The script is written as a function **FIFO** in Python.

ASSUMPTIONS TAKEN

1. Satellites in concern are Geo-Stationary Satellites – Orbiting over the Equator or in concentric circles with respect to Latitudes for simplification of Distance/Time to the observation strip calculations.
2. Strip Location Parameter used in the script is considered to be the west-most point of the strip, mid-point across width.
3. The influence of width of the Strip is neglected in computing Observation time.
4. Geographic Coordinate System (Latitude and Longitude) is used to represent the Location of the Satellite and the Observation Strip. An added parameter of Altitude exists for the Satellite.
5. Earth velocity: 1670 kmph; Earth Radius: 6378.137 km. Distance units are in Kilometres and Time units are in Hours.
6. Each degree of Latitude is considered to be 110.567 km.
7. It is assumed that Setting Up of the satellite can be finished before starting observation (Start Time).

INPUTS, OUTPUTS & DATA STRUCTURE

The script is deployed considering a constellation of satellites operate and the Data Structures used are capable of handling that. Scalable for single satellite as well as a constellation.

Dataframe is used as the preferred data structure for the ease of working and the ability to handle and process large data.

1. INPUT – I: **DwInt**
    a. DwInt is a Dictionary of Python Pandas Dataframes.
    b. There exist a Dataframe for every satellite in the constellation which has all the info related to the Download schedule of the satellite
    c. The Identifier for a Dataframe in the dictionary is the Satellite ID

	Dwd_task_ID	Ground_Station_ID	Download_SetUp_Time	Download_Start_Time	Donwload_Processing_Time
0	x	x	x	x	x
1	x	x	x	x	x
2	x	x	x	x	x
3	x	x	x	x	x
4	x	x	x	x	x
5	x	x	x	x	x

2. INPUT – II: **SatInfo**
    a. SatInfo is a Dataframe that has the required info related to the satellites in the constellation.

	Satellite_ID	c1	c2	SatLocation_Long	SatLocation_Lat	Sat_Altitude	d_min	d_max	d_status	e_min	e_max	e_status
0	x	x	x	x	x	x	x	x	x	x	x	x
1	x	x	x	x	x	x	x	x	x	x	x	x
2	x	x	x	x	x	x	x	x	x	x	x	x
3	x	x	x	x	x	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x	x	x	x	x	x

3. INPUT – III: **ObsvTasks**
    a. ObsvTasks is a Dataframe that has the required info related to the Observation Strips.

	Obsv_Task_ID	StripLocation_Long	StripLocation_Lat	Strip_Length
0	x	x	x	x
1	x	x	x	x
2	x	x	x	x
3	x	x	x	x
4	x	x	x	x
5	x	x	x	x

4. OUTPUT: **Sat_Schedule**
    a. Sat_Schedule is a Dictionary of Python Pandas Dataframes.
    b. There exist a Dataframe for every satellite in the constellation which has the complete schedule for the satellite obtained using the FIFO algorithm.
    c. The Identifier for a Dataframe in the dictionary is the Satellite ID
    d. Energy_Status and Data_Status carries the amount of Data and Energy left in the satellite after every observation/download.

	Task_ID	Process_Type	SetUp_Time	Start_Time	Processing_Time	Energy Status	Data_Status
0	x	x	x	x	x	x	x
1	x	x	x	x	x	x	x
2	x	x	x	x	x	x	x
3	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x

IMPLEMENTATION

Major points in the implementation of the Algorithm are explained in this section. The points are ordered in same order as the script. For detailed info, refer to the comments in the python script.

1. A point to be noted is that, Initializing Energy and Data status as zero as mentioned in the Publication will cause issues since the script is not supposed to work alone. Nevertheless, the values as mentioned in the Publication is used for Initialization as a placeholder.
2. Observation time windows for the Satellites in the constellation is computed using a nested for-loop. Primary loop goes over the satellites list and secondary loop iterates over observation tasks.
3. The Start Time of observation depends upon the distance from the observation point. The distance and required roll angle are computed in a separate function **distance_and_roll**.
4. SetUp time is computed separately from the roll angle
5. A python Dictionary called **TWO** with Dataframes with possible observation schedules for each satellite is created. The Individual Dataframes for each satellites use Satellite ID as identifier.
6. The Main loop of the script iterates over the **TWO** dictionary with the Observation Scheduling for each Satellite.
7. If an Observation Task is assigned to any satellite in this loop, if the same Observation Task belongs to the schedule of any other satellite Dataframe in **TWO**, it is deleted to avoid repeating of the same task by a different satellite.
8. The Complete schedule combining Observation Schedule from **TWO** and Download Schedule from **DwInt** including downloading tasks after the last observation is stored as and passed on to the main function as a dictionary called **Sat_Schedule** while ensuring no conflicts exist.