Task Execution Analysis

Introduction

For this project implementation we have chosen a premise where tasks are scheduled for execution and the relevant data needs to be stored and analyzed further. To implement this project we will use concepts such as searching, sorting, linked lists.

Assume a scenario in which you are receiving data for each task that includes *task_id, start_time and end_time*. *start_time* and *end_time* are available in equivalent integer format. Each of these data points need to be stored in a node that will have three values. Once the data points are available and stored in the linked list we will perform analysis based on the data points that are stored in the linked list.

Housekeeping points

- This is a minimal example and may not follow some standard practices
- The focus is on the main flow, with minimal error handling. Errors in validation logic should be handled appropriately though.

Program Organization

This section will contain the information related to the different files available in the project.

1. *src/LinkedList.py:* This file will consist of the implementation related to the task/jobs that need to be stored in form a linked list. This will have a class named as linked list that will contain the insert, update and delete operation for the linked list. This file will also include the structure of the Node that needs to be created in order to store the details of each task/job.
2. *src/Aggregate.py:* This python file will have implementation related to the various analysis that will be performed on the saved tasks. This will have implementation related to finding the minimum, maximum and average of the given stored data.
3. *config.py:* This file will contain the pre-defined data points that can be used directly to create the nodes and which can be used to set up the linked list.
4. *main.py:* This will have method calls that will initiate the linked list creation process and this will also invoke the methods related to tasks mentioned in the problem statement section

Problem Statement

The program structure is already set and there are specific methods that you are expected to implement. Please also read the comments in the code, especially in the methods to be implemented. You can modify main.py to add further demonstration calls.

1. **LinkedList.py:** This file already has the class structure and constructor defined in it to save the data for each node. You have to work on the following method implementation
    a. *insert_node:* This method is supposed to insert the node in the linked list. You can design this function to insert the data either at the end or beginning of the linked list.

    b. *print_linked_list:* This method will print the linked list by traversing each node in the linked list. This should print the task_id atleast for each node.

    c. *print_in_reverse:* Implement this method to print the node information in the reverse order. Please understand that here we are not asking to reverse the linked list. Rather you just need to print the data in the reverse order.

2. **Aggregate.py:** This file contains the implementations as mentioned below related to the created LinkedList:

    a. Mininum_timed_task: This method aims at fetching the minimum timed task from the given LinkedList. If two or more nodes satisfy the condition then please return the task id of the node which is inserted towards the ending of the LinkedList.
        i. For Example : For nodes, we have task_id, start_time and end_time. The difference between start_time and end_time as (end_time - start_time) will be the time required to execute the task. The minimum of the mentioned difference is the minimum timed task.

    b. Maximum_timed_task: This method aims at fetching the maximum timed task from the given LinkedList. If two or more nodes satisfy the condition then please return the task id of the node which is inserted towards the ending of the LinkedList.
        i. For Example : For nodes, we have task_id, start_time and end_time. The difference between start_time and end_time as (end_time - start_time) will be the time required to execute the task. The maximum of the mentioned difference is the maximum timed task.

    c. Average_timed_task: This method aims at fetching the average time of the tasks from the given LinkedList.
        i. For Example : For nodes, we have task_id, start_time and end_time. The difference between start_time and end_time as (end_time - start_time) will be the time required to execute the task. The average of the

mentioned differences for all the nodes is the average of all the times of the tasks from the created LinkedList.

Evaluation Rubric

Total Project Points: **100**

● Basic compilation without errors (**10%**)	: **10 Points**
● Correctness:	
■ Problem statement - 1.a (**10%**)	: **10 Points**
■ Problem statement - 1.b (**10%**)	: **10 Points**
■ Problem statement - 1.c (**10%**)	: **10 Points**
■ Problem statement - 2.a.i (**20%**)	: **20 Points**
■ Problem statement - 2.b.i (**20%**)	: **20 Points**
■ Problem statement - 2.c.i (**20%**)	: **20 Points**

Program Instructions

1. Download the zipped folder named **M01-P02-DSA-Task-Execution-Analysis.zip**, and unzip it on your local machine. Go into the directory named **M01-P02-DSA-Task-Execution-Analysis.**

2. Make sure that you have Python 3.6, or higher, installed. At your command prompt, run:
```
$ python --version
Python 3.7.3
```
   If not installed, install the latest available version of Python 3.

3. To run the code in the source code folder, run the following command:
```
$ python3 main.py (On many Linux/Mac platforms)
           OR
$ python main.py       (On Windows/Mac platforms)
```
   In any case, one of these two commands should work.

4. Alternatively, you could install a popular Python IDE, such as PyCharm or Visual Studio Code, and select a command to build the project from there.

5. You will be making very frequent changes into the LinkedList.py and Aggregate.py python files. The idea is to complete both the scripts so that it satisfies all the requirements. Once you have completed the changes in the code and it is executed

without any error. Zip the folder as **M01-P02-DSA-Task-Execution-Analysis.zip** & now it is ready for submission.