la opción seleccionada. Mantiene el programa en un bucle hasta que el usuario elija salir.

## 8. Pruebas Y Resultados

Pruebas del programa

Para verificar el correcto funcionamiento del programa, se realizaron diversas **pruebas manuales** desde la consola de Python.

El proceso de pruebas se centró en comprobar que cada función cumpliera con su propósito y que el

flujo completo del menú funcionara sin errores.

## **1. Prueba de conexión a la web**

- a. Se ejecutaron las funciones `obtener_tabla_posiciones()` y `obtener_goleadores()` individualmente para confirmar que la conexión con *Transfermarkt* fuera exitosa.
- b. Se revisó que el código manejara correctamente los errores cuando la página no respondía (por ejemplo, devolviendo un mensaje de “Error al conectar”).

## **2. Prueba de extracción de datos**

- a. Se inspeccionaron las tablas obtenidas para confirmar que los datos coincidieran con la información real de la página web.
- b. Se validó que no se repitieran filas ni se perdieran columnas importantes (posición, equipo, puntos, goles, etc.).

## **3. Prueba del menú principal**

- a. Se ejecutó el programa completo con el comando `python main.py` y se probaron todas las opciones del menú (1 al 5).
- b. Se verificó que las opciones inválidas mostraran el mensaje “Opción no válida, intenta de nuevo”.

## **4. Prueba de exportación de datos**

- a. Al seleccionar la opción “Descargar tablas”, se comprobó que los archivos CSV se generaran correctamente en la carpeta del proyecto y que los datos se guardaran con formato legible.

## **5. Pruebas de validación y vacíos**

- a. Se simularon casos donde la página no devolvía datos (por ejemplo, desconectando internet) para confirmar que el programa no se cerrara abruptamente y mostrara los mensajes de advertencia adecuados.

En conjunto, las pruebas demostraron que el programa cumple su función principal: obtener y mostrar correctamente las tablas de posiciones y goleadores de LaLiga.

```
ons/ms-python.debugpy-2025.16.0/bundled/libs/debugpy/adapter/../../debugpy/launcher 53231 --
--- MENÚ LIGA ESPAÑOLA (TEMPORADA 2024/25) ---
1. Ver tabla de posiciones
2. Ver tabla de goleadores
3. Ver tabla de asistidores (no disponible)
4. Descargar tablas (sin asistidores)
5. Salir

Elige una opción: 4

Descargando tablas...
 Tabla de posiciones obtenida correctamente.
Archivo guardado como: /workspaces/An-lisis-de-datos-de-LA-LIGA/tabla_posiciones.csv
 Tabla de goleadores obtenida correctamente.
Archivo guardado como: /workspaces/An-lisis-de-datos-de-LA-LIGA/tabla_goleadores.csv
 Tablas descargadas correctamente.

--- MENÚ LIGA ESPAÑOLA (TEMPORADA 2024/25) ---
1. Ver tabla de posiciones
2. Ver tabla de goleadores
3. Ver tabla de asistidores (no disponible)
4. Descargar tablas (sin asistidores)
5. Salir

Elige una opción: 1
 Tabla de posiciones obtenida correctamente.

=====
                TABLA DE POSICIONES
=====

Pos Equipo      PJ  G   E   P GF      GC DG Pts
 1     Barcelona 38 28  4  6 102:39 63  88
 2     Real Madrid 38 26  6  6 78:38 40  84
 3     Atlético 38 22 10  6 68:30 38  76
 4     Athletic Club 38 19 13  6 54:29 25  70
 5     Villarreal 38 20 10  8 71:51 20  70
 6     Real Betis 38 16 12 10 57:50  7  60
 7     Celta de Vigo 38 16  7 15 59:57  2  55
 8     Rayo Vallecano 38 13 13 12 41:45 -4  52
 9     CA Osasuna 38 12 16 10 48:52 -4  52
10     RCD Mallorca 38 13  9 16 35:44 -9  48
```

1. Ver tabla de posiciones
2. Ver tabla de goleadores
3. Ver tabla de asistidores (no disponible)
4. Descargar tablas (sin asistidores)
5. Salir

Elige una opción: 2

Tabla de goleadores obtenida correctamente.

=====

TABLA DE GOLEADORES

=====

Pos	Jugador	Equipo	Goles
1	Kylian Mbappé		31
2	Robert Lewandowski		27
3	Ante Budimir		21
4	Alexander Sørloth		20
5	Ayoze Pérez		19
6	Raphinha		18
7	Julián Alvarez		17
8	Oihan Sanctet		15
9	Kike García		13
10	Javi Puado		12

(Se muestran los primeros 10 registros)

==== MENÚ LIGA ESPAÑOLA (TEMPORADA 2024/25) ===

1. Ver tabla de posiciones
2. Ver tabla de goleadores
3. Ver tabla de asistidores (no disponible)
4. Descargar tablas (sin asistidores)
5. Salir

Elige una opción: 3

Tabla de asistidores no disponible actualmente.

```
== MENU LIGA ESPANOLA (TEMPORADA 2024/25) ==
1. Ver tabla de posiciones
2. Ver tabla de goleadores
3. Ver tabla de asistidores (no disponible)
4. Descargar tablas (sin asistidores)
5. Salir

Elige una opción: 3

✗ Tabla de asistidores no disponible actualmente.

== MENÚ LIGA ESPAÑOLA (TEMPORADA 2024/25) ==
1. Ver tabla de posiciones
2. Ver tabla de goleadores
3. Ver tabla de asistidores (no disponible)
4. Descargar tablas (sin asistidores)
5. Salir

Elige una opción: 5

Saliendo del programa...
@AndresDAVM →/workspaces/An-lisis-de-datos-de-LA-LIGA (main) $
```

## Manual de usuario

A continuación, se describen los pasos que debe seguir el usuario para ejecutar y utilizar el programa correctamente:

### 1. Requisitos previos

- a. Tener **Python 3.10 o superior** instalado.
- b. Contar con conexión a internet.
- c. Instalar las librerías necesarias ejecutando los siguientes comandos en la terminal:

```
pip install requests
pip install beautifulsoup4
pip install pandas
pip install matplotlib
pip install seaborn
```

### 2. Ejecución del programa

- a. Abrir la carpeta donde se encuentra el archivo principal (por ejemplo, **main.py**).
- b. Desde la terminal o el entorno de desarrollo (VS Code, PyCharm, etc.), ejecutar:

```
python main.py
```

### 3. Uso del menú principal

Una vez ejecutado, el programa mostrará un menú con las siguientes opciones:

```
== MENU LIGA ESPAÑOLA (TEMPORADA 2024/25) ==
```

1. Ver tabla de posiciones
  2. Ver tabla de goleadores
  3. Ver tabla de asistidores (no disponible)
  4. Descargar tablas (sin asistidores)
  5. Salir
- 
- a. **Opción 1:** muestra en consola la tabla de posiciones actual de LaLiga.
  - b. **Opción 2:** muestra la lista de los principales goleadores.
  - c. **Opción 3:** indica que la tabla de asistidores aún no está disponible.
  - d. **Opción 4:** descarga las tablas de posiciones y goleadores en formato .csv dentro de la carpeta del proyecto.
  - e. **Opción 5:** cierra el programa.

#### 4. Archivos generados

- a. Al elegir la opción 4, se crean los siguientes archivos en la carpeta del proyecto:
  - i. `tabla_posiciones.csv`
  - ii. `tabla_goleadores.csv`Estos pueden abrirse con **Excel**, **Google Sheets** o cualquier editor de texto para analizar los datos.

#### 5. Mensajes del programa

- a. Si el sitio web no responde o la tabla no se encuentra, el programa mostrará mensajes como:

Error al conectar (404)  
No se encontró la tabla de posiciones.

Esto indica que no se pudieron obtener los datos en ese momento.

## 9. Conclusiones

### Lecciones aprendidas

Durante el desarrollo del proyecto LaLiga Insights Pro, aprendimos a aplicar de forma práctica muchos de los temas vistos en clase, como el uso de funciones, estructuras de datos y modularización del código.

También comprendimos cómo Python puede utilizarse más allá de ejercicios básicos, permitiendo automatizar tareas reales como la recolección y análisis de información desde internet.

Además, aprendimos a trabajar en equipo, organizando las funciones y responsabilidades de manera que el proyecto avanzara de forma ordenada. Cada uno aportó en distintas partes, desde la búsqueda de la información hasta la programación y las pruebas del código.

Otra lección importante fue entender la importancia de manejar errores y validar datos, ya que al depender de una página web externa, el código debía estar preparado para posibles fallos de conexión o cambios en la estructura del sitio.

## Dificultades encontradas y cómo se resolvieron

- Problemas con el scraping:  
Al inicio, el programa no encontraba las tablas de Transfermarkt porque la estructura HTML era más compleja de lo esperado. Esto se resolvió usando el método `find()` con la clase "items" y accediendo directamente al elemento `<tbody>` para recorrer las filas.
- Errores de conexión (403 o 404):  
Algunas peticiones eran bloqueadas por la página. Para solucionarlo, agregamos un encabezado User-Agent, haciendo que la petición pareciera provenir de un navegador real.
- Datos incompletos o vacíos:  
En algunos casos, las tablas se descargaban parcialmente o con valores vacíos. Se agregaron validaciones para mostrar mensajes de advertencia y evitar que el programa se cerrara abruptamente.
- Formateo de salida:  
Al guardar los archivos CSV, se presentaban errores con los acentos. Esto se corrigió usando la codificación "utf-8-sig" al momento de guardar los archivos.

En general, cada problema sirvió para entender mejor cómo funcionan las peticiones web, el manejo de errores en Python y la importancia de hacer pruebas constantes.

## Posibles mejoras o ideas futuras

- Agregar una función para mostrar la tabla de asistencias, ampliando el análisis más allá de goles y posiciones.
- Implementar gráficos automáticos con Matplotlib o Seaborn para visualizar la evolución de los equipos a lo largo de la temporada.
- Crear una interfaz gráfica (GUI) usando Tkinter o Streamlit, para que el usuario pueda interactuar de forma más visual.
- Añadir un sistema de actualización automática, que ejecute el scraping de forma programada y mantenga los datos siempre al día.
- Incorporar modelos predictivos simples, por ejemplo, para estimar el rendimiento final de los equipos o el número de goles de los jugadores.

## 10. Bibliografía

- [Documentación de Python](#)
- [Documentación de Pandas](#)
- [Documentación de BeautifulSoup](#)
- [Documentación de solicitudes](#)
- [Transfermarkt](#)
- [Documentación de Matplotlib](#)
- [Documentación de Seaborn](#)
- [Tutorial de scraping con Python y Real Python](#)
- [Kaggle – Análisis de datos de fútbol](#)
- Artículos sobre análisis estadístico en fútbol

