N3 – PROY: Restaurantes

IMPORTANTE: Este proyecto debe realizarse de forma **completamente individual**.

Objetivo general

Practicar los conceptos clave estudiados en el Nivel 3 del curso.

Objetivos específicos

1. Practicar la lectura de archivos en formato *CSV* (*Comma-Separated Values*).
2. Ejercitar la implementación de algoritmos de manipulación de listas.
3. Practicar la manipulación de estructuras de datos compuestas.
4. Practicar la técnica "Dividir y Conquistar".

Actividad 1 | Entendimiento del problema

Factores como la cercanía, el rating y el precio, entre otros, influyen significativamente en cómo los consumidores eligen restaurantes. En este proyecto, usted creará una aplicación para que los clientes puedan recibir recomendaciones y encontrar restaurantes que se ajusten a sus preferencias.

Cálculo de la distancia cartesiana

La *distancia cartesiana* se refiere a la distancia mínima entre dos puntos en un plano. Los puntos P_1 y P_2 se definen por sus coordenadas geográficas: latitud (lat_1, lat_2) y longitud (lon_1, lon_2). Entonces, para dos puntos $P_1 = (lat_1, lon_1)$ y $P_2 = (lat_2, lon_2)$ la distancia cartesiana (o mínima) entre estos será:

$$d = \sqrt{(lat_2 - lat_1)^2 + (lon_2 - lon_1)^2} \qquad F1$$

Descripción de la aplicación

En este proyecto usted manipulará información de restaurantes de los Estados Unidos. Su aplicación debe cargar la información de los restaurantes a partir de un archivo *CSV* (`restaurantes.csv`) que tiene 11 columnas, como se explica en la Tabla 1:

Columna	Descripción	Tipo	Ejemplo
name	Nombre del restaurante.	str	Subway
category	Categoría del restaurante.	str	Sandwich shop
rating	Calificación de clientes al restaurante. El rango del *rating* está entre 1.0 y 5.0.	float	4.2
latitude	Coordenada de latitud del restaurante.	float	37.3549757
longitude	Coordenada de longitud del restaurante.	float	-85.3264818
city	Ciudad en donde se ubica el restaurante.	str	Campbellsville
state	Estado en donde se ubica el restaurante.	str	Kentucky
price	Indicador de precio del restaurante. A más símbolos de "$" más costoso. Las posibilidades son: "$", "$$", "$$$", "$$$$".	str	$$$
review_count	Número de *reviews* dadas por usuarios al restaurante.	int	8951
delivery	Indicador de domicilio. 1 si tiene el servicio de domicilios y 0 si no tiene.	int	1
address	Dirección del restaurante.	str	1602 E Broadway St

Tabla 1. Descripción de las columnas del archivo restaurantes.csv

A continuación, se muestra un fragmento del archivo restaurantes.csv. La primera línea contiene el nombre de las columnas. Posteriormente, cada línea describe un restaurante y sus datos relevantes:

```
name,category,rating,latitude,longitude,city,state,price,review_count,delivery,address
Subway,Sandwich shop,4.2,37.3549757,-85.3264818,Campbellsville,Kentucky,$$$,8951,1,1602 E Broadway St
Taco Bell,Fast food,3.4,34.179879,-93.0712665,Arkadelphia,Arkansas,$$$,6962,1,121 Valley Rd
Pizza Hut,Pizza delivery,4.3,38.3720139,-97.6241374,McPherson,Kansas,$$$$,4881,0,2215 E Kansas Ave
IHOP,Breakfast,3.1,41.2505211,-75.8430915,Springfield,Massachusetts,$,616,0,770 Kidder St
IHOP,Breakfast,3.4,35.7326496,-84.3912019,Lowell,Massachusetts,$$$$,5200,0,12502 TN-72
KFC,Chicken,3.7,35.2714408,-77.6034355,Kinston,North Carolina,$$$,9448,1,1613 W Vernon Ave
Papa Johns Pizza,Pizza,3.2,34.7966715,-92.2555786,North Little Rock,Arkansas,$$,2587,1,4612 JFK Blvd Ste 6
Dunkin,Coffee shop,3.7,42.5854222,-71.777832,Fitchburg,Massachusetts,$$$$,6100,1,580 John Fitch Hwy
Los Pollos Hermanos,Chicken,4.8,35.0145774,-106.6875825,Albuquerque,New Mexico,$,9689,1,699 Susan Case ave
Juan Valdez Café,Coffee shop,4.5,25.81239,-80.33069,Doral,Florida,$$$,7205,1,37 Hunter Spencer Ct
Burger King,Fast food,3.1,46.829359,-100.8716328,Mandan,North Dakota,$$,6494,0,1400 E Main St
Dunkin,Coffee shop,3.6,42.4976361,-71.1257662,Woburn,Massachusetts,$$$,1469,1,344 Washington St
Aziza,Moroccan,4.6,37.7804334,-122.481706,San Francisco,California,$$$,5068,1,5800 Geary Blvd
```

CSV Ejemplo. Fragmento del contenido del archivo restaurantes.csv[1]

La aplicación debe construir una estructura compuesta basada en un ***diccionario de estados*** (el diccionario principal), donde las llaves son los estados y los valores son listas de diccionarios. Cada lista puede contener uno o varios diccionarios que almacenan la información de cada restaurante, llamados ***diccionario de restaurantes.***

[1] Basado en el dataset: Kwxdata. (2023, October 1). 380,000 restaurants mostly USA based. Kaggle.

A continuación, se muestra una imagen que ejemplifica la estructura deseada. El único diccionario de restaurante mostrado pertenece a la primera línea del CSV Ejemplo.

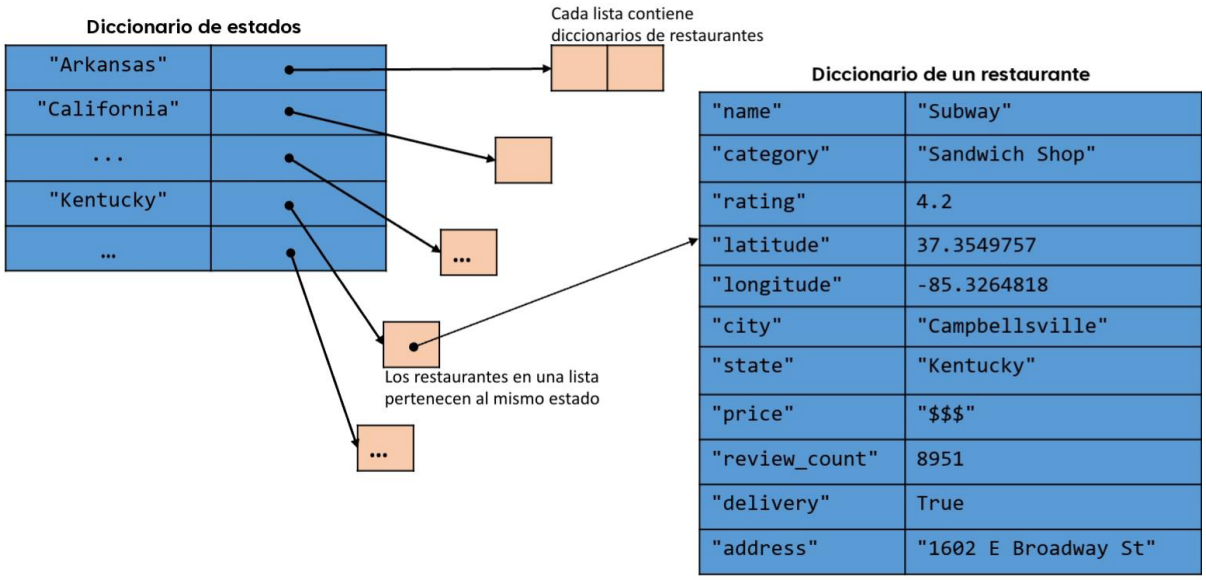

Figura 1. Ejemplo de la estructura deseada para el manejo de los datos. La llave `"delivery"` se debe cargar como del tipo `bool` (y no como `int`), como se explica más adelante en la descripción de la Función 1.

La aplicación debe permitir al usuario ejecutar las siguientes acciones (cuyas funciones asociadas se describen más adelante):

1. Cargar un archivo con la información de los restaurantes.
2. Buscar todos los restaurantes en un área seleccionada.
3. Buscar el restaurante con más sucursales en un estado. Una *sucursal* es una sede del restaurante que tiene el mismo nombre, pero diferente ubicación.
4. Estandarizar las direcciones de los restaurantes.
5. Buscar todos los restaurantes cuyo nombre sea palíndromo. Un nombre es *palíndromo* si se lee igual de izquierda a derecha que de derecha a izquierda, ignorando espacios y mayúsculas.
6. Buscar el restaurante más cercano a un punto de referencia.
7. Buscar al primer restaurante que cumpla con varias preferencias del usuario.

Actividad 2 | Preparación del ambiente de trabajo

1. Cree una carpeta para trabajar, poniéndole su nombre o *login*.
2. Descargue de Bloque Neón el archivo con el "esqueleto" del proyecto (`n3-esqueleto.zip`) y descomprímalo en su carpeta de trabajo. El esqueleto incluye dos archivos (`consola_restaurantes.py` y `restaurantes.csv`).
3. Abra Spyder y cambie la carpeta de trabajo para que sea la carpeta con el esqueleto.

Actividad 3 | Construir el módulo de la lógica

Usando Spyder, cree en su carpeta de trabajo un nuevo archivo con el nombre "`restaurantes.py`". En este archivo debe construir el módulo de la lógica implementando las funciones que responden a los requerimientos de la aplicación. **Defina**, **documente** e **implemente** las funciones descritas a continuación en su nuevo archivo. Lea las descripciones de las funciones para determinar los parámetros de entrada con sus tipos, y el valor de retorno con su respectivo tipo. Es fundamental evitar la duplicación de código. Determine si una función debe llamar a otra (composición) para evitar que sus funciones contengan código repetido. Puede usar funciones auxiliares si lo desea. En el caso de recorridos sobre diccionarios o listas, determine si requiere usar un patrón de recorrido total o alternativamente, un recorrido parcial.

Función 1:

Implemente la función `cargar_restaurantes`, la cual carga un archivo *CSV* con la información de los restaurantes. La función recibe el nombre del archivo como único parámetro (`str`), carga los datos en el diccionario de estados y lo retorna. El diccionario tiene como llaves los estados y como valores las listas de diccionarios de restaurantes (ver Figura 1). Cada diccionario de restaurante debe contener las llaves y tipos indicados en la Tabla 1, excepto la llave `"delivery"`, que debe ser del tipo `bool`. Para ello, debe asumir que `1` es equivalente a `True` y `0` es equivalente a `False`.

ATENCIÓN: Use la opción `encoding='utf-8'` al invocar a la función `open()` con el fin de garantizar que el archivo se lea con soporte a caracteres especiales.

Función 2:

Nombre		`buscar_restaurantes_en_area`
Descripción		Busca a los restaurantes ubicados en un área acotada por las latitudes y longitudes dadas por parámetro.
Parámetros		
Nombre	**Tipo**	**Descripción**
`estados`	`dict`	Diccionario de estados.
`latitud_min`	`float`	Límite inferior del área a analizar en latitud.
`latitud_max`	`float`	Límite superior del área a analizar en latitud.
`longitud_min`	`float`	Límite izquierdo del área a analizar en longitud.
`longitud_max`	`float`	Límite derecho del área a analizar en longitud.
Retorno	`list`	Lista de restaurantes ubicados en el área especificada. Si no se encuentran restaurantes en el área, retorna una lista vacía. Si consideramos el CSV Ejemplo, la lista retornada dados los parámetros: `latitud_min = 34`, `latitud_max = 36`, `longitud_min = -78` y `longitud_max = -76`, sería: `[{'name': 'KFC', 'category': 'Chicken', 'rating': 3.7, 'latitude': 35.2714408, 'longitude': -77.6034355, 'city': 'Kinston', 'state': 'North Carolina', 'price': '$$$', 'review_count': 9448, 'delivery': True, 'address': '1613 W Vernon Ave'}]`

Función 3:

<table>
<tr><td colspan="2">Nombre</td><td><code>buscar_restaurante_mas_sucursales</code></td></tr>
<tr><td colspan="2">Descripción</td><td>Busca el restaurante con más sucursales en un estado.

Recuerde que las sucursales se refieren a sedes del restaurante que tienen el mismo nombre, pero diferentes ubicaciones.</td></tr>
<tr><td colspan="3">Parámetros</td></tr>
<tr><td>Nombre</td><td>Tipo</td><td>Descripción</td></tr>
<tr><td><code>estados</code></td><td><code>dict</code></td><td>Diccionario de estados.</td></tr>
<tr><td><code>estado_buscado</code></td><td><code>str</code></td><td>Estado del cual se quiere encontrar el restaurante con más sucursales.</td></tr>
<tr><td>Retorno</td><td><code>dict</code></td><td>Diccionario con las llaves:

<code>"nombre_restaurante"</code>: Nombre del restaurante con más sucursales en el estado.

<code>"numero_sucursales"</code>: Número de sucursales del restaurante.

Si hay un empate, se retorna el primero encontrado.

Si no se encuentran restaurantes en el estado, retorna: <code>{"nombre_restaurante":"", "numero_sucursales": 0}</code>.

Si consideramos el CSV Ejemplo, el diccionario retornado para el estado <code>"Massachusetts"</code>, sería: <code>{'nombre_restaurante': 'IHOP', 'numero_sucursales': 2}</code></td></tr>
</table>

Función 4:

Las direcciones de los restaurantes pueden incluir abreviaciones. Debe estandarizarlas reemplazando las abreviaciones por su forma completa, como se indica en las siguientes tablas:

Abreviación	Expansión
St	Street
Ave	Avenue
Sq	Square
Hwy	Highway
Blvd	Boulevard
Rd	Road
Dr	Drive
Ln	Lane
Ct	Court

Abreviación	Expansión
Pl	Place
Pk	Park
Cir	Circle
Expy	Expressway
Trl	Trail
Rte	Route
Mt	Mount
Fwy	Freeway

Tabla 2. Forma completa de abreviaciones usadas en las direcciones de restaurantes en el archivo `restaurantes.csv`

ATENCIÓN: Las abreviaciones pueden tener mayúsculas y minúsculas por lo que se recomienda el uso de funciones como `str.capitalize()`. También, recuerde la utilidad de `str.replace()`, `str.split()` y `str.join()`.

A continuación, se provee una descripción de la función deseada:

Nombre de la función		`estandarizar_direcciones`
Descripción		Reemplaza las abreviaciones en las direcciones de los restaurantes que se describen en la Tabla 2, por su forma completa.
Parámetros		
Nombre	**Tipo**	**Descripción**
`estados`	`dict`	Diccionario de estados.
Retorno	`None`	Esta función **modifica directamente el diccionario de estados**, actualizando las direcciones, de tal forma que las abreviaciones se reemplazan por su forma completa. Si consideramos **únicamente** al restaurante `"Aziza"` en el CSV Ejemplo, su dirección modificada en el diccionario de estados después de ejecutar la función sería: `"5800 Geary Boulevard"`.

ATENCIÓN: La función `estandarizar_direcciones` altera **permanentemente** el diccionario de estados. Por lo tanto, dependiendo del orden en el que decida correr las funciones en la consola, las direcciones de los restaurantes pueden aparecer con abreviaciones o en su forma completa.

Función 5:

Nombre		`buscar_restaurantes_palindromos`
Descripción		Busca a todos los restaurantes cuyos nombres sean palíndromos, ignorando mayúsculas y espacios. Ejemplo: Si un restaurante se llamara `"Evil Olive"`, se consideraría un palíndromo.
Parámetros		
Nombre	**Tipo**	**Descripción**
`estados`	`dict`	Diccionario de estados.
Retorno	`list`	Lista de diccionarios con la información de los restaurantes cuyos nombres sean palíndromos. Si no se encuentra ningún restaurante con nombre palíndromo, retorna una lista vacía. Si consideramos el CSV Ejemplo, el diccionario retornado sería: `[{'name': 'Aziza', 'category': 'Moroccan', 'rating': 4.6, 'latitude': 37.7804334, 'longitude': -122.481706, 'city': 'San Francisco', 'state': 'California', 'price': '$$$', 'review_count': 5068, 'delivery': True, 'address': '5800 Geary Blvd'}]`

Función 6:

Nombre		`buscar_restaurante_cercano`
Descripción		Busca el restaurante más cercano a un punto de referencia (latitud y longitud) ingresado por parámetro utilizando la distancia cartesiana (ver F1).
Parámetros		
Nombre	**Tipo**	**Descripción**
`estados`	`dict`	Diccionario de estados.
`latitud_ref`	`float`	Latitud del punto de referencia.
`longitud_ref`	`float`	Longitud del punto de referencia.
Retorno	`dict`	Diccionario del restaurante más cercano al punto de referencia. En caso de que haya dos restaurantes a la misma distancia, retornar el último encontrado. Si consideramos el CSV Ejemplo, el diccionario retornado dados los parámetros: `latitud_ref = 35.01` y `longitud_ref = -106.68`, sería: `{'name': 'Los Pollos Hermanos', 'category': 'Chicken', 'rating': 4.8, 'latitude': 35.0145774, 'longitude': -106.6875825, 'city': 'Albuquerque', 'state': 'New Mexico', 'price': '$', 'review_count': 9689, 'delivery': True, 'address': '699 Susan Case ave'}`

Función 7:

Nombre de la función		`buscar_restaurante_preferido`
Descripción		Busca el primer restaurante en un área determinada que cumpla con preferencias de precio, número de *reviews* y *rating*.
Parámetros		
Nombre	**Tipo**	**Descripción**
`estados`	`dict`	Diccionario de estados.
`latitud_min`	`float`	Límite inferior del área a analizar en latitud.
`latitud_max`	`float`	Límite superior del área a analizar en latitud.
`longitud_min`	`float`	Límite izquierdo del área a analizar en longitud.
`longitud_max`	`float`	Límite derecho del área a analizar en longitud.
`precio_maximo`	`str`	Precio máximo que está dispuesto a pagar.
`minimo_reviews`	`int`	Umbral de número de *reviews* que debe superar un restaurante para ser considerado en la función.
`rating_minimo`	`float`	Umbral de *rating* que debe superar un restaurante para ser considerado en la función.
Retorno	`dict`	Diccionario con la información del primer restaurante que cumpla con las preferencias. Si no se encuentra ningún restaurante, retorna un diccionario vacío. Si consideramos el CSV Ejemplo, el diccionario retornado dados los parámetros: `latitud_min` = 37, `latitud_max` = 39, `longitud_min` = -99, `longitud_max` = -97, `precio_maximo` = "$$$$", `minimo_reviews` = 4000 y `rating_minimo` = 4, sería: `{'name': 'Pizza Hut', 'category': 'Pizza delivery', 'rating': 4.3, 'latitude': 38.3720139, 'longitude': -97.6241374, 'city': 'McPherson', 'state': 'Kansas', 'price': '$$$$', 'review_count': 4881, 'delivery': False, 'address': '2215 E Kansas Ave'}`

Actividad 4 | Completar la interfaz de usuario basada en consola

1. En esta actividad usted tiene que construir la interfaz basada en consola para que el usuario interactúe con la aplicación. Para construir esta interfaz usted debe completar el archivo "`consola_restaurantes.py`". Note que las funciones de este módulo están debidamente documentadas y tienen etiquetas `TODO`, numeradas (1-5) indicando que usted debe completarlas. Por favor reemplace la instrucción `pass` con su implementación de la función según la documentación.

ATENCIÓN: En la consola se provee las funciones: `mostrar_restaurante` y `mostrar_restaurantes` para facilitar la impresión de la información de los restaurantes. La función: `ejecutar_estandarizar_direcciones` provee un ejemplo de su uso.

2. Pruebe la interfaz basada en consola ejecutando el archivo "**`consola_restaurantes.py`**". Verifique que las funcionalidades de su aplicación se comporten de acuerdo con lo esperado.
3. Pruebe su aplicación cargando el archivo "**`restaurantes.csv`**" o cree su propio archivo de prueba respetando el mismo formato. Si una función presenta errores o resultados inesperados, revise su implementación. Los errores comunes incluyen nombres incorrectos, argumentos insuficientes, en exceso, o en el orden equivocado. Repita las pruebas tras cada corrección. Cuantas más pruebas (de casos normales, extremos y anormales) realice, más indicios tendrá de que su programa está bien construido.

ATENCIÓN: Su interfaz debe seguir el estándar de construcción de consolas del curso y debe solicitar al usuario todos los valores de argumentos requeridos para invocar correctamente a las funciones de la lógica.

Entrega

1. Comprima la carpeta con su proyecto resuelto. El archivo debería llamarse **`N3-PROY-login.zip`**, donde *login* es su nombre de usuario de Uniandes.

ATENCIÓN: El archivo debe ser un comprimido del tipo `.zip`.

2. Entregue el archivo comprimido a través de Bloque Neón en la tarea designada como **Proyecto del Nivel 3**.

IMPORTANTE: La solución del proyecto debe ser de su completa autoría.