Estructura de Datos

Trabajo Práctico Nº 1: Motor de Rutas

Fechas de entrega
turno martes: 13 de septiembre de 2016
turno jueves: 15 de septiembre de 2016

Enunciado

Necesitamos programar un motor para operar con rutas entre ciudades, para lo cual vamos a utilizar POO, Python 3.X y la API de Google Maps para obtener la distancia entre ciudades y otros datos.

Se define como ruta al camino entre dos ciudades y trayecto a una sucesión ordenada de ciudades, tales que existe una ruta entre dos ciudades consecutivas.

El motor de rutas debe permitir realizar las siguientes operaciones (vía menú de texto):

- Crear un trayecto a partir de dos ciudades: Dado el nombre de dos ciudades, debe buscar en la API de Google si hay un camino entre los puntos dados, y si no hay camino debe mostrar un mensaje de error. Origen y destino deben ser ciudades distintas. Además cada trayecto tiene un nombre que lo identifica.

- Agregar una ciudad al final de un trayecto: Dado un trayecto y el nombre de una ciudad, debe agregar la ciudad al final del trayecto, si no es posible debe mostrar un mensaje de error.

- Agregar una ciudad intermedia a un trayecto: Dado un trayecto, una ciudad que pertenece a ese trayecto y el nombre de otra ciudad, agregar la nueva ciudad antes de la ciudad que ya pertenece al trayecto. Para poder agregarla debe verificar que hay rutas para que el trayecto final sea válido.

- Concatenar dos trayectos: Dados dos trayectos cualquiera concatenarlos si existe una ruta entre la última ciudad del primer trayecto y la primera ciudad del segundo trayecto. En caso de error se debe mostrar un mensaje.

- Comparar dos trayectos: Dados dos trayectos los debe poder comparar por distancia y por tiempo, para lo cual se usarán los parámetros: "d" para comparar por distancias y "t" para comparar por tiempo.

- Mostrar un trayecto: Debe mostrar el nombre o identificador del trayecto y la lista ordenada de ciudades, la distancia total (suma de las rutas entre ciudades consecutivas) y tiempo total estimado de viaje (suma de los tiempos de cada ruta) con el siguiente formato:

```
<nombre>: <ciudad1>, <ciudad2>,...,<ciudadn>
distancia: <distancia en km> km
tiempo estimado de viaje: <dias> días, <horas>, hs
```

- Mostrar rutas: Dado un trayecto debe mostrar todas las rutas que forman el trayecto con el siguiente formato:

```
<origen 1> - <destino 1>
<distancia en km> km
<dias> días, <horas> hs
<línea en blanco>
<destino 1> - <destino 2>
<distancia en km> km
<dias> días, <horas> hs
.
.
.
```

- Listar los trayectos calculados: Debe listar los nombres de los trayectos ya calculados que se encuentran en el sistema.
- Almacenar en disco los trayectos calculados: Debe permitir almacenar un trayecto dado su nombre o todos los trayectos si no se especifica un trayecto particular.
- Recuperar de disco los trayectos almacenados: Debe recuperar todos los trayectos almacenados en disco
- Salir del sistema: antes de salir debe guardar todos los trayectos en memoria.

Para cada ruta se debe almacenar la siguiente información:

- Origen
- Destino
- Distancia en km
- Tiempo estimado de viaje

y para cada trayecto:

- Lista ordenada de ciudades
- Distancia total en km
- Tiempo total estimado

Entrega

El TP se podrá realizar en forma individual o en grupos de hasta dos personas.
La entrega se realizará por mail hasta la fecha indicada a las siguientes direcciones:

```
mfranzo+tp1+ed2016@gmail.com
rositaw+tp1+ed2016@gmail.
```

Se deberá entregar el código fuente, incluyendo el archivo de configuración config.py sin la clave de google (se adjunta ejemplo del archivo config.py)
Informe en pdf con al menos la siguiente información:
Carátula identificando el trabajo y los autores
Diagrama de clases
Decisiones de diseño
Análisis de la API Google Maps Distance Matrix (resumen para comprender como la utilizan en este TP)
Conclusiones

Anexo: API Google Maps en Python

Para consultar las rutas entre ciudades se va a utilizar la API de Google Maps a través de la librería *Python Client for Google Maps Services* disponible en https://github.com/googlemaps/google-m
La forma más sencilla de instalarla es a través del comando:

```
$ pip install -U googlemaps
```

a veces, y dependiendo de la configuración se debe utilizar pip3 en lugar de pip
Se adjunta archivo con ejemplo de uso.
Para poder utilizar las API de Google es preciso habilitar una clave para la aplicación.
La clave se puede obtener siguiendo los siguientes pasos:

1. Visitar https://developers.google.com/console e iniciar sesión con una cuenta de google.
2. Crear un nuevo proyecto.
3. Habilitar Google Maps Distance Matrix API
4. Crear una server key para la API habilitada

Si todo funcionó bien, se debe poder ejecutar el ejemplo dado, completando el string KEY del archivo config.py