Manipulando grandes cantidades de datos

Prof. Fabiola Uribe Plata

Julio Arath Rosles Olidén A01630738

Oscar Miranda Escalante A01630791

Fundamentos de programación

Tecnológico de Monterrey

21 de noviembre de 2018

El objetivo de este programa es buscar, manipular y reportar información de un archivo que contiene miles de datos de hospitales de Estados Unidos obtenido de *Centers for Medicare & Medicare Services (CMS)*. El programa interactúa con el usuario a través de un menú para saber qué información buscar y cómo manipularla. Se busca realizar este proyecto aplicando distintas estructuras y librerías en Python como funciones, condicionales, ciclos, listas, datetime, entre otras.

Descripción del diseño

A) Librería `datetime`: se importa al inicio del programa para poder incluir la fecha de generación de archivos en los reportes.

B) Archivos para lectura:
   - pvc.csv: este es el archivo descargado desde CMS. Por cuestriones prácticas, se le puso este nombre corto.
   - states.csv: este archivo contiene las abreviaturas de cada uno de los 50 estados en E.U. con sus respectivos nombres para hacer conversiones entre nombre y abreviatura.

C) Funciones auxiliares:
   - `make_list(linea)`: esta función es la más utilizada en el programa. Se utiliza para convertir una linea de datos separados por comas de cualquier archivo en una lista.
   - `short(estado)`: recibe un nombre completo de estado y devuelve su abreviatura.
   - `long(estado)`: recibe una abreviatura de estado y devuelve el nombre completo del estado.
   - `traduce_afeccion(afeccion)`: recibe una de las cuatro afecciones disponibles (ataque/falla/cadera/neumonía) y devuelve su nombre como aparece en el archivo pvc.csv para poder hacer búsquedas relacionadas.

D) Funciones principales:

   Cada una de estas funciones inica con una estructura de validación para abrir por lo menos un archivo (`try-except-else`). Al terminar de usar el archivo, se cierra utilizando la función `.close()` para hacer más eficiente el programa.

- `hospitales_estado(estado)`: esta función recibe el nombre de un estado para generar un archivo csv con los datos de todos los hospitales en el estado. Incluye una validación adicional al momento de crear el archivo "archivo_output_hospitales". En el momento en que se crea, se incluye la el nombre del reporte en la primera línea, la fecha (llamando a `datetime`) en la segunda línea y etiquetas para dos columnas en la tercera línea. Para buscar información en pvc.csv utiliza una estructura `for` que va reescribiendo una lista de cada línea del archivo pvc.csv. Hace comparaciones entre columnas mediante la estructura `if` para encontrar la información correcta y, finalmente, la añade al archivo que generó en un principio con la función `.write()`.
- `lowest_high(state, afeccion)`: esta función utiliza el parámetro `state` de la misma manera que `hospitales_estado(estado)` y también reescribe una lista por cada línea en el archivo pvc.csv. Además, debe filtrar más la información de acuerdo al parámetro `afeccion`, para lo cual incluye otra estructura if que compara datos entre columnas. Como algunos hospitales no cuentan con esta información, es importante agregar otro filtro para que la función ignore aquellas líneas donde el dato es igual a "Not available". Finalmente, la función debe decidir si el dato del pago es mayor que el anterior (haciendo uso del acumulador `high_estimate`) y registrar el nombre del hospital correspondiente en la variable `hospital`, misma que devuelve al final del ciclo `for`.
- `menor_que_promedio(city, state, affection)`: inicia convirtiendo el nombre del estado (parámetro `state`) por su abreviatura y traduciendo la afección (parámetro `affection`) a su su versión en el archivo pvc.csv. La información se filtra de acuerdo a `city`, `state`, `affection` y, si el indicador en la columna 10 indica "Less than the National Average Payment", entonces se le suma 1 al contador `num_hospitales`. El dato que devuelve la función al final es el valor del contador.

- `mayor_que_promedio(afeccion)`: recibe una afeccion para generar un archivo con el porcentaje de hospitales de cada estado que atienden esta afección y que indican “Greater than the National Average Payment”. Fue necesario definir una lista `us_states` con las abreviaturas de cada estado para poder ir generando el porcentaje que le corresponde a cada uno. Mediante una estructura `for` con en el rango (0, 50) (porque son 50 estados), se filtró la información para cada estado dentro de `us_states` calculando los porcentajes con el uso de los contadores `total_hospitales` y `num_hospitales`. Es importante validar que el contador `total_hospitales` no sea igual a 0 para evitar una división entre cero. Para evitar tener que abrir y cerrar el archivo por cada estado, se utilizó la función `.seek(0)`, la cual mueve el puntero del archivo a la posición cero. De esta forma, se puede volver a leer el archivo completo.
- `promedio_costo_CP(cp, afeccion)`: recibe del usuario el código postal (`cp`) y la afección (`afeccion`). Mediante una estructura `for`, el contador `cantidad` y el acumulador `suma_payments` calcula el promedio de pago para la afección, siempre y cuando `cantidad` no sea igual a 0 y el dato sea diferente de “Not available”.

E) Aspectos importantes del menú:

- Inicia con la explicación del programa y la bienvenida.
- Utiliza una estructura while para que el programa termine hasta que el usuario lo desee.
- La parte else valida si la opción seleccionada es válida.
- Notamos que el proceso de la función 4 es mucho más tardado que cualquier otro, por lo que decidimos incluir un print “procesando…”

Problemas enfrentados durante el desarrollo del programa:

- Tuvimos que cambiar los nombres de las variables para que las partes que se trabajaron por separado fueran compatibles.

- En un principio, la función 4 devolvía únicamente la información de alabama, por lo cual descubrimos que el programa no estaba leyendo el archivo pvc.csv desde el principio para cada estado. Para solucionarlo, investigamos y encontramos la función .seek().
- Fue necesario hacer modificaciones al archivo pvc.csv con ayuda de Excel para poder generar las listas de manera correcta.

Conclusión

Logramos escribir un programa funcional relacionado con el manejo de datos en grandes cantidades, lo cual no ayudó a comprender muy a fondo las aplicaciones de Python y también a integrar todos los conocimientos adquiridos durante el curso de Fundamentso de programación.