Guía de Ejercicios UVA9: Archivos

1.- La Policía de Investigaciones mantiene el registro de todas las personas que han ingresado al país en un archivo que contiene en cada línea un identificador único, el nombre de la persona, su país de procedencia y la fecha de ingreso a territorio nacional. La estructura de cada línea es la siguiente:

identificador:nombre**:**país**:**fecha.

La fecha se expresa en el formato día-mes-año. Por ejemplo:

inmigrantes.txt

```
456:Cristina Palermo:Argentina:07-05-2019
523:Sophia Watson:EEUU:02-05-2019
364:Alan Moore:EEUU:12-12-2018
111:Karen Lee:EEUU:05-09-2019
646:Nicol´as Cabrera:Venezuela:01-01-2019
...
```

La función **filtrar(nombre_archivo, año)** recibe dos parámetros: un string con el nombre de un archivo como el anterior, y un entero que especifica un año. Esta función retorna un diccionario cuyas llaves son los países de procedencia de los inmigrantes en el año ingresado como parámetro. El valor asociado a cada llave es una lista de strings con los nombres de las personas de ese país que ingresaron a territorio nacional el año indicado. Estudie el siguiente ejemplo para comprender lo que hace la función:

```
>>> print(filtrar('inmigrantes.txt', 2019))
{'Argentina': ['Cristina Palermo'],   'EEUU': ['Sophia Watson',
'Karen Lee'], 'Venezuela': ['Nicolás Cabrera'    ]}
```

Ordenamiento

A continuación se muestran las instrucciones que forman el código de la función filtrar, pero se encuentran desordenadas y sin indentación. Usted debe ordenarlas e indentarlas adecuadamente para que la función lleve a cabo el proceso descrito. Escriba la función completa a la derecha del código desordenado.

No debe agregar nuevas instrucciones y todas las instrucciones dadas deben ser utilizadas.

```
nombre = campos[1]
dicc[pais].append(nombre)
if int(fecha[2]) == a˜no:
dicc[pais] = []
archivo.close()
for linea in archivo:
pais = campos[2]
if pais not in dicc:
archivo = open(nombre_archivo, 'r')
dicc = {}
return dicc
fecha = campos[3].split('-')
def filtrar(nombre_archivo, a˜no):
campos = linea.strip().split(':')
```

2.- NOTA: Puede utilizar la función filtrar de la Pregunta 1. No es necesario que la copie de nuevo.

Siguiendo el contexto anterior, escriba la función **contar_ingresos(nombre_archivo, año)** que recibe dos parámetros: un string con el nombre de un archivo con el formato indicado, y un entero que especifica un año. La función debe retornar un diccionario cuyas llaves sean los países de procedencia de los inmigrantes en el año señalado como parámetro. El valor de cada llave debe ser la cantidad de personas de ese país que ingresó a territorio nacional el año indicado.

Ejemplo 1:

```
>>> print(contar_ingresos('inmigrantes.txt', 2019))
{'Argentina': 1, 'EEUU': 2, 'Venezuela': 1}
```

Ejempo2:

```
>>> print(contar_ingresos('inmigrantes.txt', 2018))
{'EEUU': 1}
```

3.- NOTA: Puede utilizar las funciones de las preguntas anteriores. No es necesario que las copie de nuevo.

Siguiendo el contexto anterior, escriba la función **escribir_ingresos(archivo, año)** que recibe dos parámetros: un string con el nombre de un archivo como el descrito, y un entero que especifica un año. La función debe crear un archivo de nombre **ingresosX.txt**, donde X corresponde al año ingresado como parámetro. El archivo debe contener los países de procedencia de los inmigrantes en el año especificado en el parámetro, ordenados de mayor a menor según cantidad de personas de ese país que ingresaron a territorio nacional ese año. Debe incluir en cada línea su número correlativo partiendo desde el 1. Además, la función debe retornar el número total de personas que ingresaron al país ese año. Guíese por los ejemplos.

Ejemplo 1:

```
>>> print(escribir_ingresos('inmigrantes.txt', 2019))
4
```

ingresos2019.txt

```
1-EEUU:2
2-Venezuela:1
3-Argentina:1
```

Ejemplo 2:

```
>> escribir_ingresos('inmigrantes.txt', 2018)
1
```

ingresos2018.txt

```
1-EEUU:1
```

4.- Cesar Pyllan, el encantador de perros, mantiene el registro de las mascotas que ha atendido en un archivo cuya estructura es **`fecha;nombre;raza;problema;solucion`**. Un ejemplo de este archivo sería:

perros.txt

```
01-01-2015;boby;salchicha;muerde;collar de ahorque
02-03-2016;billy;pastor aleman;miedo;toque al costado
10-12-2016;boby;san bernardo;muerde;toque al costado
24-01-2017;willson;salchicha;miedo;toque al costado
04-02-2017;willson;san bernardo;miedo;toque al costado
03-02-2018;firulais;labrador;estres;SHHH
03-03-2018;firulais;pequines;estres;SHHH
…
```

La función **leer_perros(na)** se comporta de la siguiente manera:

```
>>> print(leer_perros("perros.txt"))
{'boby': ['collar de ahorque', 'toque al costado'], 'billy': ['toque al
costado'],
'willson': ['toque al costado', 'toque al costado'], 'firulais': ['SHHH',
'SHHH']}
```

El código de la función se encuentra desordenado. Usted debe ordenar las líneas e identarlas correctamente.

```
_,n,r,p,s = l.strip().split(';')
pe[n].append(s)
return pe
a = open(na)
a.close()
def leer_perros(na):
pe[n] = []
for l in a:
if n not in pe:
pe = {}
```

Finalmente, explique en menos de 30 palabras lo que realiza la funcion anterior.

5.- NOTA: Considere el contexto de la pregunta anterior. Puede utilizar la función leer_perros(na) de esa pregunta.

.

En esta oportunidad César Pyllan le solicita su ayuda para lo siguiente:

a.- Implemente la función **leer_razas(nombre_archivo)**, que recibe un string con el nombre de un archivo y clasifica los animales por raza. La función retorna un diccionario cuya llave es el nombre de la raza y el valor una lista con los nombres de los perros que pertenecen a ella. Guíese por el ejemplo a continuación.

Ejemplo

```
>>> print(leer_razas('perros.txt'))
```

```
{'salchicha': ['boby', 'willson'], 'pastor aleman': ['billy'],
'san bernardo': ['boby', 'willson'], 'labrador': ['firulais'],
'pequines': ['firulais']}
```

b.- César ha notado que existen perros con el mismo nombre en distintas razas. Sabiendo que los nombres de los perros son únicos, esto sólo puede significar que existen mestizos. Desarrolle una función llamada **mestizos(raza1,raza2, nombre_archivo)** que recibe dos razas de perro y el nombre del archivo.
Esta función debe retornar una lista con los nombres de los perros mestizos de dichas razas. Si no existen perros mestizos para las razas consultadas, la función debe retornar una lista vacía. Guíese por los ejemplos a continuación.

Ejemplos

```
>>> print(mestizos('salchicha','san bernardo','perros.txt'))
['boby', 'willson']
>>> print(mestizos('pequines','labrador','perros.txt'))
['firulais']
>>> print(mestizos('pequines','pastor aleman','perros.txt'))
[]
```

6.- La línea aérea LANX desea premiar a sus pasajeros frecuentes incrementando sus kilómetros LANX-Pass de acuerdo al kilometraje acumulado a la fecha. para ello, cuenta con el archivo pasajeros.txt con campos separados por ":" con la siguiente estructura: rut:nombre:kilometraje, tal como se muestra en el siguiente ejemplo:

```
10043499-7:Javier Diaz:553450
7655511-7:Sebastian Rojas:134987
17273221-4:Elizabeth Bugueno:342021
18653834-4:Benjamin Bueno:2000
```

Considere que el archivo anterior puede tener muchos datos. Lo anterior es sólo un ejemplo

a. Desarrolle la función crear_lista(archivo) que reciba como parámetro el nombre del archivo y retorne una lista de lista con los datos de cada pasajero. El campo kilometraje **debe transformarse a entero.**

   >>>crear_lista("pasajeros.txt")}
   [
   ['10043499-7', 'Javier Diaz', 553450], ['7655511-7', 'Sebastian Rojas', 134987], ['17273221-4', 'Elizabeth Bugueno', 342021] ['18653834-4', 'Benjamin Bueno', 2000]
   ]

b. Desarrolle la función kilometraje(lista) que reciba como parámetro la lista creada anteriormente, e incremente en ella el kilometraje de cada pasajero de acuerdo a la siguiente tabla. La función retorna la lista actualizada.

KM Actual	KM a incrementar
0-15.000	3.000
15.001 - 50.000	6.000
50.001 - 150.000	10.000
más de 150.000	40.000

>>>kilometraje(lista)
[['10043499-7', 'Javier Diaz', 593450], ['7655511-7', 'Sebastian Rojas', 144987], ['17273221-4', 'Elizabeth Bugueno', 382021], ['18653834-4', 'Benjamin Bueno', 5000]]

c. Desarrolle la función **escribir_archivo(archivo,lista)**, la cual recibe como parámetros el nombre de un archivo y la lista actualizada. La función debe guardar la información contenida en la lista, en un archivo con el mismo formato que el archivo original, es decir: rut:nombre:kilometraje.

>>>escribir_archivo("salida.txt", lista)

```
10043499-7:Javier Diaz:593450
7655511-7:Sebastian Rojas:144987
17273221-4:Elizabeth Bugueno:382021
18653834-4:Benjamin Bueno:5000
```

7.- [2da o tercera clase] En el pueblito de pelotillehue se desarrollarán las elecciones municipales, para la cual existen dos candidatos doña Tremebunda opción A y Condorito opción B.
Se sabe que el pueblito de Pelotillehue está segmentado en tres sectores: Norte, Centro y Sur. Para estas elecciones el actual alcalde Eugenio, ha querido dejar el legado de un sistema de verificación y conteo de las votaciones, para lo cual se elaborará un archivo llamado votaciones.txt el cual posee la estructura que muestra en la figura.

Votaciones.txt

```
A
Sur
B
Norte
B
Norte
A
Centro
…
```

Condorito.txt

```
B
Sur
B
Norte
B
Norte
B
Centro
…
```

Tremebunda.txt

```
A
Sur
A
Norte
A
Norte
A
Centro
…
```

Eugenio sabe que usted está tomando la asignatura de programación y que tiene el dominio necesario para desarrollar aplicaciones que trabajen con archivos, para lo cual le solicita lo siguiente:

•Desarrolle un programa que lea el archivo votaciones.txt, el cual está lleno con el tipo de información como la desplegada en la figura, y cree dos archivos llamados Condorito.txt y Tremebunda.txt los cuales deben contener solo los votos pertenecientes a cada candidato además del sector. (ver ejemplo en la figura arriba).
• Cree el Archivo Resultados.txt, el cual debe dar el resultado de las elecciones por sector.

```
Sector sur
Doña Tremebunda 32 votos
Condorito 40 votos

Sector Centro
Doña Tremebunda 40 votos
Condorito 30 votos

Sector Norte
Doña Tremebunda 40 votos
Condorito 30 votos
```