

Práctica de Programación en Entornos de Datos 2025-2026

1. INSTRUCCIONES

Para poder implementar la solución a la práctica, se requieren una serie de instalaciones previas. A continuación, se proporcionan los enlaces de descarga de *Python* para cada sistema operativo.

- Windows: <https://www.python.org/downloads/windows/>
- Linux: <https://www.python.org/downloads/source/>
- MacOS: <https://www.python.org/downloads/macOS/>
- Otros: <https://www.python.org/download/other/>

Se deben instalar aquellos paquetes que sean necesarios. Para ello, abrimos la aplicación *Python* que se ha instalado (o bien por terminal) e instalamos dichos paquetes.

- Documentación: [The Python Tutorial — Python 3.12.0 documentation](https://docs.python.org/3.12.0/)
- Instalación de paquetes: [12. Virtual Environments and Packages — Python 3.12.0 documentation](https://docs.python.org/3.12.0/library/index.html#module-index)

Serán necesarios aquellos paquetes utilizados durante el contenido de la asignatura, por ejemplo, *numpy*, *pandas*, etc...

MUY IMPORTANTE: El código contenido en el fichero **practica.py** debe funcionar correctamente siendo llamado desde **main.py** sin necesidad de realizar ningún cambio. Por ello, se deben seguir los pasos que se indican durante la práctica de manera obligatoria. **LOS RESULTADOS DEBEN CORRESPONDERSE CON LOS PROPORCIONADOS EN EL FICHERO DE SALIDA DEL JUEGO DE PRUEBAS.**

En esta práctica vamos a plantear un caso práctico que se tendrá que resolver con los contenidos vistos en el curso de manera completamente **individual**. Cada estudiante puede buscar y utilizar todas las funciones adicionales que crea necesarias, justificando su elección y explicando su uso. El uso eficiente de las estructuras de datos, reutilización de código y otras optimizaciones, serán valoradas positivamente.

La entrega de la práctica consiste en **dos ficheros** que se subirán a la entrega de tareas del curso virtual de la convocatoria correspondiente. Estos ficheros son los siguientes:

1. Un fichero de texto de nombre **practica.py** que contendrá las funciones que se indican en cada apartado de la práctica, junto a todas aquellas funciones auxiliares que se consideren necesarias.
2. Un fichero **memoria.pdf** que contendrá la información adicional que se indica en cada apartado de la práctica.

Como el sistema solo permite la entrega de un único fichero, **los dos ficheros anteriores se deberán comprimir y subir un fichero comprimido (.zip).**

1. ENUNCIADO DE LA PRÁCTICA

Una red de bibliotecas públicas quiere analizar los préstamos de libros realizados por sus usuarios durante varios años, con el fin de identificar patrones y hábitos de lectura. Cada préstamo tiene una duración máxima de 30 días.

La información se ha recopilado en múltiples ficheros CSV, uno por año natural. La primera línea de cada CSV contiene una cadena de texto con el año en cuestión y desde la segunda línea en adelante cada línea a un préstamo individual registrado por el sistema en el momento en que finaliza con los siguientes campos separados por comas:

- Día del mes (1-31)
- Nombre del mes (“Enero”, “Febrero” ...).
- Identificador del lector/a (cadena de texto).
- Título del libro (cadena de texto).
- Autor/a del libro (cadena de texto).
- Género literario (cadena de texto).
- Duración del préstamo en días (entero no negativo).
- Información sobre si este préstamo es una renovación de uno anterior (“Si” / “No”).

2. TRABAJO A REALIZAR

Para cada uno de los siguientes apartados será necesario realizar la implementación de la función (o funciones) indicada(s) en el fichero `practica.py` y aportar, en el documento `memoria.pdf`, respuesta a las posibles cuestiones planteadas.

1. Diseña un *DataFrame* de Pandas con la información que consideres adecuada dado el enunciado. Explica todas las decisiones tomadas y qué optimizaciones de memoria consideras que se pueden realizar sin comprometer los datos.
2. Una vez diseñado el *DataFrame* de Pandas, será necesario cargar toda la información en memoria con la información de todos los préstamos realizados a lo largo del tiempo.

Para ello se deberá crear la función `loadData(file_paths)`, que recibirá una lista con la ruta completa y nombre de todos los ficheros de datos “.csv” y devolverá un *DataFrame* de Pandas con los préstamos de todos los años que seguirá la estructura definida en el ejercicio anterior. Al cargar los datos se aplicarán, también, todas las optimizaciones de memoria indicadas en el ejercicio anterior.

En el documento `memoria.pdf` se pide explicar el procedimiento seguido para cargar todos los ficheros.

3. Uno de los objetivos principales del análisis de préstamos en la biblioteca es conocer el interés de sus lectores y llevar un control de los propios préstamos. Para ello, vamos a realizar una serie de funciones que permitan analizar diferentes situaciones.

- a) Queremos saber cuál es el libro más prestado, tanto de todo el periodo del que se tienen datos como de cada año en particular. Para ello se realizarán dos funciones:
- **bestBook(df)**, que recibe el *DataFrame* 'df' de Pandas y devuelve una tupla de tres elementos indicando el título del libro más prestado durante todo el periodo analizado, junto con su autor y género literario.
 - **bestBookYear(df, y)**, que recibe el *DataFrame* 'df' de Pandas y un entero 'y' y devuelve una tupla de tres elementos indicando el título del libro más prestado durante el año y, junto con su autor y género literario.
- b) Análogamente al apartado anterior, queremos saber cuál es el género más prestado, tanto de todo el periodo analizado, como de cada año en particular. Para ello se realizarán dos funciones:
- **bestGenre(df)**, que recibe el *DataFrame* 'df' de Pandas y devuelve el género literario más prestado durante todo el periodo analizado.
 - **bestGenreYear(df, y)**, que recibe el *DataFrame* 'df' de Pandas y un entero 'y' y devuelve el género literario más prestado durante el año y.
- c) Otro dato interesante es saber cuál es el lector o lectora con más préstamos, en global y por cada año. Para ello se emplearán las funciones:
- **bestReader(df)**, que recibe el *DataFrame* 'df' de Pandas y devuelve una tupla de tres elementos indicando el identificador del usuario con más préstamos en global, indicando, además, cuál es su género y autor favoritos.
 - **bestReaderYear(df, y)**, que recibe el *DataFrame* 'df' de Pandas y un entero 'y' y devuelve una tupla de tres elementos indicando el identificador del usuario con más préstamos en el año y, indicando, además, cuál es su género y autor favoritos.
- d) Siguiendo con el tema anterior, ahora queremos saber quién fue el lector o lectora que más días ha tenido libros prestados (tanto el global, como por año) y cuántos préstamos ha formalizado. Para ello habrá que programar las funciones:
- **mostAccumulatedDays(df)**, que recibe el *DataFrame* 'df' de Pandas y devuelve una tupla de dos elementos indicando el identificador del usuario con más días de préstamos durante todo el periodo de tiempo analizado, indicando, además, el número de préstamos que ha formalizado en total.
 - **mostAccumulatedDaysYear(df, y)**, que recibe el *DataFrame* 'df' de Pandas y un entero 'y' y devuelve una tupla de dos elementos indicando el identificador del usuario con más días de préstamos en el año y, indicando, además, el número de préstamos que ha formalizado durante ese tiempo.
- e) Por último, queremos una función que nos proporcione información sobre los préstamos de un usuario concreto. Para ello, programaremos las funciones:

- **userInfo(df, u)**, que recibe el *DataFrame* 'df' de Pandas y el identificador 'u' de un usuario y devuelve un diccionario con los siguientes datos (clave:contenido) sobre dicho usuario durante todo el periodo de tiempo analizado:
 - "Prestamos": número de préstamos formalizados.
 - "Media duración préstamos": número medio de días que ha durado cada préstamo.
 - "Géneros": un diccionario con el número de préstamos de cada género literario leído por el usuario.
 - "Mes más lector": el nombre del mes en el que más préstamos ha formalizado.
- **userInfoYear(df, u, y)**, que recibe el *DataFrame* 'df' de Pandas, el identificador 'u' de un usuario y un entero 'y' y devuelve un diccionario con los siguientes datos (clave:contenido) referidos al lector con más días de préstamo acumulados durante el año y:
 - "Prestamos": número de préstamos formalizados.
 - "Media duración préstamos": número medio de días que ha durado cada préstamo.
 - "Géneros": un diccionario con el número de préstamos de cada género literario leído por el usuario.
 - "Mes más lector": el nombre del mes en el que más préstamos ha formalizado.

Para la realización de este apartado, debe reutilizar código de forma que se minimice el código repetido en las funciones anteriores. En el documento [memoria.pdf](#) se pide explicar cómo se obtienen los diferentes resultados y también cómo se ha reutilizado el código.

4. Otro dato que se desea conocer es la información correspondiente a cuánto tardan los libros en ser leídos, incluyendo información sobre las renovaciones de los préstamos.
 - a) Queremos saber cuál es el libro que más se ha renovado tanto durante todo el periodo de tiempo, como por cada año. Para ello realizaremos las siguientes funciones:
 - **mostRenewedBook(df)**, que recibe el *DataFrame* 'df' de Pandas y devuelve una tupla de tres elementos indicando el título del libro más renovado durante todo el periodo analizado, junto con su autor y género literario.
 - **mostRenewedBookYear(df, y)**, que recibe el *DataFrame* 'df' de Pandas y un entero 'y' y devuelve una tupla de tres elementos indicando el título del libro más renovado durante el año y, junto con su autor y género literario.
 - b) Ahora queremos saber cuál es el libro que se ha leído más rápido, tanto en global como de cada año. Para ello se realizarán dos funciones:
 - **fastestBook(df)**, que recibe el *DataFrame* 'df' de Pandas y devuelve una tupla de tres elementos indicando el título del libro que menos días ha estado prestado durante todo el periodo analizado, junto con su autor y la media de duración de cada préstamo.
 - **fastestBookYear(df, y)**, que recibe el *DataFrame* 'df' de Pandas y un entero 'y' y devuelve una tupla de tres elementos indicando el título del libro que menos días ha

estado prestado durante el año y, junto con su autor y la media de duración de cada préstamo.

c) De igual manera, queremos saber cuál es el libro que más les ha costado leer a los usuarios de la biblioteca, tanto en global como para cada año. Se programarán, para ello, las siguientes funciones:

- **slowestBook(df)**, que recibe el *DataFrame* 'df' de Pandas y devuelve una tupla de tres elementos indicando el título del libro que más días ha estado prestado durante todo el periodo analizado, junto con su autor y la media de duración de cada préstamo.
- **slowestBookYear(df, y)**, que recibe el *DataFrame* 'df' de Pandas y un entero 'y' y devuelve una tupla de tres elementos indicando el título del libro que menos días ha estado prestado durante el año y, junto con su autor y la media de duración de cada préstamo.

d) De manera similar a lo anterior, ahora queremos saber el género literario que más rápido se ha leído en media, tanto durante todo el periodo como por años. Para ello se usarán las siguientes funciones:

- **fastestGenre(df)**, que recibe el *DataFrame* 'df' de Pandas y devuelve el género que menos días ha estado prestado durante todo el periodo analizado y la media de duración de cada préstamo.
- **fastestGenreYear(df, y)**, que recibe el *DataFrame* 'df' de Pandas y un entero 'y' y devuelve el género que menos días ha estado prestado durante ese año y la media de duración de cada préstamo.

e) Para finalizar, queremos saber el género literario que más ha tardado en leerse en media, en total y para cada año. Usaremos estas funciones:

- **slowestGenre(df)**, que recibe el *DataFrame* 'df' de Pandas y devuelve el género que más días ha estado prestado durante todo el periodo analizado y la media de duración de cada préstamo.
- **slowestGenreYear(df, y)**, que recibe el *DataFrame* 'df' de Pandas y un entero 'y' y devuelve el género que más días ha estado prestado durante ese año y la media de duración de cada préstamo.

Para la realización de este apartado, debe reutilizar código de forma que se minimice el código repetido en las funciones anteriores. En el documento [memoria.pdf](#) se pide explicar cómo se obtienen los diferentes resultados y también cómo se ha reutilizado el código.

5. Se desea implementar un sistema de avisos para llevar el control de los usuarios que han excedido el número de días de préstamo.

a) Enriquecer el *DataFrame* de Pandas creado en el ejercicio 2 con una columna de avisos para aquellos usuarios que han superado el número de días máximo del préstamo (el cual, recordemos, es de 30 días). Se deberá programar una función **addAlerts(df)**, que

reciba el *DataFrame* 'df' y devuelva un *DataFrame* enriquecido con una columna que indique si el préstamo genera un aviso o no.

b) Gracias a las alertas, podemos saber qué usuarios incumplieron el plazo máximo de devolución de libros, tanto en general, como en global. Para ello se pide implementar las siguientes funciones:

- **showAlerts(dfe)**, que recibe el *DataFrame* 'dfe' de Pandas (enriquecido con las alertas) y devuelve una lista de tuplas (lector,número de alertas) ordenada decrecientemente por el número de alertas que cada lector ha generado durante todo el periodo analizado.
- **showAlertsYear(dfe,y)**, que recibe el *DataFrame* 'dfe' de Pandas (enriquecido con las alertas) y un entero 'y' y devuelve una lista de tuplas (lector,número de alertas) ordenada decrecientemente por el número de alertas que cada lector ha generado durante el año y.

Para la realización de este apartado, debe reutilizar código de forma que se minimice el código repetido en las funciones anteriores. En el documento [memoria.pdf](#) se pide explicar cómo se obtienen los diferentes resultados y también cómo se ha reutilizado el código.

6. Tras ver los resultados del estudio anterior, se constata que algunos usuarios tienen la mala costumbre de no devolver los préstamos a tiempo. Por lo tanto, la biblioteca plantea la necesidad de saber cuántos días de exceso suele haber.

Para ello se pide programar la función **tooLate(dfe)**, que recibe el *DataFrame* 'dfe' de Pandas (enriquecido con las alertas) y devuelve una tupla con los siguientes tres valores:

- La media de días de exceso de todas las alertas detectadas a lo largo de todo el periodo de tiempo estudiado.
- La media de días de exceso del usuario que acumuló un mayor número total de días de exceso.
- La media de días de exceso del usuario que acumuló un menor número total de días de exceso.

Para la realización de este apartado, debe reutilizar código de forma que se minimice el código repetido en las funciones anteriores. En el documento [memoria.pdf](#) se pide explicar cómo se obtienen los diferentes resultados y también cómo se ha reutilizado el código.

3. ENTREGA Y EVALUACIÓN DE LA PRÁCTICA

La entrega de la práctica se realizará mediante la tarea que se abrirá a tal efecto en el curso virtual, estableciéndose la fecha tope de entrega como el domingo 1 de febrero de 2026.

Las prácticas se evaluarán comprobando que resuelven correctamente el problema, que hagan un uso correcto de las técnicas vistas en la asignatura (por ejemplo empleando adecuadamente operaciones vectorizadas en lugar de bucles) y las explicaciones aportadas en la memoria.

MUY IMPORTANTE: El código contenido en el fichero `practica.py` debe funcionar correctamente siendo llamado desde `main.py` sin necesidad de realizar ningún cambio. Por ello, se deben seguir los pasos que se indican durante la práctica de manera obligatoria. **LOS RESULTADOS DEBEN CORRESPONDERSE CON LOS PROPORCIONADOS EN EL FICHERO DE SALIDA DEL JUEGO DE PRUEBAS.**