Fecha Límite de entrega: domingo 25 de enero, 2026, 23:59:00.

La secretaría del IES Pío Baroja nos ha entregado un archivo llamado notasAlumnos.csv que contiene las calificaciones de DAW de varios módulos profesionales. Lamentablemente, los datos han sido introducidos manualmente y contienen múltiples errores de formato que impiden su procesamiento automático.

Errores que pueden aparecer: espacios en blanco, nombres no normalizados ("Alejandro", "alejandro " y "ALEJANDRO" deben ser tratados como la misma persona), formatos numéricos. Si una nota contiene texto (ej. "pendiente" o "no presentado"), el programa no debe detenerse; simplemente ignorará esa entrada y continuará con la siguiente. Ignoraremos también aquellas notas que no tengan un formato decimal válido “.” (ej. “4,5” en lugar de “4.5”).

Además del archivo de notas, el programa debe leer un segundo archivo llamado becas.txt. Debes cruzar la información de ambos archivos. En el informe final y en la gráfica, se debe indicar claramente si el alumno cuenta con beca o no.

Modificación de los archivos proporcionados

1. Incluye tu propio nombre en el archivo notasAlumnos.csv con notas en 3 módulos
2. Incluye tu propio nombre en el archivo becas.txt indicando si tienes beca o no.

Debes desarrollar un programa en Python que limpie, organice y visualice esta información aplicando tipos avanzados de datos.

No se permite el uso de librerías externas de gestión de datos como Pandas.

Debes utilizar un Diccionario principal donde la clave sea el nombre del alumno.

El valor de cada clave debe ser una Lista que almacene todas las notas numéricas encontradas para ese alumno.

El informe por consola ahora tendrá 4 columnas: Nombre, Promedio, Estado y Beca.

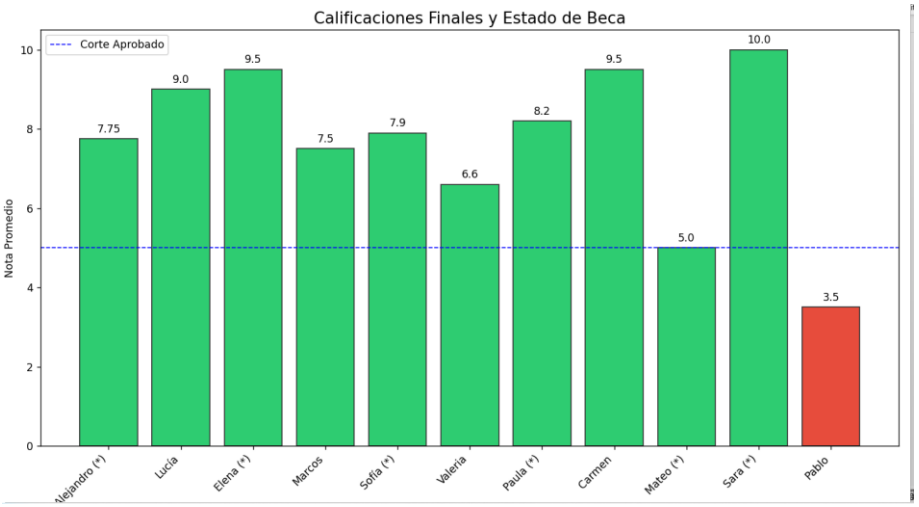
Salida por consola:

```
ALUMNO          | NOTA   | ESTADO     | BECA
-----------------------------------------------
Alejandro       | 7.75   | APROBADO   | SI
Lucía           | 9.00   | APROBADO   | NO
Elena           | 9.50   | APROBADO   | SI
Marcos          | 7.50   | APROBADO   | NO
Sofía           | 7.90   | APROBADO   | SI
Valeria         | 6.60   | APROBADO   | NO
Paula           | 8.20   | APROBADO   | SI
Carmen          | 9.50   | APROBADO   | NO
Mateo           | 5.00   | APROBADO   | SI
Sara            | 10.00  | APROBADO   | SI
Pablo           | 3.50   | SUSPENSO   | NO
```

Visualización de Datos

Utiliza la librería matplotlib para generar una gráfica de barras:

- **Eje X:** Nombres de los alumnos.
- **Eje Y:** Nota media.
- **Lógica de color:** Las barras de los alumnos aprobados deben ser de color **verde** y las de los suspensos de color **rojo**.
- Debe incluir una línea horizontal discontinua en el valor 5.0 para marcar el límite de aprobado.



Instrucciones para la entrega:

- Debes crear un repositorio en GitHub con tu usuario, que será el enlace que entregarás para la corrección. **Cualquier commit que se realice pasada la fecha de entrega, la anulará por completo y la nota será 0**. Debes incluir un **.gitignore** para evitar subir al repositorio cualquier fichero que no se indique en la jerarquía descrita más abajo. Tu repositorio debe contener únicamente estos ficheros:
  - ej30becas.txt → Datos de entrada de becas
  - ej30notasdaw.csv → Datos de entrada de notas de asignaturas
  - main.py → Código de la práctica
  - notes_graph.png → Gráfico generado en la práctica
  - README.md → Documentación del proyecto
  - requirements.txt → Librerías necesarias para la ejecución
- Debes crear un **entorno virtual** (virtualenv) con **venv** para la instalación de las librerías incluidas en el fichero **requirements.txt** que se te ha facilitado a través del aula virtual. *Las instrucciones de creación del entorno virtual a través de venv se explican más abajo.
- Debes crear un fichero **README.md** donde se reflejen los comandos para crear y activar el virtualenv así como para la instalación de las librerías. Se valorará que el README.md incluya también una breve explicación del código de la práctica.
- El flujo principal del código debe estar incluido en un fichero denominado "**main.py**". Se valorará el uso de funciones en este archivo para una mayor legibilidad.

***Instrucciones para la creación del entorno virtual:**

Un entorno virtual (venv) es un directorio aislado que contiene una instalación de Python y unas dependencias específicas para un proyecto. Esto evita conflictos entre las librerías de diferentes proyectos.

Abre una terminal y accede al directorio del proyecto donde vas a realizar la práctica (en este caso, el directorio es **python-dicts**. Una vez dentro del directorio del proyecto, crea el entorno virtual:

```
PS C:\Users\Alumno\Documents\workspace\alvasenj\python-dicts> python -m venv practica_diccionarios
```

La activación varía según el sistema operativo y la terminal que uses:

- En macOS y Linux (Bash/Zsh):

```
> source practica_diccionarios/bin/activate
```

- En Windows (Command Prompt - CMD):

```
> practica_diccionarios\Scripts\activate.bat
```

- En Windows (PowerShell, terminal por defecto en VS Code o PyCharm):

```
PS C:\Users\Alumno\Documents\workspace\alvasenj\python-dicts> .\practica_diccionarios\Scripts\Activate.ps1
```

*Un signo de que la activación ha ido bien será que se muestre el nombre de tu entorno virtual entre paréntesis en tu terminal.

```
(practica_diccionarios) PS C:\Users\Alumno\Documents\workspace\alvasenj\python-dicts>
```

Si al ejecutar este comando recibes un error sobre que la ejecución de scripts está deshabilitada, abre una nueva terminal de PowerShell y ejecuta lo siguiente (solo necesitas hacerlo una vez en tu equipo):

```
> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

Confirma la acción con `S` o `Y` y vuelve a intentar la activación.

En el aula virtual, junto con el enunciado y los ficheros de entrada .csv y .txt se te ha proporcionado un fichero **requirements.txt** con las librerías que vas a necesitar para poder realizar la práctica. Incluye este fichero en el directorio raíz del proyecto. En nuestro caso **python-dicts.** Para esta práctica, solo necesitaremos la librería de Python **“matplotlib”**. Ahora, con el entorno activado tras el paso anterior y dentro de tu directorio, instala las librerías necesarias en tu entorno virtual **“practica_diccionarios”**.

Tu salida deberá ser algo como esto:

```
(practica_diccionarios) PS C:\Users\Alumno\Documents\workspace\alvasenj\python-dicts> python -m pip install -r requirements.txt
Collecting matplotlib (from -r requirements.txt (line 1))
  Downloading matplotlib-3.10.8-cp314-cp314-win_amd64.whl.metadata (52 kB)
Collecting contourpy>=1.0.1 (from matplotlib->-r requirements.txt (line 1))
  Downloading contourpy-1.3.3-cp314-cp314-win_amd64.whl.metadata (5.5 kB)
Collecting cycler>=0.10 (from matplotlib->-r requirements.txt (line 1))
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib->-r requirements.txt (line 1))
  Downloading fonttools-4.61.1-cp314-cp314-win_amd64.whl.metadata (116 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib->-r requirements.txt (line 1))
  Downloading kiwisolver-1.4.9-cp314-cp314-win_amd64.whl.metadata (6.4 kB)
Collecting numpy>=1.23 (from matplotlib->-r requirements.txt (line 1))
  Downloading numpy-2.4.1-cp314-cp314-win_amd64.whl.metadata (6.6 kB)
Collecting packaging>=20.0 (from matplotlib->-r requirements.txt (line 1))
  Downloading packaging-25.0-py3-none-any.whl.metadata (3.3 kB)
Collecting pillow>=8 (from matplotlib->-r requirements.txt (line 1))
  Downloading pillow-12.1.0-cp314-cp314-win_amd64.whl.metadata (9.0 kB)
Collecting pyparsing>=3 (from matplotlib->-r requirements.txt (line 1))
  Downloading pyparsing-3.3.1-py3-none-any.whl.metadata (5.6 kB)
Collecting python-dateutil>=2.7 (from matplotlib->-r requirements.txt (line 1))
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting six>=1.5 (from python-dateutil>=2.7->matplotlib->-r requirements.txt (line 1))
  Downloading six-1.17.0-py2.py3-none-any.whl.metadata (1.7 kB)
Downloading matplotlib-3.10.8-cp314-cp314-win_amd64.whl (8.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.3/8.3 MB 35.5 MB/s  0:00:00
Downloading contourpy-1.3.3-cp314-cp314-win_amd64.whl (232 kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.61.1-cp314-cp314-win_amd64.whl (2.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.3/2.3 MB 74.9 MB/s  0:00:00
Downloading kiwisolver-1.4.9-cp314-cp314-win_amd64.whl (75 kB)
Downloading numpy-2.4.1-cp314-cp314-win_amd64.whl (12.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.4/12.4 MB 6.2 MB/s  0:00:01
Downloading packaging-25.0-py3-none-any.whl (66 kB)
Downloading pillow-12.1.0-cp314-cp314-win_amd64.whl (7.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.2/7.2 MB 48.1 MB/s  0:00:00
Downloading pyparsing-3.3.1-py3-none-any.whl (121 kB)
```

Matplotlib tiene dependencias de otras librerías, por lo que verás que junto a ella se instala el resto de dependencias.

Ahora, si ejecutas el siguiente comando, puedes verificar que la instalación de las librerías en el entorno virtual ha sido un éxito:

```
(practica_diccionarios) PS C:\Users\Alumno\Documents\workspace\alvasenj\python-dicts> python -m pip list
Package         Version
--------------- -----------
contourpy       1.3.3
cycler          0.12.1
fonttools       4.61.1
kiwisolver      1.4.9
matplotlib      3.10.8
numpy           2.4.1
packaging       25.0
pillow          12.1.0
pip             25.3
pyparsing       3.3.1
python-dateutil 2.9.0.post0
six             1.17.0
```

Con estos pasos, ya tendrás tu entorno virtual aislado de las instalaciones de Python y con las librerías necesarias para comenzar a programar.

Rúbrica de evaluación de la práctica (10 puntos)

Nota Importante: La entrega debe realizarse obligatoriamente a través de un enlace a un repositorio de GitHub. De no ser así, la práctica se considerará no entregada y la calificación será de 0, independientemente de los siguientes criterios.

Criterio 1: Funcionalidad del programa (4 Puntos)

(1 pto) Lectura y validación: Lee y valida correctamente los ficheros csv y txt, descartando filas con un número de campos incorrecto.

(1 pto) Limpieza y agrupación: Normaliza los nombres de alumnos, ignora notas no válidas y agrupa todas las notas de cada alumno de forma correcta.

(1 pto) Cálculos: Calcula el promedio exacto para cada alumno y cruza los datos de becas eficientemente.

(1 pto) Visualización: Genera correctamente tanto la tabla en consola como la gráfica notes_graph.png con todos los requisitos visuales (colores, línea de aprobado, etc.).

Criterio 2: Calidad y estructura del código (3 Puntos)

(1 pto) Modularidad: El código está bien organizado en funciones con responsabilidades claras

(1 pto) Eficiencia y estructuras de Datos: Utiliza diccionarios para agrupar notas y para la búsqueda de becas, evitando bucles anidados e ineficientes.

(1 pto) Claridad y estilo: El código es legible, con nombres de variables descriptivos y un formato consistente. Se valora positivamente el uso de docstrings y type hints.

Criterio 3: Estructura y entrega del proyecto (3 Puntos)

(1 pto) Gestión del entorno: El README.md explica cómo crear el entorno virtual y el fichero requirements.txt existe y es correcto.

(1 pto) Documentación (README.md): El README.md es claro, completo e incluye una descripción del proyecto, las instrucciones de instalación/ejecución y una breve explicación del código.

(1 pto) Jerarquía de ficheros: La estructura de ficheros del proyecto es limpia y contiene todos los artefactos necesarios (.gitignore, main.py, README.md, etc.).