Práctica 5. Tipos de datos estructurados en Python

1. Objetivos

- Comprender el manejo de datos estructurados en Python mediante diccionarios y listas.
- Implementar funciones modulares para organizar mejor el código.
- Leer, procesar y analizar datos de archivos estructurados.
- Desarrollar un programa interactivo que analice información de un sistema de metro.

2. Introducción

El manejo de datos estructurados en Python se basa principalmente en estructuras como diccionarios y listas, las cuales permiten organizar la información de manera eficiente y flexible. Los diccionarios, en particular, son útiles para almacenar datos de manera asociativa, permitiendo una búsqueda rápida de valores mediante claves. Las listas, por su parte, facilitan la manipulación de conjuntos de datos ordenados.

En esta práctica, se procesarán archivos de texto que contienen datos sobre estaciones y registros de viajes del metro. Para extraer información clave, se aplicarán técnicas de lectura de archivos y manipulación de cadenas de texto, como segmentación e indexación. Finalmente, se implementarán funciones para calcular estadísticas relevantes, como la cantidad de viajes, hora pico y trayectos más frecuentes, demostrando la utilidad del manejo eficiente de datos estructurados.

3. Requerimientos del programa

Esta práctica consiste en la implementación de un sistema de análisis de datos del metro. Los estudiantes deben implementar funciones en Python para leer y procesar información de archivos de texto, y presentar estadísticas mediante un menú interactivo.

3.1. Resumen del Funcionamiento

El programa debe iniciar cargando la información de las estaciones desde un archivo (`stations.info`) y los registros de viajes desde otro (`metro.log`). Luego, presenta un menú interactivo que permite al usuario consultar estadísticas generales del sistema de metro o estadísticas específicas de una estación. Las estadísticas generales incluyen el número total de

usuarios, cantidad de viajes, hora pico, ingresos totales y trayectos más populares. Las estadísticas por estación permiten conocer la cantidad de viajes, las horas pico de ingreso y salida, y las estaciones de origen y destino más frecuentes. Para ello, el código utiliza estructuras de datos como diccionarios y listas, aplicando funciones modulares definidas en **`stations.py`** y **`system.py`**, garantizando así un código organizado y eficiente.

A continuación se muestra un ejemplo del formato utilizado que tiene almacenados los registros de viaje en el archivo `metro.log`, el cuál contiene la siguiente información:

- `STATION_ID`: Número de la estación (según el archivo `stations.info`)
- `USER_ID`: Número que identifica al usuario que ingresa o sale de la estación
- `EVENT_TIME`: Hora del evento en formato 24 horas.
- `EVENT_TYPE`: Tipo de evento, `IN` si el usuario ingresó a la estación, `OUT` si sale de la estación.

metro.log

```
STATION_ID    USER_ID   EVENT_TIME  EVENT_TYPE
010           69253198  04:03       IN
002           67143113  04:03       IN
002           69253198  04:46       OUT
018           67143113  04:53       OUT
```

Respecto a la información relacionada a las estaciones almacenada en `stations.info`, se tiene el siguiente formato:

- `ESTACION`: Nombre de la estación del metro.
- `ESTACION_ID`: Número que identifica a cada estación.
- `LATITUD, LONGITUD`: Coordenadas geográficas de cada estación.

stations.info

```
ESTACION,ESTACION_ID,LATITUD,LONGITUD
Niquía,001,6.337783,-75.544365
Bello,002,6.330476,-75.553434
Madera,003,6.315388,-75.555302
```

3.2. Funcionamiento del programa

El programa debe iniciar mostrando un menú con opciones para consultar estadísticas generales y estadísticas de una estación específica.

Las estadísticas generales incluyen:

- Número total de usuarios y viajes
- Hora pico
- Estaciones más usadas
- Distancia promedio de viaje
- Ingresos totales
- Número promedio de viajes
- Los 5 trayectos más populares

Las estadísticas por estación incluyen:

- Total de viajes en la estación
- Horas pico de ingreso y salida
- Estaciones de origen y destino más comunes
- Cantidad de usuarios que ingresaron por hora del día.
- Cantidad de usuarios que salieron por hora del día.

3.3. Estructura del programa

El código fuente debe estar organizado en los siguientes archivos:

- **`main.py`**: Contiene la estructura general del programa y las llamadas a las funciones.
- **`stations.py`**: Contiene funciones para manejar información sobre las estaciones.
- **`system.py`**: Contiene funciones de análisis general sobre los datos del metro.

4. Evaluación

La evaluación se basará en la correcta implementación de las funciones requeridas, el funcionamiento del menú interactivo y la claridad del código. Se tendrá en cuenta:

- Correcta estructura y uso de funciones modulares.
- Funcionamiento del programa de acuerdo a lo especificado.
- Lectura y procesamiento de archivos correctamente implementados.
- Documentación y comentarios en el código.