GUÍA DE ARCHIVOS

1. Construya un programa en Python que cuente la cantidad de vocales y consonantes en un texto en inglés entregado en un archivo de entrada llamado 'texto.txt', un ejemplo de entrada sería:

```
Now here I go,
Hope I don't break down,
I won't take anything, I don't need anything,
Don't want to exist, I can't persist,
Please stop before I do it again,
Just talk about nothing, let's talk about nothing,
Let's talk about no one, please talk about no one, someone, anyone

You and me have a disease,
You affect me, you infect me,
I'm afflicted, you're addicted,
You and me, you and me
```

2. Construya un programa que encuentre la palabra más larga en un texto, considere que el texto le será entregado desde un archivo llamado 'texto.txt'. Use como base el ejercicio para una sola línea visto en clases.

3. Construya un programa que reciba como entrada un texto a través de archivo y una palabra e indique si la palabra está en el texto o no. No puede usar el operador `in` para este ejercicio, ni funciones o métodos para encontrar substrings o sublistas (tales como .find()).

4. Construya un programa en Python que reciba como entrada una cantidad *n* de filas y una cantidad *m* de columnas y entregue como resultado un archivo con una matriz con números aleatorios de dichas dimensiones, el archivo de salida debe llamarse 'matriz.txt' y cada celda debe contener un número aleatorio entre -n * m y n * m.

5. Construya un programa en Python que reciba como entrada un número *n* y genere el patrón solicitado en cada caso en un archivo 'pattern.txt', considere que, en todos los casos, se asume para el ejemplo que n = 5.

   a)
```
*****
*****
*****
```

```
*****
*****
```

b)
```
     *****
    +++++
   *****
  +++++
*****
```

c)
```
*****
*   *
*   *
*   *
*****
```

d)
```
*
*0
*0*
*0*0
*0*0*
```

e)
```
    *
   **
  * *
 *  *
*****
```

f)
```
*****
 0000
  ***
   00
    *
```

g)
```
*********
 *******
  *****
   ***
    *
```

h)
```
*********
*       *
```

```
    *   *
     * *
      *
```

i)
```
      *
     ***
    *****
   *******
  *********
   *******
    *****
     ***
      *
```

j)
```
      +
      +
      +
      +
  +++++++++
      +
      +
      +
      +
```

k)
```
*****
  ****
    ***
      **
        *
      **
    ***
  ****
*****
```

l)
```
0000000000
0000  0000
000    000
00      00
0        0
0        0
00      00
000    000
```

```
0000    0000
0000000000
```

6. Construya un programa en Python que escriba en un archivo de texto 'triangle.txt' un triángulo de Pascal con *n* filas, dónde el valor de *n* sea un valor entregado por el usuario. Por ejemplo, si la entrada fuese n = 10, entonces la salida debería ser:

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5  10  10   5   1
    1   6  15  20  15   6   1
  1   7  21  35  35  21   7   1
1   8  28  56  70  56  28   8   1
1  9  36  84 126 126  84  36   9   1
```

7. **(Ejercicio PEP)** Una compañía dedicada a la manufactura de celulares está desarrollando el sistema operativo de su nuevo teléfono el iXus 12, sin embargo, sus dispositivos están presentando problemas a la hora de conectarse a Wi-Fi.

   Para encontrar una solución al problema, se le ha pedido a usted que construya un programa que determine a qué red debería conectarse el teléfono, con el fin de verificar si el problema está en la conexión misma o en la elección de la red que el celular realiza.

   El celular tiene en su memoria dos archivos:

   - "`signal-strength.txt`" en el cual se registra el nombre de cada red Wi-Fi en el área y la potencia con la que el dispositivo recibe la señal (medida en dBm), separados por comas.
   - "`networks.txt`" en el que se registra el nombre de todas las redes en las que el dispositivo se ha conectado exitosamente en el pasado.

   A partir de ambos archivos, se requiere que usted genere un programa que escriba un archivo de texto plano llamado "`conn-priority.txt`" en el que se indique a qué redes podría conectarse el equipo, ordenadas desde la mejor a la peor opción, considerando que este sólo puede conectarse a redes del archivo "`networks.txt`".

   Vale la pena señalar que el nivel de señal de una red Wi-Fi se mide en decibel milliwatts y que normalmente los valores que las mediciones entregan siempre son negativas.

signal-strenght.txt
Git-gud,-32 Hogwarts Great hall Wifi,-67 God is my rock,-56 Interwebz,-75 OptimusPrime,-32 gg-wp,-53 UdeS-Alumnos,-90

networks
Que-sucede Get-off-my-lawn Git-gud gg-wp All the pancakes My-name-is-Lucifer Optimus-prime Hogwarts Great hall Wifi

En este caso, dado que existen 4 redes a las que es posible conectarse, el archivo "`conn-priority.txt`" debería ser:

conn-priority.txt
1,OptimusPrime,-32 2,Git-gud,-32 3,gg-wp,-53 4,Hogwarts Great hall Wifi,-67

Como se puede ver, en caso de que existan dos redes con la misma potencia, no importa cuál queda en primer lugar.

Considere que los archivos entregados corresponden a un ejemplo y que el programa debería funcionar para cualquier par de archivos que mantengan el mismo formato.

Para esta solución, no se permite usar métodos o funciones nativas o importadas para ordenar, u obtener máximos o mínimos.

8. **(EJERCICIO PEP)** Los traductores de textos clásicos chinos tienen disyuntivas respecto a las palabras que utilizan en sus traducciones, pues muchas veces creen que sus traducciones son originales, pero terminan traduciendo una gran cantidad de términos y oraciones de una misma forma. Para solucionar inicialmente este problema el traductor desea conocer las palabras de su traducción, que están en otra traducción, es por ello, que se le solicita a usted que a partir de dos archivos "`TraduccionPropia.txt`" y "`TraduccionDistinta.txt`", indique todas las palabras que sólo aparecen en el primer archivo, y todas las que son exclusivas del segundo.

   Considere que los archivos pueden tener mayúsculas y minúsculas, puntos y comas. El caso presentado a continuación es un ejemplo, sin embargo, su solución debiera funcionar para todo par de archivos.

TraduccionPropia.txt
The Tao that can be spoken is not the eternal Tao **The name that can be named is not the eternal name** **The nameless is the origin of Heaven and Earth** **The named is the mother of myriad things** **Thus, constantly free of desire** **One observes its wonders** **Constantly filled with desire** **One observes its manifestations** **These two emerge together but differ in name** **The unity is said to be the mystery** **Mystery of mysteries, the door to all wonders**

TraduccionDistinta.txt
Tao called Tao is not Tao. **Names can name no lasting name.** **Nameless, the origin of heaven and earth.** **Naming, the mother of ten thousand things.** **Empty of desire, perceive mystery.** **Filled with desire, perceive manifestations.** **These have the same source, but different names.** **Call them both Deep, Deep and again Deep, the gateway to all mystery.**

9. **(EJERCICIO PEP**) La biblioteca local está teniendo problemas para calcular las multas de aquellos que deben libros, por lo que se desea que usted construya un programa que automatice dicho proceso.

   Las reglas de multas de la biblioteca son las siguiente:

   - Si el usuario devuelve el libro en la fecha o antes de la fecha de entrega, no se aplica multa.
   - Si el usuario devuelve el libro después de la fecha de entrega, el mismo mes y año, la multa es de 30 pesos por día de atraso.
   - Si el usuario devuelve el libro después de la fecha de entrega, en un mes distinto, pero dentro del mismo año, la multa es de 300 pesos por mes de atraso.
   - Si el usuario devuelve el libro en un año distinto al que debía entregarlo, se le aplica directamente una multa de 10.000 pesos fija.

   Independientemente de cuánto tiempo haya pasado, siempre se aplica la multa de mayor valor, es decir, si una persona debía entregar un libro el 31/12/2017, si lo entrega el 02/01/2018 se aplica la multa por regla de año distinto

   Para registrar la información, la biblioteca tiene un archivo registro.txt en donde cada fila representa una solicitud de libro, en dónde se tiene el identificador del usuario, el identificador del libro pedido, la fecha de entrega fijada por la biblioteca y la fecha de entrega final.

registro.txt	multas.txt
JEPA22, ISBN3344-66744, 10/10/2018, 12/12/2018 CLUDIO, ISBN22554-2352, 10/10/2017, 31/01/2019 ROGER, ISBN312-9814767, 02/01/2019, 31/12/2018	CLUDIO, 10000 MAURI, 10000 JEPA22, 1320

JEPA22, ISBN412-445212, 01/12/2018, 25/12/2018 MAURI, ISBN367-8885343, 25/12/2018, 02/01/2019	

A partir de este archivo, se necesita que calcule las multas que registra cada usuario de la biblioteca y las entregue en un archivo "multas.txt" ordenadas de la multa de mayor valor a la de menor valor. En dicho archivo sólo deben aparecer los usuarios que registran multas. Para su solución considere también que los usuarios pueden haber pedido más libros y registrar multas por más de uno de ellos.

10. **(EJERCICIO PEP)** Julio es un andinista que lleva años planificando y entrenando para escalar las montañas más altas del continente, para planificar su expedición ha creado un archivo "`cumbres.txt`" con las montañas a las que le interesa ir, para poder ubicarlas rápidamente, registra su nombre, el país en el que se encuentra y la altura de cada montaña en una línea separada.

    Por otro lado, la esposa de Julio, preocupada por la seguridad de su esposo, ha revisado la dificultad de cada expedición y ha decidido vetar las que considera más peligrosas, para ello, le ha dado a Julio un listado de las montañas a las que no desea que vaya, en un archivo "vetadas.txt".

cumbres.txt
Aconcagua,Argentina,6962 Ojos del Salado, Chile,6891 Denali,USA,6190 Huascarán,Peru,6768 Mount Logan,Canada,5247 Pico de Orizaba,México,5636 Tres Cruces,6739,Argentina Llullaillaco,6739

vetadas.txt
Mount McKinley Everest K2 The Matterhorn Charro Chalten Mount Logan Llullaillaco Nanga Parbat Pico de Orizaba

Para poder llevar a cabo sus expediciones, Julio necesita un programa que ordene las montañas que su esposa no ha vetado, partiendo de la de mayor elevación a la más pequeña, a fin de poder priorizar las montañas a escalar, y las escriba en un archivo de texto "`cumbres-permitidas.txt`". Esto debido a que teme que, con los años, la lista de las montañas vetadas vaya aumentando.

En este caso, la esposa de Julio ha vetado 3 montañas de su lista, por lo que el archivo de salida debería presentarse con las 5 montañas permitidas, de la siguiente forma:

cumbres-permitidas.txt

```
1,Aconcagua,6962
2,Ojos del Salado,6891
3,Huascarán,6768
4,Tres Cruces,6739
5,Denali,6190
```

11. **(EJERCICIO PEP)** Las ciudades-estado griegas acostumbran a identificar a sus mayores figuras con su nombre solamente, pues aún no se inventaban los apellidos, sin embargo, debido a que los nombres se repiten de tanto en tanto, la solución para identificar a la persona es agregar, además del nombre, la ciudad a la que estos pertenecen. Por ejemplo: al destacado general Espartano Leónidas, se le conoce e identifica como “Leónidas de Esparta”.

    Un historiador de la Universidad de Atenas tiene la tarea de identificar cronológica y geográficamente a los personajes más connotados de la antigua Grecia y para ello tiene el trabajo de su antiguo colega, quién estaba levantando a la población de cada ciudad de Grecia.

    Hasta el minuto ha ordenado toda la información en un archivo “censo-griego.txt”, el que cuenta con varias líneas indicando la ciudad, el año de nacimiento aproximado, la persona identificada y su profesión más destacada.

    Sin embargo, sus jefes le han solicitado que ordene cronológicamente a los personajes según su profesión, del más antiguo al más moderno y que además identifique a la persona con el nombre de ciudad. Cómo los griegos fueron un pueblo lleno de personajes destacados, el tiempo para realizar este proceso manualmente no le alcanza, por lo que solicita a usted, que a partir del archivo “censo-griego.txt” y la profesión que se desea filtrar, genere un nuevo archivo con el nombre de la profesión en el que se indique el año de nacimiento y el nombre de la persona, con su ciudad de origen.

    Consideremos el siguiente ejemplo, en el que los jefes del historiador le solicitaron la lista de los personajes con la profesión de “filósofo”, para dicho caso, el funcionamiento del programa debería ser como se ilustra a continuación:

censo-griego.txt
Atenas,427 AC,Platón,Filósofo Seleucia,240 AC,Diógenes,Filósofo Citio,336 AC,Zenón,Filósofo Atenas,440 AC,Socrates,Filósofo Cirene,270 AC,Erastótenes,Astrónomo Esparta,488 AC,Leónidas,Rey Sinope,412 AC,Diógenes,Filósofo

filosofo.txt
440 AC Sócrates de Atenas 427 AC, Platón de Atenas 412 AC, Diógenes de Sinope 336 AC, Zenón de Citio 240 AC, Diógenes de Seleucia

Construya un programa en Python que realice el proceso de filtrado, ordenado y escriba el archivo de salida solicitado por el historiador, considere que el archivo "`censo-griego.txt`", es un ejemplo y que el programa debe funcionar para cualquier listado que cumpla el formato.

Para esta solución, no se permite usar métodos o funciones nativas o importadas para ordenar, u obtener máximos o mínimos.

12. **(EJERCICIO PEP)** El Sudoku es un juego matemático que se inventó en Japón a finales de 1970 y que se hizo mundialmente popular hace unos 15 años, cuando numerosos periódicos comenzaron a incluirlo en la sección de pasatiempos. El objetivo del Sudoku es rellenar una matriz de 9 x 9 celdas (81 casillas) dividida en sub-matrices de 3 x 3 llamadas "cajas" o "regiones" con cifras del 1 al 9 partiendo de algunos números dispuestos en algunas celdas, llamados pistas.

    Para rellenar exitosamente la matriz, se deben cumplir ciertas condiciones las cuáles son:

    - Un número no se puede repetir en una misma fila de la matriz.
    - Un número no se puede repetir en la misma columna de la matriz.
    - Un número no se puede repetir dentro de una "caja" o "región".

    A partir de las reglas dadas anteriormente, se solicita a usted la construcción de un programa destinado a validar si una solución entregada para un Sudoku cumple con las reglas anteriormente indicadas. Para ello, se le hará entrega de un archivo de texto "intento.txt" que contendrá 9 filas (separadas por un salto de línea) cada una con 9 números (separados por un espacio).

    El programa deberá leer el archivo e informar al usuario si el intento es válido o contiene errores. Para ilustrar el funcionamiento del programa, se entregan a continuación dos ejemplos, uno de una respuesta válida y otro de una respuesta con errores. Considere que estas entradas corresponden a ejemplos, y que su programa deberá funcionar para cualquier combinación de 81 números, siempre que mantengan el formato descrito en el párrafo anterior.

Intento válido								
5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Intento no válido								
7	3	5	6	1	4	8	9	2
8	4	2	9	7	3	5	6	1
9	6	1	2	8	5	3	7	4
2	8	6	3	4	9	1	5	7
4	1	3	**8**	5	7	9	2	6
5	7	9	1	2	**8**	4	3	8
1	5	7	4	9	2	6	8	3
6	9	4	7	3	**8**	2	1	5
3	2	8	2	**8**	6	1	7	9

Como se puede observar, el intento no válido, tiene dos veces el valor 8 en una misma región, el que a su vez, se repite en la sexta columna de la matriz, por lo que no sería una solución válida. Mientras que el intento válido, tiene todos los números solo una vez en cada fila, columna y región.

Un ejemplo del funcionamiento puede verse a continuación, en el que se entrega un intento que si bien, cumple la regla de las filas y de las columnas, no cumple la regla de las regiones:

intentos.txt
1 2 3 4 5 6 7 8 9 2 3 4 5 6 7 8 9 1 3 4 5 6 7 8 9 1 2 4 5 6 7 8 9 1 2 3 5 6 7 8 9 1 2 3 4 6 7 8 9 1 2 3 4 5 7 8 9 1 2 3 4 5 6 8 9 1 2 3 4 5 6 7 9 1 2 3 4 5 6 7 8

Respuesta
El sudoku ingresado es incorrecto.

13. **(EJERCICIO PEP)** Un profesor de lenguaje tiene un curso de estudiantes, frustrado con su falta de disciplina, ha decidido que realizará un control al inicio de la clase siguiente, cuando tenga menos de 50% de asistencia a tiempo.

    Para evitar que el profesor deba realizar el cálculo en cada ocasión se solicita a usted desarrollar un programa que indique si se debe realizar control o no a la clase siguiente. Para ello, él le proveerá la hora de inicio de la clase, y un archivo de texto “lista.txt” con la lista del curso, el archivo contiene en cada línea el nombre de cada estudiante y la hora en la que marcó su asistencia separado por coma. Un ejemplo de la ejecución esperada del programa, para una clase que comienza a las 8:00 puede verse a continuación

lista.txt
Roberto Martínez,08:05 **Juana Fuentes,07:59** **Felipe Padilla,07:58** **Alejandro Rice,07:51** **Andrés Cisterna,08:08** **Alfredo Memento,08:15** **Javiera Sepúlveda,08:00** **Cristián Salazar,07:55** **Jaime Jara,07:56** **Daniela Ramírez,07:45**

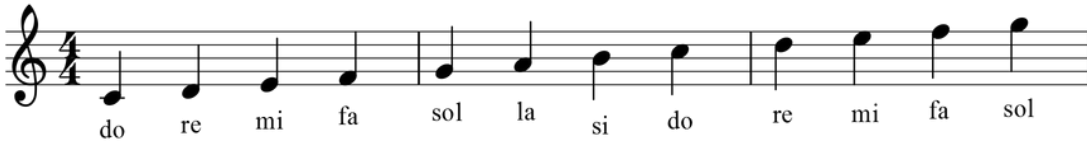
Salida
Tiene 70% de asistencia a tiempo, por lo tanto, la próxima clase NO HAY CONTROL

Considere que, si un estudiante llega a la hora exacta de clases, se considera a tiempo.

Recuerde que su solución debe funcionar para cualquier combinación de hora y archivo de texto, siempre y cuando se mantenga el formato, la solución no puede funcionar únicamente para el ejemplo dado.

14. **(EJERCICIO PEP)** Matías está aprendiendo a leer partituras, sin embargo, siente que aún es muy lento para ubicar las notas en el pentagrama. Con el objetivo de hacer esta tarea menos tediosa y aprender las canciones que su profesor le pide de forma más veloz, desea construir un script que le ayude a identificar las notas musicales según su ubicación en el pentagrama.

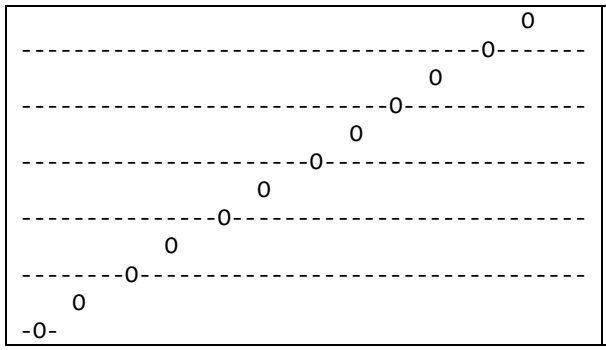
Él sabe que, dependiendo de su ubicación en el pentagrama, cada nota recibe un nombre, y para este trabajo, ha considerado sólo las siguientes:



Para facilitar la detección de estas notas, ha decidido crear un programa que traduzca las notas musicales a sus nombres en español y volcar dicha información en un archivo de salida, que podrá utilizar como apoyo cuándo lea la partitura para agilizar el proceso.

Como ejemplo, se presenta a usted el archivo de entrada “cancion.txt”, el cual representa la partitura de la figura anterior y la salida “salida.txt” respectiva que el programa debería generar en función de esta entrada.

cancion.txt	salida.txt

	DO RE MI FA SOL LA SI DO RE MI FA SOL

Para facilitar su desarrollo, considere que el archivo de entrada tiene siempre 12 líneas que representan cada una de las notas posibles, y que todas las líneas tienen el mismo largo, sin embargo, el orden y la cantidad de notas podrían alterarse.

Considere que Matías ha decidido utilizar el símbolo O para identificar las notas, y que su archivo no muestra compases, ni duraciones de las notas.

15. **(EJERCICIO PEP)** Las dinastías monárquicas y la iglesia católica acostumbran a identificar a sus máximas autoridades con su nombre, sin embargo, debido a que estas dinastías llevan siglos gobernando, los nombres se repiten, por lo tanto, la solución para no confundirlos es agregar, además del nombre, un número romano para identificarlos. Así, el primer rey de Francia en llamarse Luis, fue Luis I, el segundo Luis II, y así hasta Luis XVI cuyo reinado terminó con la llegada de la revolución francesa.

    En el caso del Papa de la Iglesia Católica, es lo mismo, sin embargo, ellos escogen su nombre al momento de asumir como Papas, así los últimos tres papas han sido Juan Pablo II, Benedicto XVI y Francisco I.

    Una antigua casa de Europa, los Wolvern, han tenido siglos de reyes, sin embargo, jamás se dieron el trabajo de numerar a sus gobernantes, por lo tanto, cuando ellos hablan del rey Roman, no saben si están hablando del primero, del segundo o del vigésimo cuarto.

    Un historiador se dio el trabajo de ordenar cronológicamente a los reyes de la dinastía Wolvern desde el primero al último y registró su trabajo en un archivo llamado “dinastia.txt”, sin embargo, con los siglos por los cuáles se extiende el reinado de esta casa, el historiador no confía en un proceso manual para ir numerando a los reyes, por lo tanto, desea un proceso automático que permita numerar a todos los reyes de la dinastía.

    Para ello, el historiador desea un programa que reciba como entrada el archivo con los reyes de la dinastía, y entregue como salida otro archivo con la dinastía numerada. Como el historiador

trabaja en esto, quiere que la solución sea general y funcione para cualquier archivo que respete el formato de entrada.

Un ejemplo del proceso solicitado, puede verse a continuación:

dinastia.txt
Julian
Roman
Julian
Marcus
Julian
Marcus
Roman
Pietrus
Vars
Roman

dinastiaNumerada.txt
Julian I
Roman I
Julian II
Marcus I
Julian III
Marcus II
Roman II
Pietrus I
Vars I
Roman III

Construya un programa en Python que realice el proceso de numeración solicitado por el historiador, considere que el archivo "dinastia.txt", es un ejemplo y que el programa debe funcionar para cualquier listado que cumpla el formato.

Considere que existe una función `convertirARomano(x)` en el módulo "numro" de Python, que permite la conversión de un número entero (x) a su representación en número romano (como string).