LAB # 3:

GRAPH THEORY & PATH SEARCHES IN PYTHON

Objectives:

- To familiarize students with the fundamentals of graph theory and path searches in Python

Hardware/Software required:

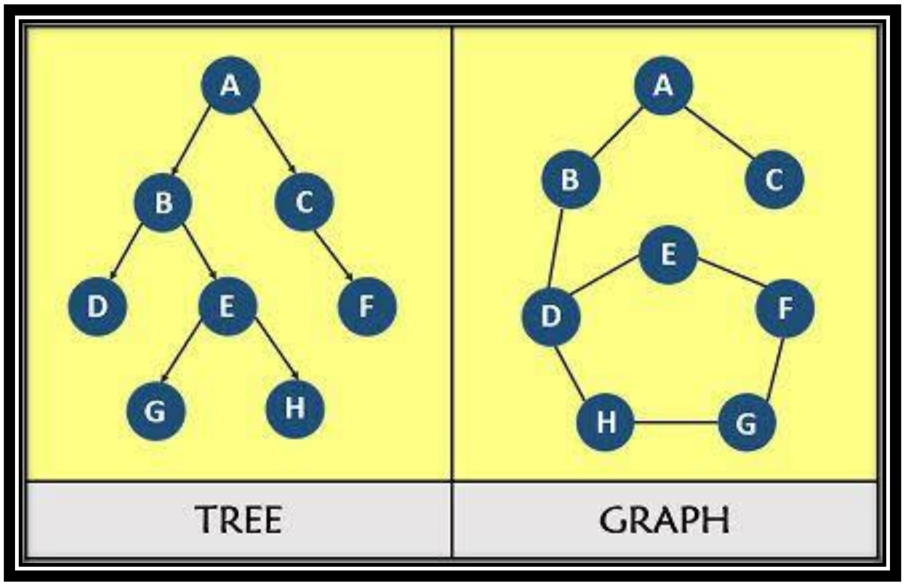
Hardware: Desktop/ Notebook Computer
Software Tool: Python 2.7/ 3.6.2

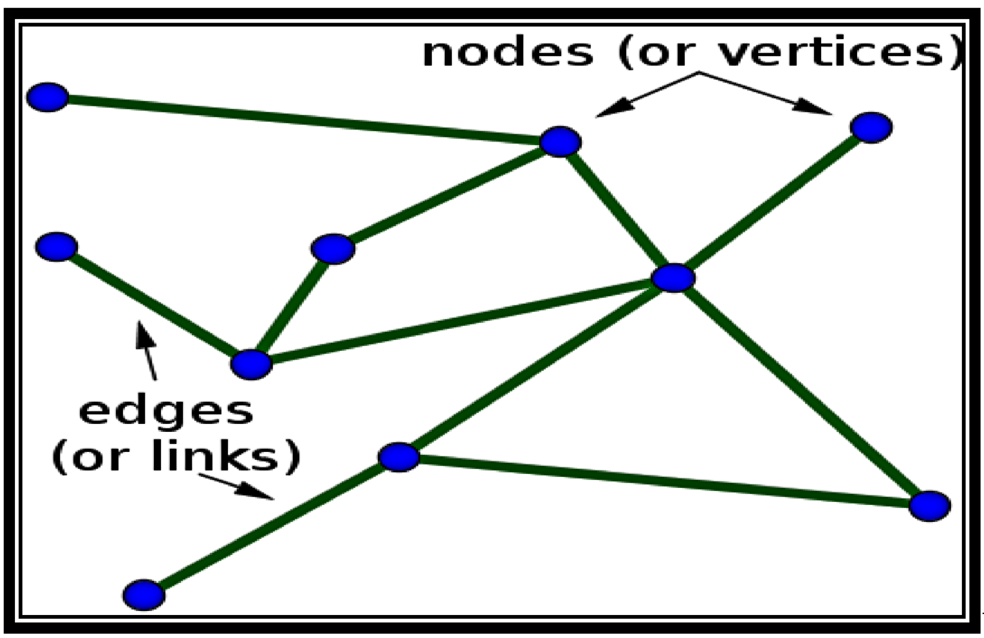
- ## Introduction:

Graph theory is the fundamental concept in mathematics and it lays the foundation of many algorithms in the field of networking, artificial intelligence and image processing. A graph is made up of **vertices, nodes, or points** which are connected by **edges, arcs, or lines.** A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another. **In Python, graphs can be easily implemented through key-value paired data structure i.e. dictionaries.**

- ## Difference between graph and tree:

	Trees	Graphs
Path	Tree is special form of graph i.e. **minimally connected graph** and having only one path between any two vertices.	In graph there can be more than one path i.e. graph can have uni-directional or bi-directional paths (edges) between nodes
Loops	Tree is a special case of graph having no **loops**, no **circuits** and no self-loops.	Graph can have loops, circuits as well as can have **self-loops**.
Root Node	In tree there is exactly one root node and every **child** have only one **parent**.	In graph there is no such concept of **root** node.
Parent Child relationship	In trees, there is parent child relationship so flow can be there with direction top to bottom or vice versa.	In Graph there is no such parent child relationship.
Complexity	Trees are less complex then graphs as having no cycles, no self-loops and still connected.	Graphs are more complex in compare to trees as it can have cycles, loops etc





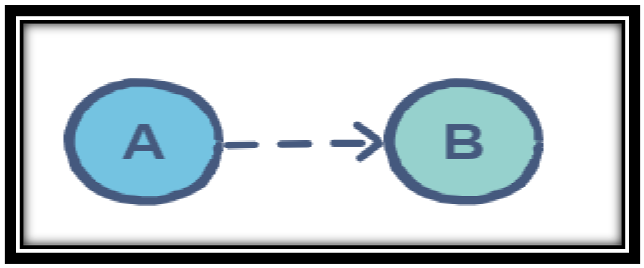
]

- **Loop:**
  If a vertex is connected to itself via an edge, that will be called as loop. ***However we will be considering Simple graphs without loop***

- **Difference between directed and un-directed graphs**

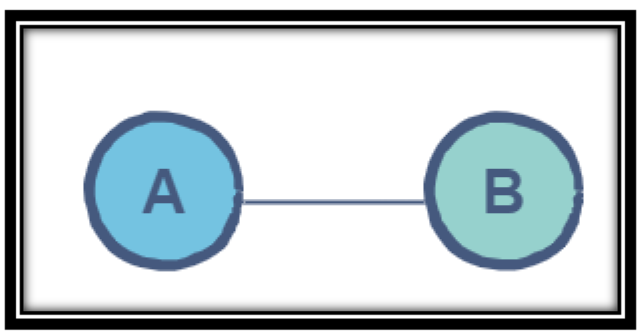
Directed Graph:

A directed graph is a set of vertices (nodes) connected by edges, **with each node having a direction** associated with it.
Edges are usually **represented by arrows** pointing in the direction using which the graph can be traversed.
In the example on the right, the graph can be traversed from vertex A to B, but not from vertex B to A.



Undirected Graph:

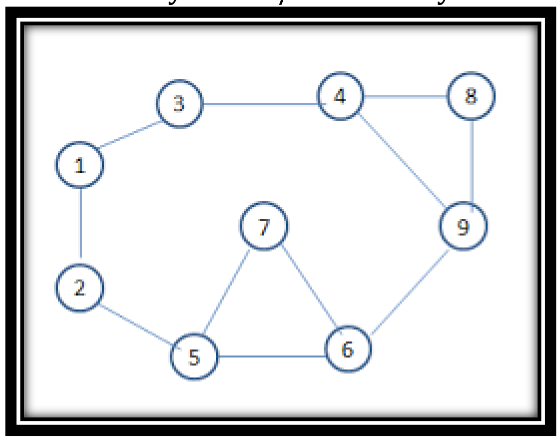
In an undirected graph the edges are **unidirectional**, with no direction associated with them. Hence, the graph can be **traversed in either direction**. The **absence of an arrow** tells us that the graph is undirected.
In the example on the left, the graph can be traversed from node **A** to **B** as well as from node **B** to **A**.



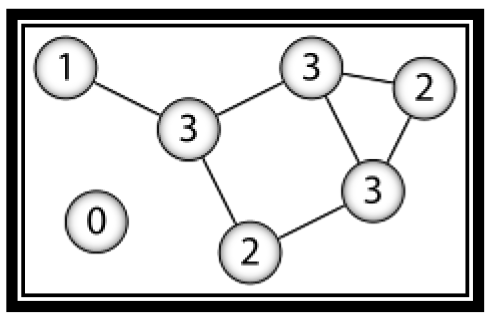
Note: Complex examples of Undirected and Directed graphs are graph1, 2 and graph 3, 4 respectively

- **Connectivity of graph:**

A graph is said to be connected if there is path between every pair of vertex, i.e. there must exist some path to traverse from every vertex/node to any vertex or node



<u>e.g.:</u> here 7 and 4 are not connected directly but there is path between 7 and 4 via 7→6→9→4 (and many others)



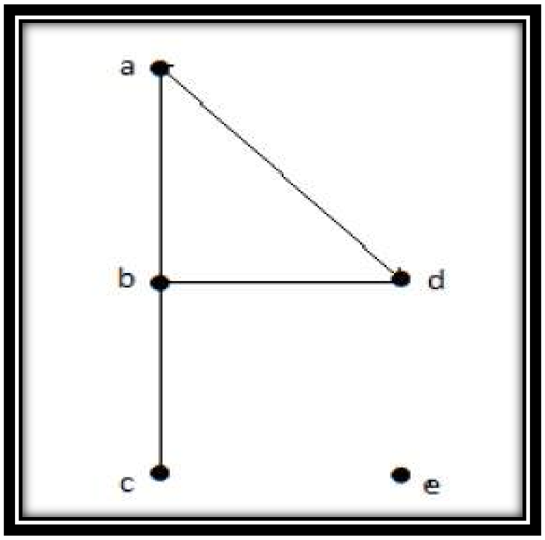
Here we say that graph is connected except for node 0, as there is no way to reach this particular node/vertex.

- **Degree of vertex/node:**

Degree of vertex is the number of the vertices with which it is connected via edges.

DEGREE OF VERTEX IN UNDIRECTED GRAPH:

In undirected graph, degree of the vertex is equal to the number of the vertices with which it is connected



deg(a) = 2 as there are 2 edges meeting at vertex 'a'.

deg(b) = 3 as there are 3 edges meeting at vertex 'b'.

deg(c) = 1 as there is 1 edge formed at vertex 'c'

deg(d) = 2 as there are 2 edges meeting at vertex 'd'.

deg(e) = 0 as there are 0 edges formed at vertex 'e'.

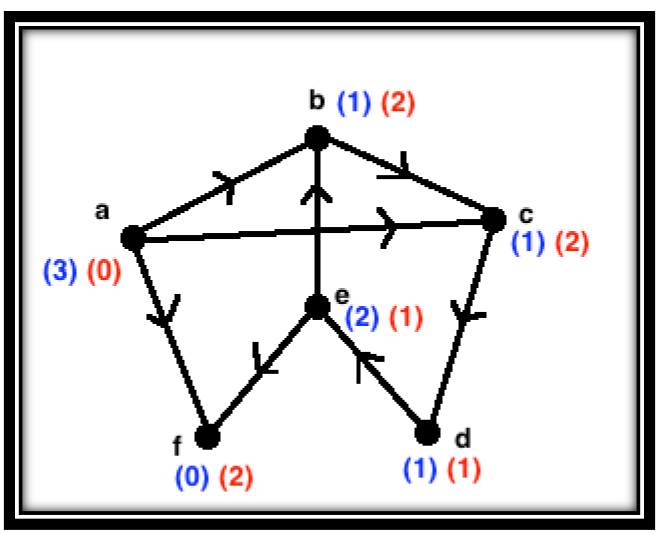
DEGREE OF VERTEX IN DIRECTED GRAPH:

Indegree of a Graph

- Indegree of vertex V is the number of edges which are coming into the vertex V.
- Notation − $deg^-(V)$.

Outdegree of a Graph

- Outdegree of vertex V is the number of edges which are going out from the vertex V.
- Notation − $deg^+(V)$.



In the above graph, the **out-degrees of each vertex are in blue**, while the **in-degrees of each vertex are in red.**

Lab Tasks:

1. Implement the undirected Graph 1 and 2 in Python. Show the connectivity as well as the degree of each node within these graphs.

2. Implement the directed Graph 3 and 4 in Python. Show the connectivity, indegree and outdegree of each node within these graphs.

3. Write a method to find any path between node 6 to node 1 in Graph 1.

4. Write a method to find any path between node E to node C in Graph 2.

5. Write a method to find any path between node 7 to node 9 in Graph 3.

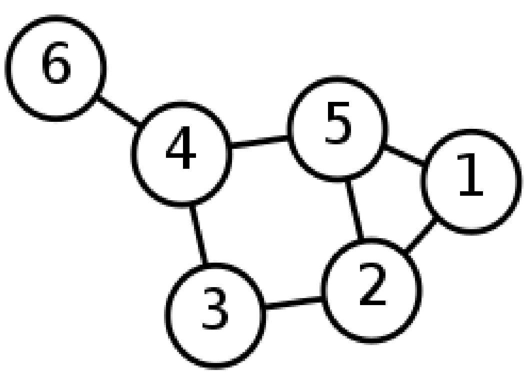
6. Write a method to find any path between node A to node F in Graph 4.

7. Modify Question # 3 to show all possible paths between node 6 to node 1 in Graph 1.

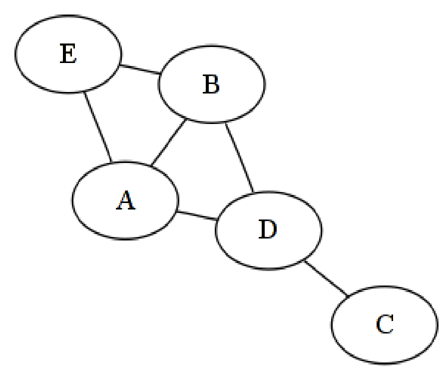
8. Modify Question # 4 to show all possible paths between node E to node C in Graph 2.

9. Modify Question # 5 to show all possible paths between node 7 to node 9 in Graph 3.

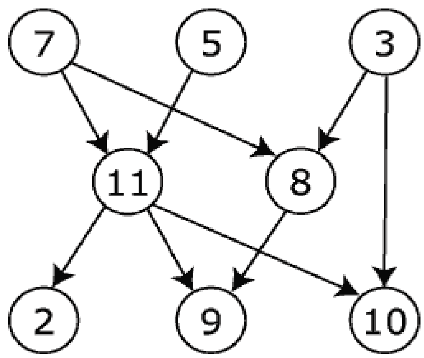
10. Modify Question # 6 to show all possible paths between node A to node F in Graph 1.



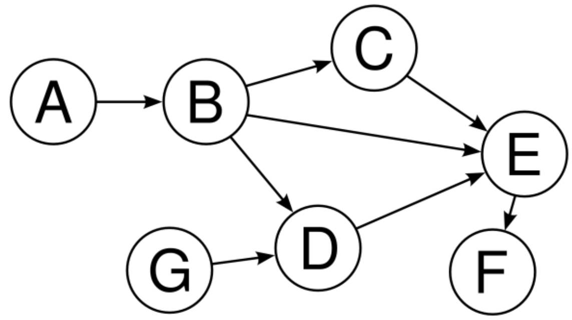
Graph 1



Graph 2



Graph 3



Graph 4

Lab Evaluation:

Q: Suppose you have been given with a following 3x3 3-bit grayscale image. You job is to decompose it into an undirected graph where each pixel within an image represent a node and adjacent nodes are connected to each other via 4-connectivity pattern. Show all possible paths between pixel 150 and pixel 165.
Also write a function to find the shortest path between 150 and 165

150	2	5
80	145	45
74	102	165

Conclusion:
Write the conclusion about this lab

NOTE: A lab journal is expected to be submitted for this lab.