Gestión de la Información en la Web
Curso 2023-24
Práctica 2 – Formatos CSV y JSON

Fecha de entrega: domingo 15 de octubre de 2023

Entrega de la práctica
La entrega de la práctica se realizará a través del Campus Virtual de la asignatura mediante un fichero `formatos_csv_json.py`. El esqueleto de este fichero se puede descargar del Campus Virtual.

Lenguaje de programación
Python 3.10 o superior.

Calificación
Se medirá la corrección mediante tests de unidad. Además de la corrección se valorará la **calidad, concisión y claridad del código**, la incorporación de **comentarios** explicativos y su **eficiencia** tanto en tiempo como en memoria.

Declaración de autoría e integridad
Todos los ficheros entregados contendrán una cabecera en la que se indique la asignatura, la práctica, el grupo y los autores. Esta cabecera también contendrá la siguiente declaración de integridad:

> *Declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona o sistema automático ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con otras personas de manera directa o indirecta. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.*

No se corregirá ningún fichero que no venga acompañado de dicha cabecera.

---

1. Formato CSV [5pt]

Considerar el archivo `AccidentesBicicletas_2021.csv`. El archivo contiene información sobre los accidentes en los que se han visto involucradas bicicletas en la ciudad de Madrid durante el año 2021. Se puede encontrar más información del conjunto de datos en `https://datos.madrid.es/FWProjects/egob/Catalogo/Seguridad/Ficheros/Estructura_DS_Accidentes_Bicicletas_desde_2019.pdf`

Se pide crear funciones que realicen las siguientes operaciones:

1. (1pt) `lee_fichero_accidentes(ruta)`
   Leer los datos del archivo y devolverlos en una lista de diccionarios, una por cada línea. El resultado con el fichero `AccidentesBicicletas_2021.csv` debería empezar así:

```
[{'cod_distrito': '14',
  'cod_lesividad': '2',
  'coordenada_x_utm': '446.426.917',
  'coordenada_y_utm': '4.473.586.644',
  'distrito': 'MORATALAZ',
  'estado_meteorológico': 'Despejado',
  'fecha': '01/01/2021',
  'hora': '11:38:00',
  'localizacion': 'CALL. JOSE BERGAMIN / CALL. FLORENCIO CANO CRISTOBAL',
  'num_expediente': '2021S000047',
  'numero': '62',
  'positiva_alcohol': 'N',
  'positiva_droga': 'NULL',
  'rango_edad': 'De 45 a 49 años',
  'sexo': 'Hombre',
  'tipo_accidente': 'Caída',
  'tipo_lesividad': 'Ingreso inferior o igual a 24 horas',
  'tipo_persona': 'Conductor',
  'tipo_vehículo': 'Bicicleta'},
 {'cod_distrito': '20',
  'cod_lesividad': '14',
  'coordenada_x_utm': '448.606.811',
  'coordenada_y_utm': '4.476.216.426',
  'distrito': 'SAN BLAS-CANILLEJAS',
  'estado_meteorológico': 'Despejado',
  'fecha': '04/01/2021',
  'hora': '11:30:00',
  'localizacion': 'PLAZA. GRECIA / AVDA. ARCENTALES wanda farola 20',
  'num_expediente': '2021S000142',
  'numero': '1',
  'positiva_alcohol': 'N',
  'positiva_droga': 'NULL',
  'rango_edad': 'De 15 a 17 años',
  'sexo': 'Hombre',
  'tipo_accidente': 'Caída',
  'tipo_lesividad': 'Sin asistencia sanitaria',
  'tipo_persona': 'Conductor',
  'tipo_vehículo': 'Bicicleta'},
...
]
```

2. (1pt) `accidentes_por_distrito_tipo(datos)`
   Obtener un diccionario que por cada distrito (columna `distrito`) y por cada tipo de accidente

(columna `tipo_accidente`) indica cuántos accidentes hubo. Esta función acepta como parámetro `datos` la lista de diccionarios obtenida del fichero CSV. El resultado de invocar a esta función con los datos del fichero `AccidentesBicicletas_2021.csv` sería:

```
{('ARGANZUELA', 'Alcance'): 11,
 ('ARGANZUELA', 'Atropello a persona'): 6,
 ('ARGANZUELA', 'Caída'): 26,
 ('ARGANZUELA', 'Choque contra obstáculo fijo'): 1,
 ('ARGANZUELA', 'Colisión fronto-lateral'): 9,
 ('ARGANZUELA', 'Colisión lateral'): 5,
 ('ARGANZUELA', 'Colisión múltiple'): 1,
 ('BARAJAS', 'Alcance'): 1,
 ('BARAJAS', 'Atropello a persona'): 4,
 ('BARAJAS', 'Caída'): 2,
 ('BARAJAS', 'Choque contra obstáculo fijo'): 3,
 ('BARAJAS', 'Colisión frontal'): 1,
 ('BARAJAS', 'Colisión fronto-lateral'): 8,
 ('BARAJAS', 'Colisión lateral'): 2,
 ...
}
```

3. (1pt) `dias_mas_accidentes(datos)`
   Función que devuelve las fechas del día o días con más accidentes, junto con ese número de accidentes, tomando como entrada una lista de diccionarios como en el primer apartado. Se debe devolver un **conjunto de parejas** `(día, número de accidentes)`. La salida esperada para el fichero `AccidentesBicicletas_2021.csv` sería:

```
{('24/04/2021', 9), ('13/10/2021', 9)}
```

4. (2pt) `puntos_negros_distrito(datos, distrito, k)`
   Función que dado un distrito, genera una lista con el *top-k* de localizaciones donde más accidentes se han producido. Esa lista incluirá parejas `(localización, número de accidentes)` ordenada de manera descendente por número de accidentes. En caso de empate a número de accidentes, se mostrarán primero las localizaciones con un nombre lexicográficamente mayor, es decir, `'PASEO. PIÑONERO, 4'` deberá aparecer antes de `'CTRA. PARQUE DE ATRACCIONES'`. Para el fichero `'AccidentesBicicletas_2021.csv'` y el distrito `'MONCLOA-ARAVACA'`, el top-16 debería ser:

```
[('PASEO. PIÑONERO, 4', 3),
 ('CTRA. PARQUE DE ATRACCIONES, 1', 3),
 ('CTRA. CIUDAD UNIVERSITARIA, 0', 3),
 ('PASEO. TORRECILLA / PASEO. AZUL', 2),
 ('PASEO. REY, 22', 2),
 ('PASEO. EMBARCADERO, 6', 2),
 ('PASEO. EMBARCADERO / PASEO. AZUL', 2),
 ('PASEO. AZUL / PASEO. EMBARCADERO', 2),
 ('PASEO PIÑONEROS KM 46,708 VIA VERDE CASA DE CAMPO', 2),
 ('CUSTA. SAN VICENTE, 44', 2),
 ('CTRA. GARABITAS, 0', 2),
 ('CTRA. CASTILLA, +00100S', 2),
 ('CALL. JOSE ANTONIO NOVAIS / CALL. RAMIRO DE MAEZTU', 2),
 ('CALL. IRUN, 1', 2),
 ('CALL. ANICETO MARINAS, 74', 2),
 ('AVDA. COMPLUTENSE, 14', 2)
]
```

2. Formato JSON [5pt]

Considerar el archivo `300356-0-monumentos-ciudad-madrid.json`. El archivo contiene información sobre los monumentos de Madrid. En este apartado hay que implementar una serie de funciones:

1. (1pt) `leer_monumentos(json_file)`
   Esta función acepta la ruta del fichero JSON y devuelve una lista de monumentos, cada uno representado como un diccionario Python.

2. (1pt) `subtipos_monumentos(monumentos)`
   Función que acepta una lista de monumentos y devuelve un **conjunto** con todos los subtipos de monumentos que se ha encontrado. La salida esperada para el fichero anterior sería:

```
{'Edificación singular',
 'Elemento conmemorativo, Lápida',
 'Elemento de ornamentación',
 'Escultura conceptual o abstracta',
 'Estatua',
 'Fuente, Estanque, Lámina de agua',
 'Grupo Escultórico',
 'Puente, construcción civil',
 'Puerta, Arco triunfal'}
```

3. (1pt) `busqueda_palabras_clave(monumentos, palabras)`
   Esta función recibe una lista de monumentos y una lista de palabras clave, y devuelve parejas `(nombre, distrito)` de aquellos monumentos que contienen las palabras clave en su nombre o en su descripción. Es necesario que aparezcan todas las palabras clave entre los dos campos para considerar que el monumento encaja con la búsqueda, pero no hay que tener en cuenta las mayúsculas o minúsculas.

   La salida esperada para los monumentos que encajan con las palabras clave `['escultura', 'agua']` sería:

```
{('Alfonso XII', 'DISTRITO'),
 ('Bravo Murillo', 'CHAMBERI'),
 ('Dodecaedro-Éter', 'ARGANZUELA'),
 ('Estanque Descubrimiento', 'SALAMANCA'),
 ('Felipe IV', 'CENTRO'),
 ('Francisco de Quevedo y Villegas', 'CHAMBERI'),
 ('Fuente Poliedros Maclados en el Cubo', 'SAN BLAS'),
 ('Fuente de Cibeles', 'RETIRO'),
 ('Fuente de Credit Lyonnais', 'CHAMBERÍ'),
 ('Fuente de la Cruz Verde', 'CENTRO'),
 ('Fuente de la Unión y el Fénix', 'CHAMBERÍ'),
 ('Fuente de los Delfines', 'CHAMARTIN'),
 ('Fuente de los Tritones', 'CENTRO'),
 ('Fuente del Ministerio de Economía', 'CHAMARTIN'),
 ('Fuente del Ministerio de Industria', 'CHAMARTIN'),
 ('Fuente del parque de las Naciones', 'CHAMBERÍ'),
 ...
}
```

4. (2pt) `busqueda_distancia(monumentos, calle, distancia)`
   Función que devuelve una lista de ternas `(nombre, subtipo, distancia en kilómetros)` de los monumentos que se encuentran a menos de la distancia indicada desde la calle pasada como parámetro. La lista de monumentos deberá estar ordenada de más cercano a lejano.

Para conocer la longitud y latitud de una calle usaremos la biblioteca GeoPy. Aquí tenéis un ejemplo de uso para conocer la longitud y latitud de la calle donde se encuentra la Facultad de Informática, `'Profesor José García Santesmases, Madrid, España'`:

```
from geopy.geocoders import Nominatim

geolocator = Nominatim(user_agent="Ejemplo")
location = geolocator.geocode(
  "Profesor José García Santesmases, Madrid, España",
  addressdetails=True)
print((location.latitude, location.longitude))
```

Por otro lado, para calcular la distancia entre dos puntos dados como (longitud, latitud) usaremos la misma biblioteca GeoPy, en este caso el módulo `distance`:

```
from geopy import distance

wellington = (-41.32, 174.81)
salamanca = (40.96, -5.50)

print(distance.distance(wellington, salamanca).km)
```

La salida esperada de los documentos a menos de 1 km de la calle `"Profesor José García Santesmases, Madrid, España"` sería la siguiente:

```
[('José Ortega y Gasset', 'Estatua', 0.7073967405324207),
 ('Camilo José Cela', 'Estatua', 0.8021271065701585),
 ('Puerta de Hierro', 'Puerta, Arco triunfal', 0.8683981362185172)
]
```