Actividad 8 - Datos y cumpleaños

Pensamiento Computacional

Segundo Cuatrimestre 2024

Pasos para resolver los ejercicios

1. **Entender qué hay que hacer**
   a) **Leer en detalle** el texto de la descripción de la función e interpretar qué se espera (por ej., qué parámetros tiene, qué hace con la entrada y cómo genera la salida).
   b) Escribir en el cuaderno distintos ejemplos de **entradas** y qué **salidas** se esperan para cada una.
2. **Armar el esqueleto de la función**
   a) Escribir exactamente como se pidió el **encabezado de la función**, incluyendo los parámetros.
   b) Definir la **variable de la salida** y ponerle un valor inicial válido (por ej., 0 para un contador)
   c) **Agregar el** `return` donde se devuelve la variable anterior.
   d) Probar **ejecutar la función** con diferentes valores, aunque todavía no hace lo que se pide.
3. **Implementar la función**
   a) **Escribir de a poco** el contenido de la función.
   b) **Usar otras funciones** para resolver el problema, ya sea que estén de antes (y las reutilizan), o que crean otras nuevas (porque nos conviene partir el problema en otros más pequeños y manejables).
4. **Probar la función**
   a) **Probar la función solamente por la consola**, usando distintos valores para los parámetros, y ver que funciona bien. No hacer las pruebas desde el editor de texto de Spyder ni usar prints.
   b) **Ver qué pasa con casos especiales**, como cuando le dan una lista vacía u otro caso donde se usa una entrada válida, pero que pone a prueba los límites de lo que se hizo.
5. **Preguntar al equipo docente** por cualquier duda o consulta.

La paradoja del cumpleaños

El objetivo de esta actividad es analizar y simular datos para estudiar la paradoja de los cumpleaños.

Algunas funciones útiles

1. Implementar una función llamada `leer_csv(ruta)` que reciba la `ruta` de un archivo y devuelva una lista de listas, donde cada lista interna representa una línea del archivo. Cada lista debe tener tantos elementos como columnas en el archivo .csv.

Ejemplo:

El archivo `prueba.csv` tiene el siguiente contenido:

```
Nombre,Posicion,Goles
Lionel Messi,Delantero,30
Cristiano Ronaldo,Delantero,25
```

La función debe devolver:

```
[['Nombre','Posicion','Goles'],
['Lionel Messi','Delantero','30'],
['Cristiano Ronaldo','Delantero','25']]
```

Ayuda: Una forma de abrir un archivo y leer su contenido es la siguiente:

```
with open('archivo.csv', 'r', encoding='utf-8') as archivo:
    # Acá podemos trabajar con el contenido del archivo
```

`archivo.readlines()` lee todas las líneas del archivo y las devuelve como una lista de strings, donde cada string es una línea del archivo.

`string.split(',')` recibe un string y devuelve una lista de elementos, utilizando la coma (,) como separador.

2. Implementar una función llamada `lista_de_listas_a_dicc(lista_de_listas)` que reciba una `lista de listas`. La función debe devolver una lista de diccionarios, donde cada diccionario contiene los datos de cada jugador.
   Notar que la primera lista contiene las claves de los diccionarios, mientras que el resto de las listas contienen los valores que corresponden a esas claves para cada jugador.

Ejemplo:

Si a la función le pasamos la siguiente lista de listas:
```
[['Nombre','Posicion','Goles'],
['Lionel Messi','Delantero','30'],
['Cristiano Ronaldo','Delantero','25']]
```

La función debe devolver:
```
[ {'Nombre': 'Lionel Messi', 'Posicion': 'Delantero', 'Goles': '30'}, {'Nombre': '
Cristiano Ronaldo', 'Posicion': 'Delantero', 'Goles': '25'}]
```

3. Implementar la función `coincidencia(lista)`, que toma una `lista` y devuelve `True` si contiene al menos dos elementos iguales entre sí, o `False` en caso contrario.

   Ayuda: Usar if `elemento` in `lista`.

Para probar en consola:

Probar la función con estas listas: ´
`[1,2,3,4,5,6]` y `[1,2,3,4,5,1]`
`["27-jun", "15-ago", "13-nov"]` y `["10-jun", "13-nov", "10-jun"]`

4. Implementar la función `sin_repetidos(lista)`, que toma una `lista` y devuelve otra lista que contiene los mismos elementos, pero sacando los repetidos.

   Ayuda: Si les sirve, usar if `elemento` in `lista`.

Para probar en consola:

Probar la función con estas listas:
```
sin_repetidos([5,1,0,1,2,3,5,7,1,1,6]) # Devuelve [5,1,0,2,3,7,6]
sin_repetidos([1,9,6,3]) # Devuelve [1,9,6,3]
sin_repetidos([]) # Devuelve []
```

5. Implementar `filtrar_filas(filas, columna, valor)`, que recibe una lista de elementos de tipo diccionario (`filas`), el nombre de una `columna` y un `valor`. La función debe devolver una lista con aquellos diccionarios donde la columna indicada toma el valor especificado.

Para probar en consola:

Para esta lista de diccionarios:
```
mis_filas = [{"Edad": 26, "Altura_cm": 180.0}, {"Edad": 26, "Altura_cm": 170.0}, {"
Edad": 30, "Altura_cm": 175.0}, {"Edad": 41, "Altura_cm": 180.0} ]
```

Probar la función `filtrar_filas` con estos parámetros:
```
filtrar_filas(mis_filas, "Edad", 26) # Devuelve una lista con los dos primeros
filtrar_filas(mis_filas, "Altura_cm", 180.0) # Devuelve una lista el 1ero y el 4to
filtrar_filas(mis_filas, "Edad", 25) # Que deberia ocurrir?
```

6. Implementar la función `convertir_en_lista(filas, columna)`, que toma una lista de diccionarios (`filas`) y el nombre de una `columna` y devuelve la lista que resulta de tomar los valores de la columna a lo largo de todas las filas.

Para probar en consola:

Para esta lista de diccionarios:

```
mis_filas = [{"Edad": 26, "Altura_cm": 180.0}, {"Edad": 26, "Altura_cm": 170.0}, {"
Edad": 30, "Altura_cm": 175.0}, {"Edad": 41, "Altura_cm": 180.0}]

convertir_en_lista(mis_filas, "Edad") # Devuelve [26,26,30,41]
convertir_en_lista(mis_filas, "Altura_cm") # Devuelve [180.0,170.0,175.0,180.0]
```

Trabajo con los datos

Ahora pasamos a usar los datos de los jugadores que estan en el campus. Tene en cuenta que para esta sección vas a necesitar algunas de las funciones que hiciste antes, y que el parámetro **filas** hace referencia a una lista de diccionarios que podes obtener a partir de `lista_de_listas_a_dicc(lista_de_listas)`.

7. Implementar la función `hay_coincidencia_en_equipo(filas, equipo)`, que dados los datos de los jugadores de los equipo, que es la lista de elementos de tipo diccionario (`filas`), y el nombre de un `equipo`, devuelve True si hay alguna coincidencia en los cumpleaños del equipo indicado, y False si esto no ocurre.

   Ayuda: Recordar que nuestros datos tienen una columna `"dia_mes"`, donde está el día y mes del cumpleaños de cada jugador (por ejemplo, "23-feb").

Para probar en consola:

Para nuestros datos anteriores:

```
hay_coincidencia_en_equipo(filas, "Boca Juniors") # Debe devolver True
hay_coincidencia_en_equipo(filas, "Huracan") # Debe devolver False
hay_coincidencia_en_equipo(filas, "River Plate") # Debe devolver True
hay_coincidencia_en_equipo(filas, "San Lorenzo") # Debe devolver False
```

8. Implementar la función `todos_los_equipos(filas)`, que dada una lista de elementos de tipo diccionario (`filas`) devuelve una lista con los equipos de los cuales tenemos datos, sin repetidos. La columna donde está el nombre de cada equipo se llama "Equipo".

   Ayuda: Recordar que tenemos las funciones `convertir_en_lista` y `sin_repetidos`.

9. Implementar la función `porc_equipos_con_coincidencia(filas, equipos)`, que dados una lista de elementos de tipo diccionario (`filas`) y una lista de nombres de equipos, devuelve el porcentaje de los equipos indicados donde hay jugadores con cumpleaños repetidos.

Para probar en consola:

Para nuestros datos anteriores:

```
algunos_equipos = ["Boca Juniors", "Huracan", "River Plate", "San Lorenzo"]
porc_equipos_con_coincidencia(filas, algunos_equipos) # Devuelve 0.5
```

   Con las anteriores funciones obtener el porcentaje del total de equipos que tienen coincidencias de cumpleaños.

   ¿Da como esperabamos según la paradoja del cumpleaños?

Simulación

En nuestros datos hay sólo 28 equipos, lo que es relativamente poco. ¿Que podemos hacer si queremos tener más? Para esta parte de la actividad vamos a pasar a modelar las fechas de cumpleaños como números del 1 al 365. Además, vamos a asumir que la probabilidad de nacimiento es la misma para todos los días del año, lo que nos va a permitir usar `random.randint(1, 365)` para simular fechas.

10. Implementar la función `fechas_cumples(n)`, que dado el tamaño del grupo de personas a simular (`n`), devuelve una lista con las n fechas de cumpleaños generadas aleatoriamente.

Para probar en consola:

Llamar a `fechas_cumples` varias veces con `n=7`, `n=23` y `n=100`.
¿Hay coincidencias dentro de alguno de los grupos simulados?

11. Implementar la función `chances_coincidencia(n, n_rep)`, que toma el tamaño de grupo `n` y la cantidad de grupos a ser simulados `n_rep`, y devuelve el cociente de casos favorables (es decir casos en los que hubo coincidencia de cumpleaños) sobre cantidad de repeticiones.

    Recordatorio: Si hacemos **muchas** repeticiones, la chance estimada se acerca a la probabilidad teórica.

Para probar en consola:

```
chances_coincidencia(7, 1000) #Da alrededor de 0.06
chances_coincidencia(15, 1000) #Da alrededor de 0.25
chances_coincidencia(30, 1000) #Da alrededor de 0.71
```

12. Estimar las chances de coincidencia para todos los `n` entre `7` y `77`, con `n_rep = 10000`. Graficar las chances estimadas en función de `n` usando `matplotlib`. ¿Qué pasa a medida aumenta `n`? ¿Y cerca de `n = 77`?
13. Usando la lista de probabilidades estimadas en el punto anterior, responder: ¿cuantas personas hacen falta como mínimo para que la probabilidad de que haya alguna coincidencia de cumples entre ellas sea mayor a 0,5?
14. Volvamos a nuestros datos anteriores, y graficar la cantidad de nacimientos por mes usando `matplotlib`. ¿Te parece que cualquier día del año tiene la misma probabilidad de tener nacimientos? ¿Cómo podríamos modificar lo anterior para obtener una distribución de cumpleaños más cercana a la real?

    Ayuda: Recordar que nuestros datos tienen una columna `"nro_mes"`, que tienen el número del mes del cumpleaños de cada jugador.

Tarea

Usando lo que hiciste en la primer parte de la guía (*"Trabajo con los datos"*), respondé:

15. ¿Cuál es el jugador más viejo de Velez?
16. ¿Cuál es el jugador más petiso de cada equipo?
17. ¿Cuál es el promedio de edad de los delanteros de Talleres de Córdoba?

Hacé las funciones que creas necesarias.

Ayuda: En los datos, los valores numéricos están como strings. La función `int('25')` devuelve 25 (el entero correspondiente).