All Ways Graph Implementation In Python

In Python, there are several ways to implement graphs, each with its own strengths and use cases.

1. Adjacency List:
   Is a popular graph representation in Python, using a dictionary to store the nodes and their corresponding adjacency lists. Each node in the graph is a key in the dictionary, and its value is a list containing the neighboring nodes connected by an edge. This implementation is particularly efficient for sparse graphs, where the number of edges is much smaller than the total number of nodes.

2. Adjacency Matrix:
   Use a 2D matrix. The rows and columns of the matrix represent vertices, and the elements of the matrix represent edges. This implementation is best suited for dense graphs, where the number of edges is close to the total number of nodes.

3. Edge List:

The edge list representation is the simplest form of graph representation, where we maintain a list of tuples. Each tuple represents an edge between two nodes.

4. Incidence Matrix:
   Use a 2D matrix. where rows represent nodes and columns represent edges. The entry matrix[i][j] is 1 if node i is incident to edge j, and 0 otherwise.

5. Linked List of Linked Lists (Adjacency List of Adjacency Lists):
   Instead of using a single dictionary, we can represent the adjacency list as a list of linked lists. Each node in the main list represents a node in the graph, and the linked list associated with it contains its adjacent nodes.