# Complex Networks

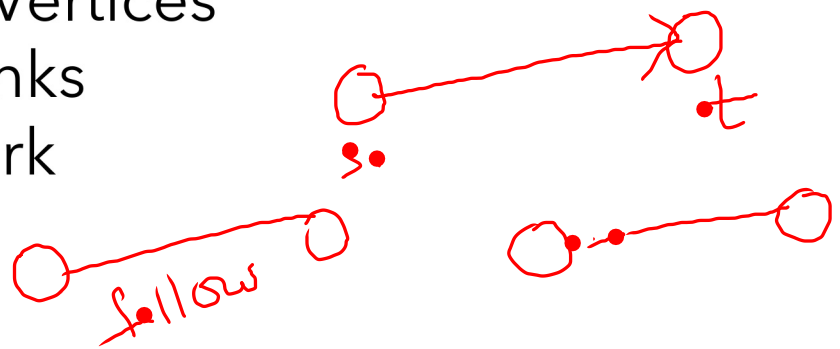
# Objectives

- Network Graphs
- Examples of Networks
- Applications

# Complex Network

**Complex Network:** is a graph or network which represents some real world systems.

- Examples: Social Networks, World Wide Web, Biological Networks.
- A complex system could be represented using:
  - Components: Nodes or Vertices
  - Interactions: Edges or Links
  - System: Graph or Network

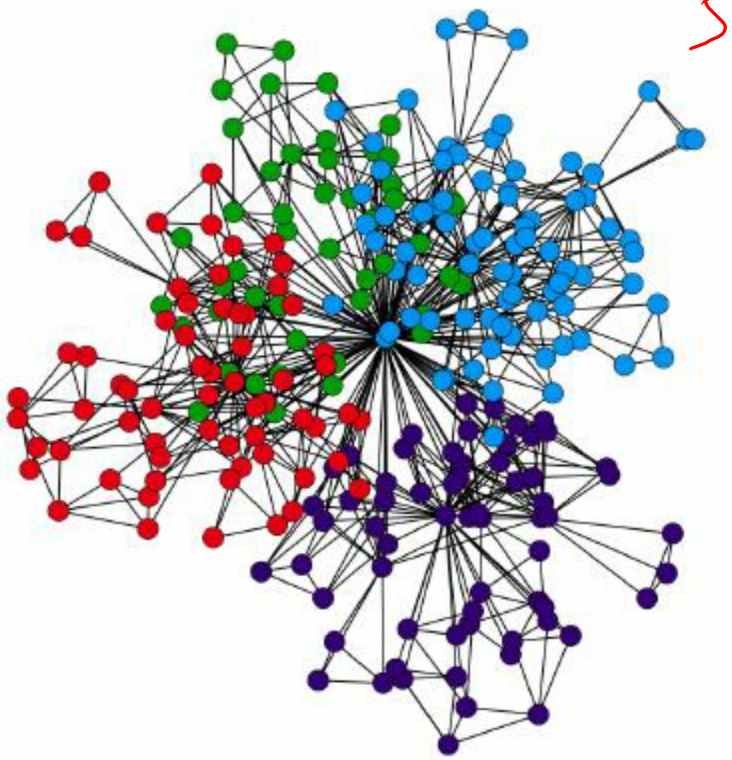


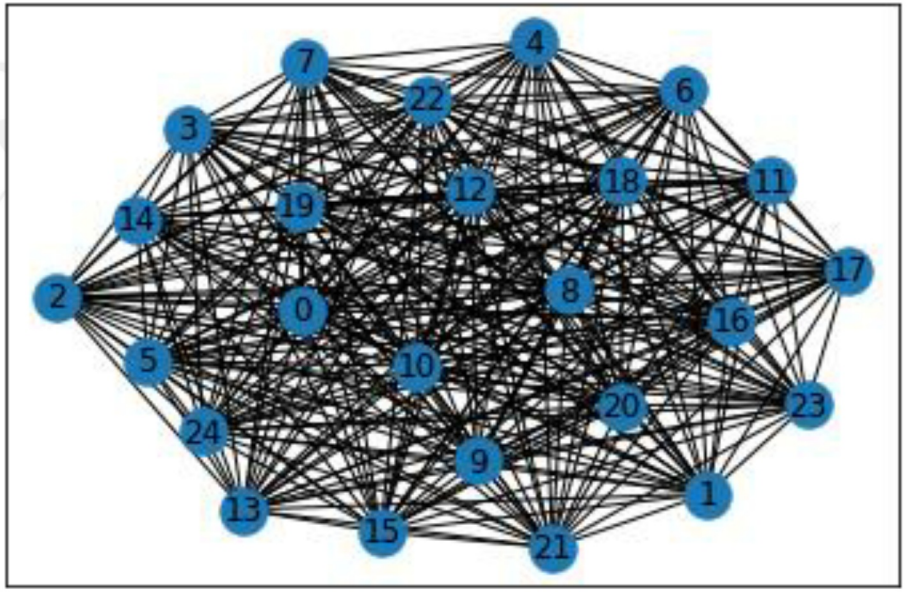
# Network Graphs

**Networks:** A set of objects (nodes) with interconnections (edges).

Dense → V >> : E >>

Sparse → V >> ; E <<

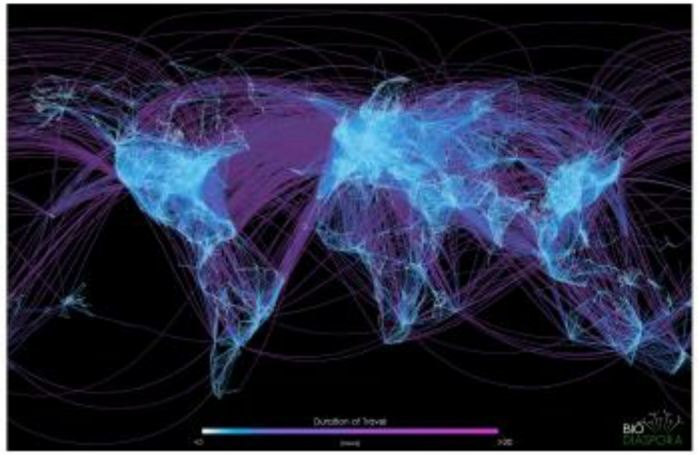




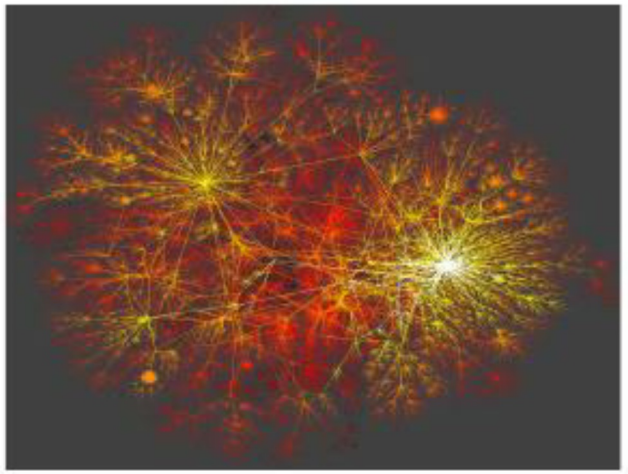
# Networks Examples

E R

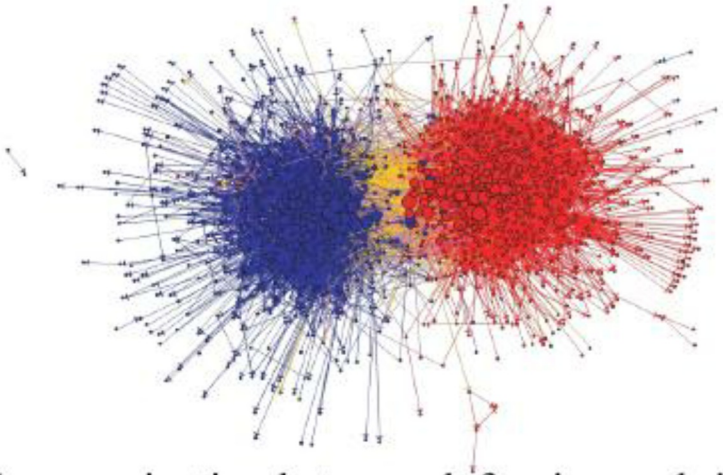
## Networks Everywhere !!!



Network of direct flights around the world [Bio.Diaspora]



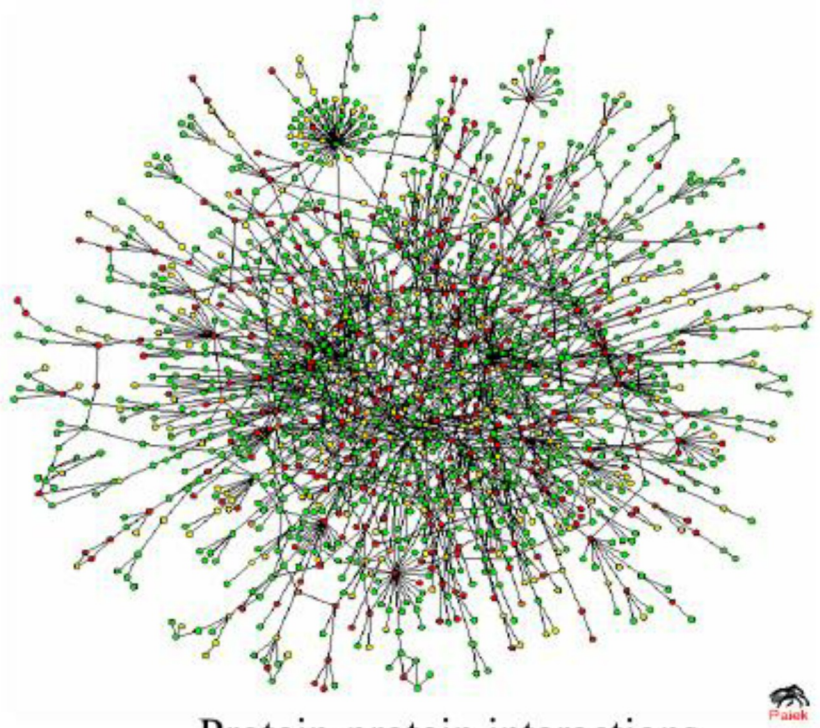
Internet Connectivity [K. C. Claffy ]



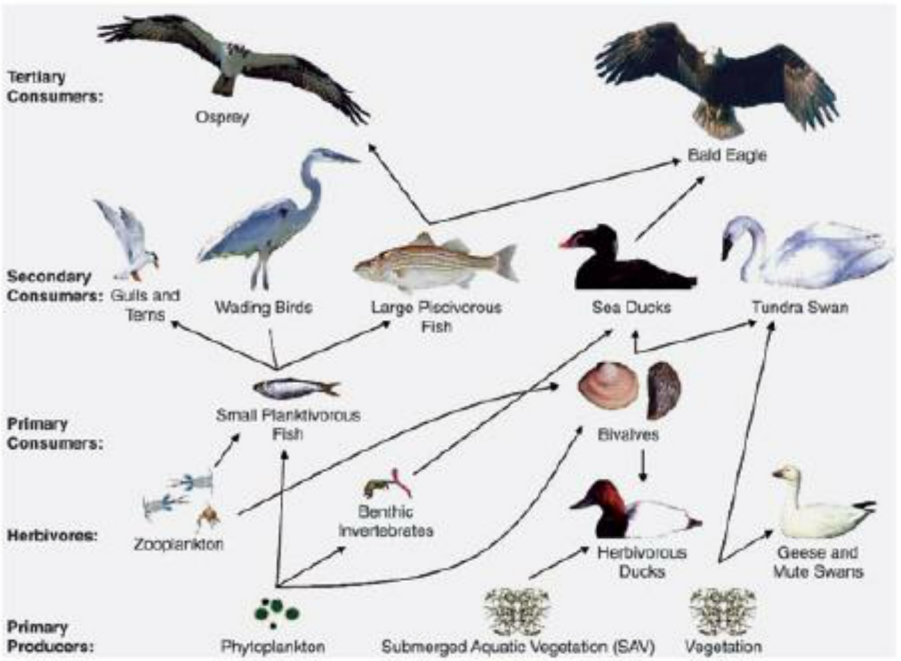
Communication between left-wing and right-wing political blogs [Adamic & Glance 2005]

# Networks Examples ..

Networks Everywhere !!!



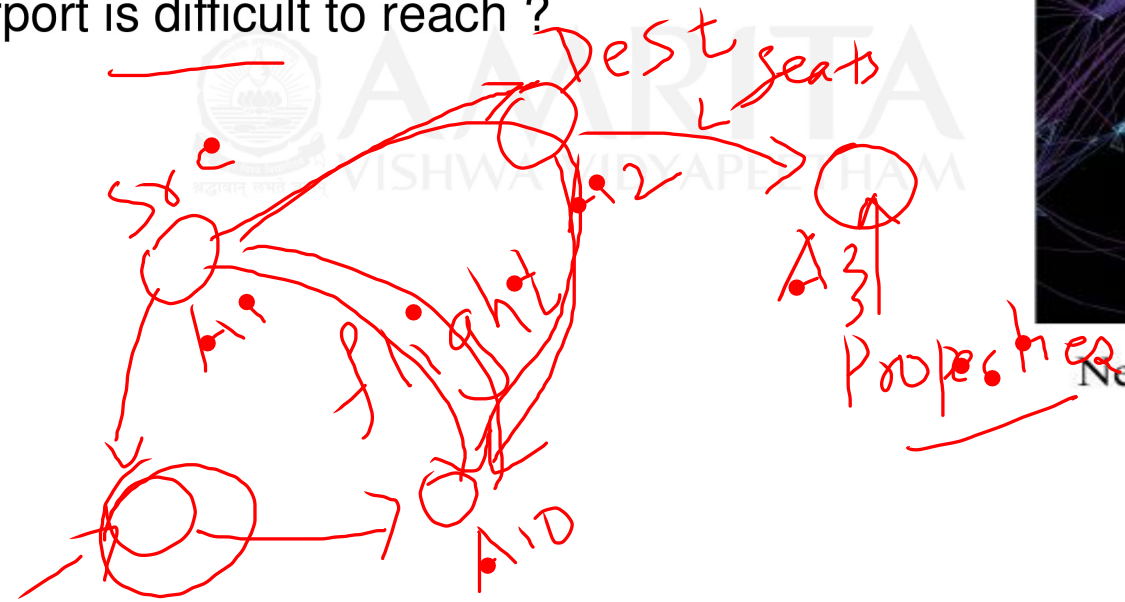
Protein-protein interactions
[Jeong et al. 2001]

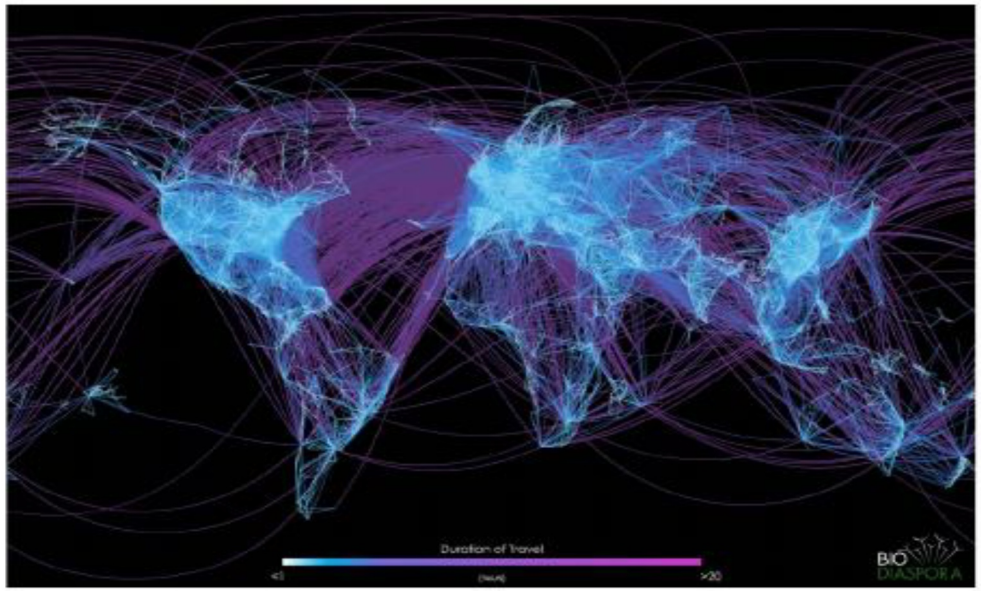



Chesapeake Bay Waterbird Food Web
[Perry et al. 2005]

# Application of Networks

- Map as a graph problem !!!
- Which Airports are at  highest risk Covid ?
- Which Airport is difficult to reach ?

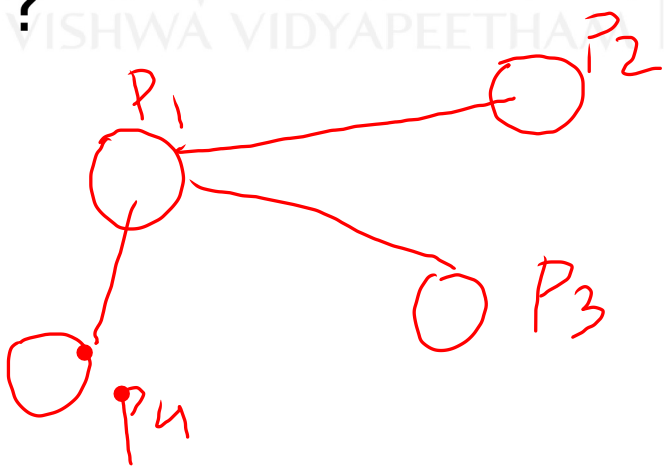


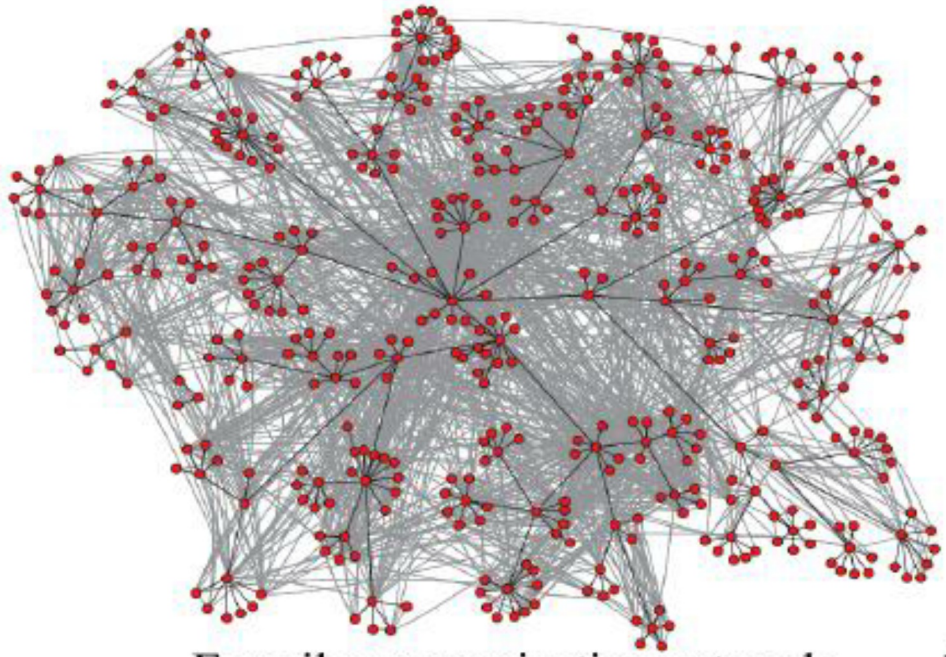




Network of direct flights around the world

# Application of Networks ..

- Graph Problems..
- Who is the most influential person in the company ?

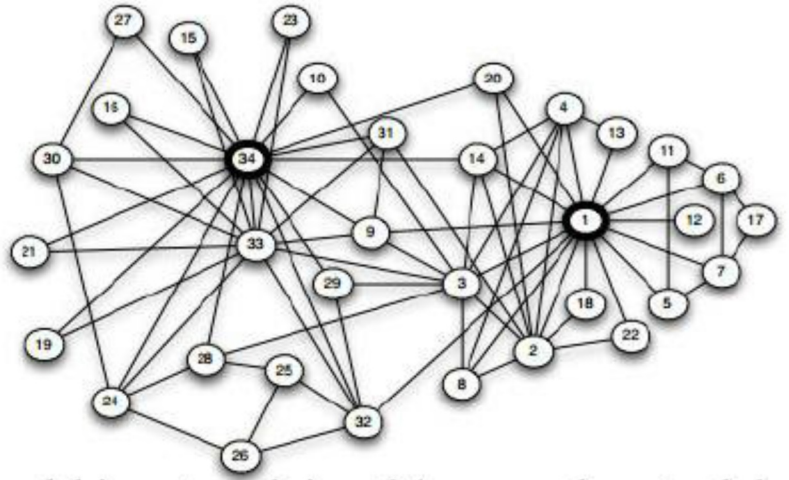


E-mail communication network among 436 HP employees

# Application of Networks ..

- Graph Problems..
- Whether the karate club will split in near future ?



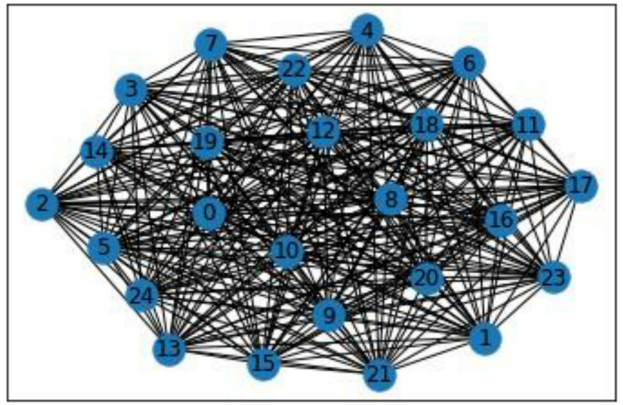

Friendship network in a 34-person karate club

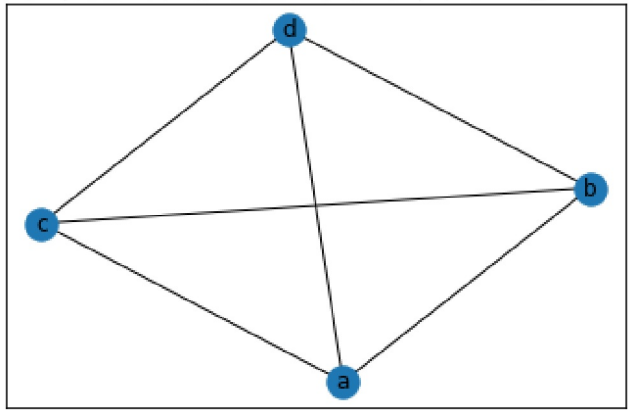
# Undirected Graphs

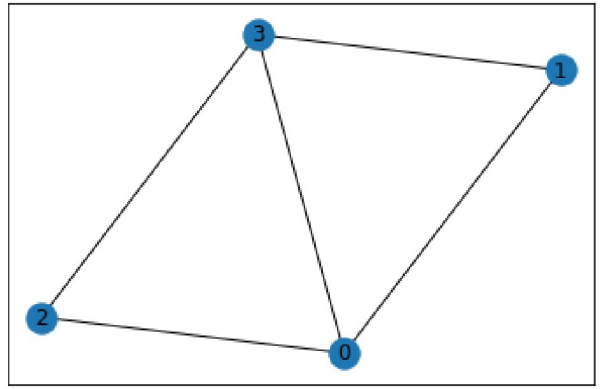
An undirected graph $G = (V, E)$ consists of a set of vertices (or nodes) $V$ together with an edge set of unordered pairs of V's

The elements of $E$ are called edges or links, undirected

The number of elements in V is called the order of G, and we often say G is a graph on V .

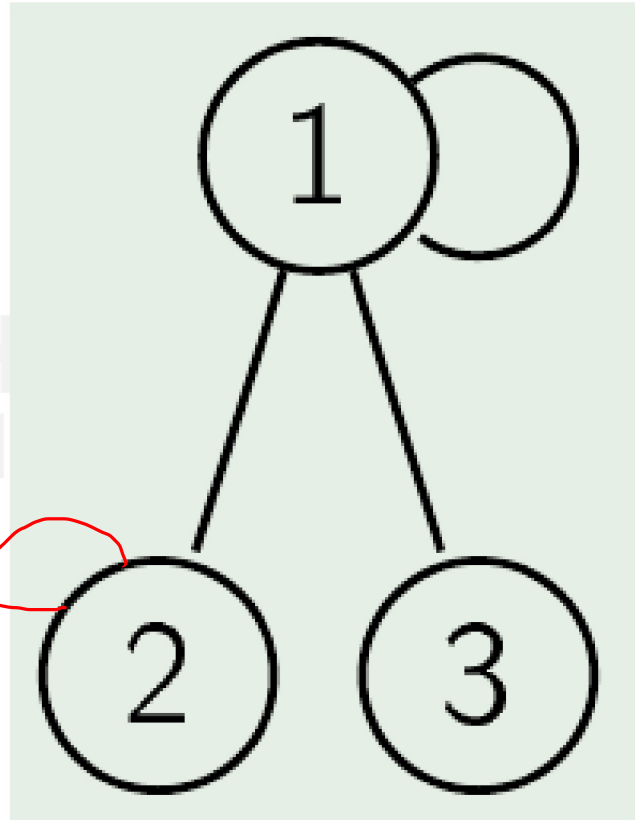


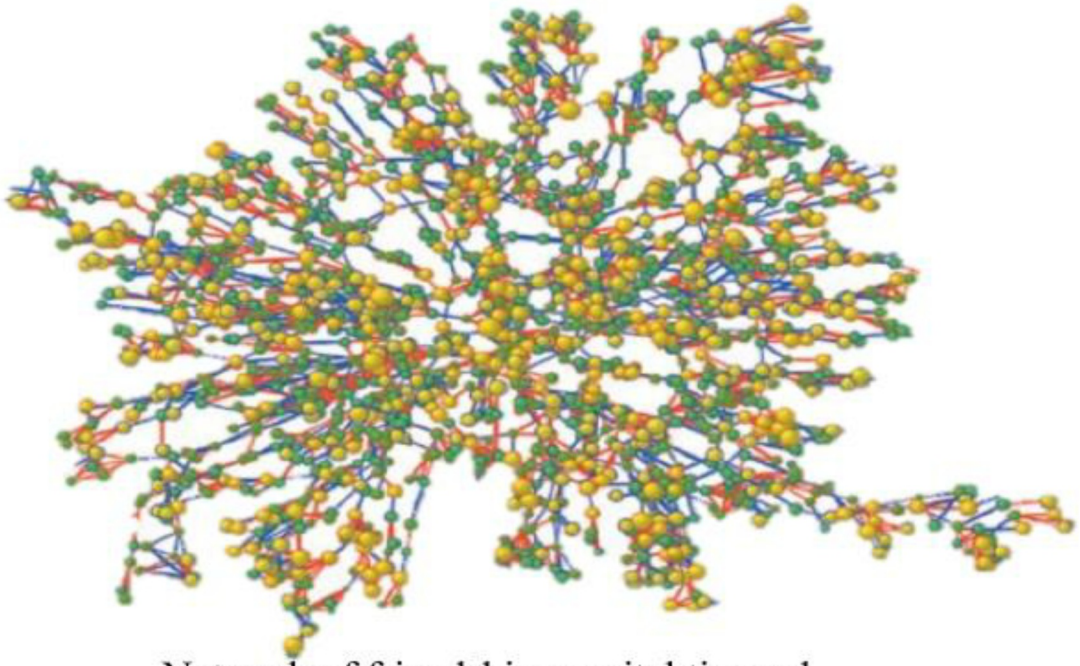
# Example

- Let V = {1, 2, 3}. Then V x V is the the set of all unordered pairs of vertices and E should be a subset of this.
- Take E = [(1,1),(1,2),(1,3)].
- To draw the graph, draw and label each node, and draw a link between two vertices if there is an edge between them.

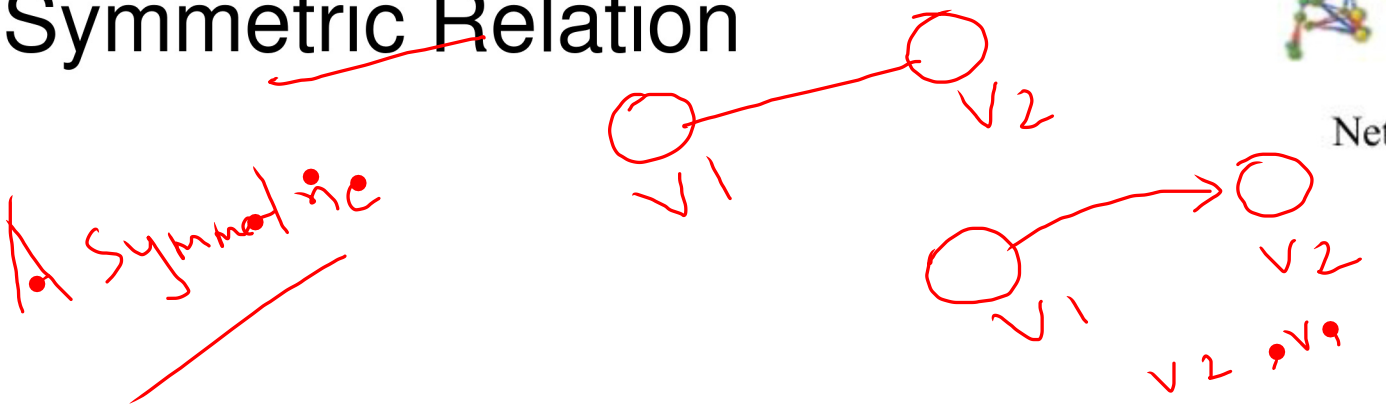


# Real World Example

- Undirected Edges
- Nodes: People
- Edges: Friendship, Marital tie
- Symmetric Relation



Network of friendship, marital tie, and family tie among 2200 people

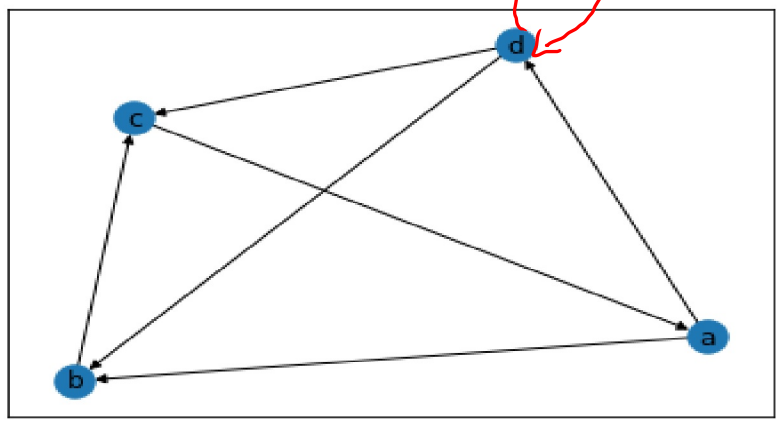


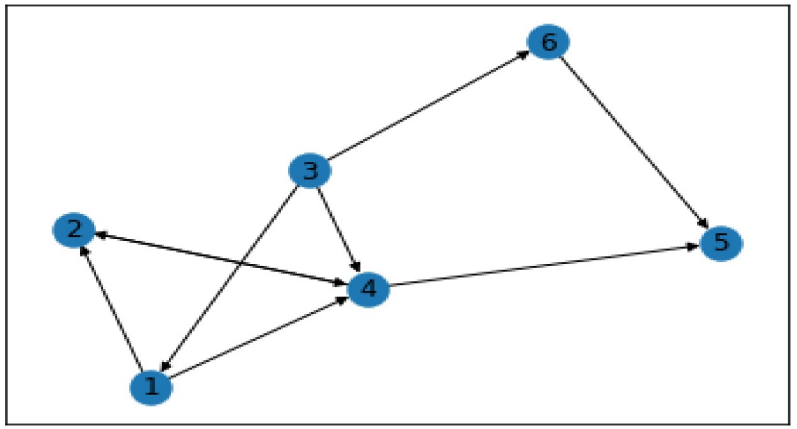
# Directed Graphs

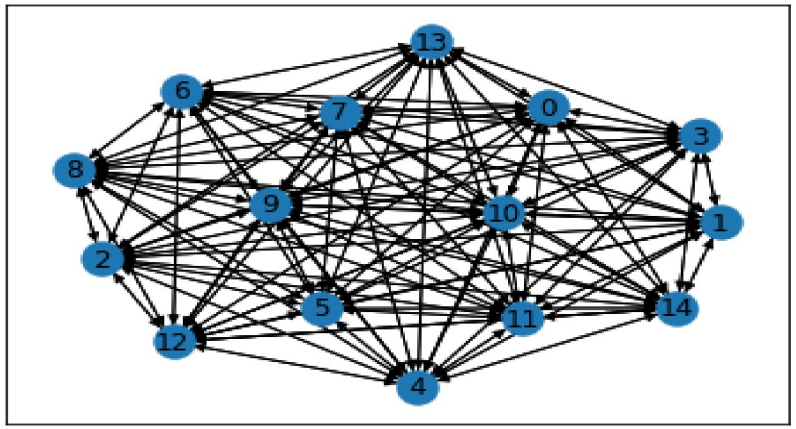
An (directed) graph (or digraph) $G = (V, E)$ consists of a set of vertices (or nodes) $V$ together with an edge set of ordered pairs of V's

The elements of $E$ are called edges or links, **unidirectional**

If $G$ contains no loops, then we say $G$ is simple graph

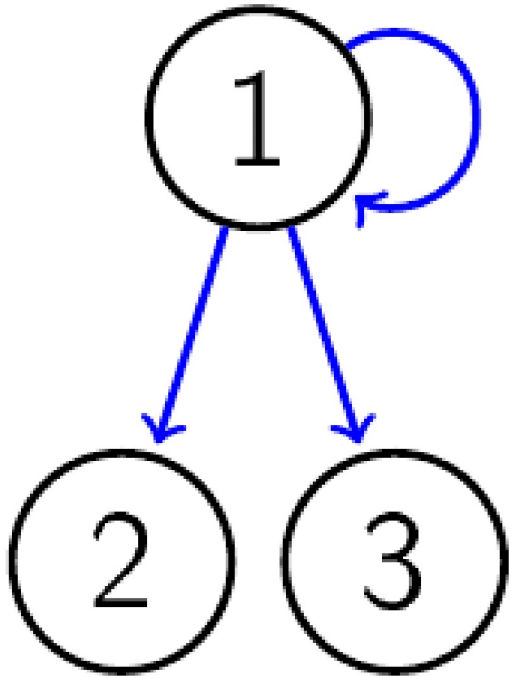


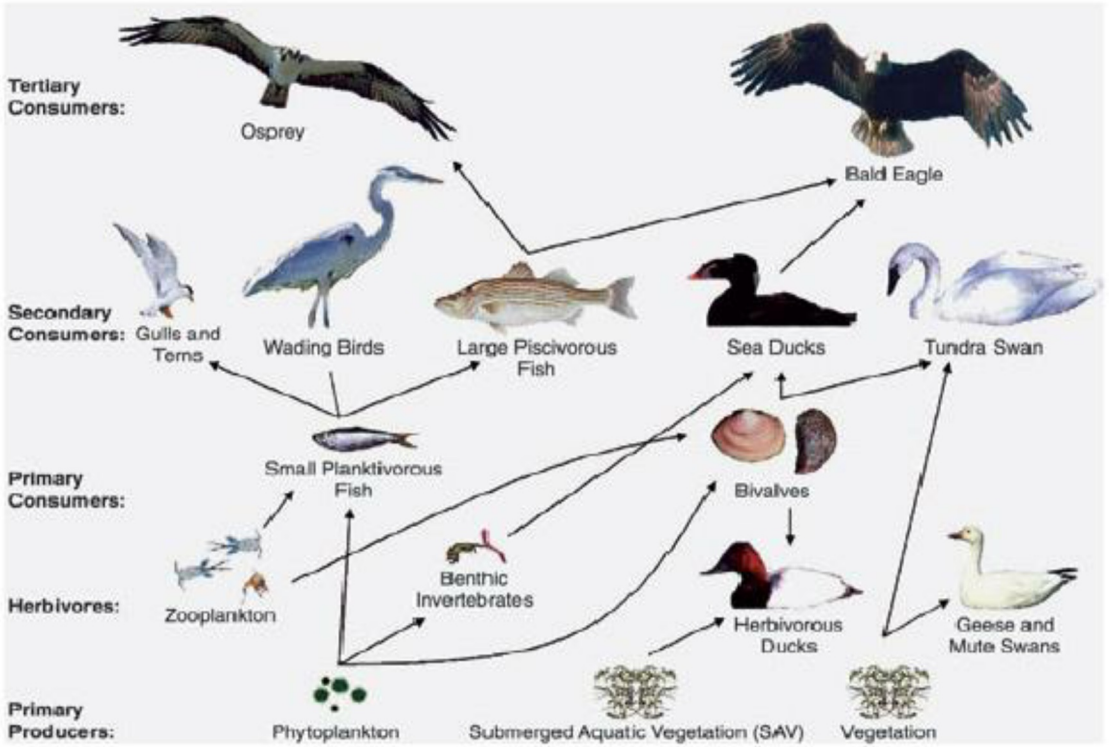
# Example

- Let V = {1, 2, 3}. Then V x V is the the set of all order pairs of vertices and E should be a symmetric subset of this.
- Take E = [(1,1),(1,2),(1,3)].
- To draw the graph, draw and label each node, and draw a link between two vertices if there is an edge between them.



# Real World Example

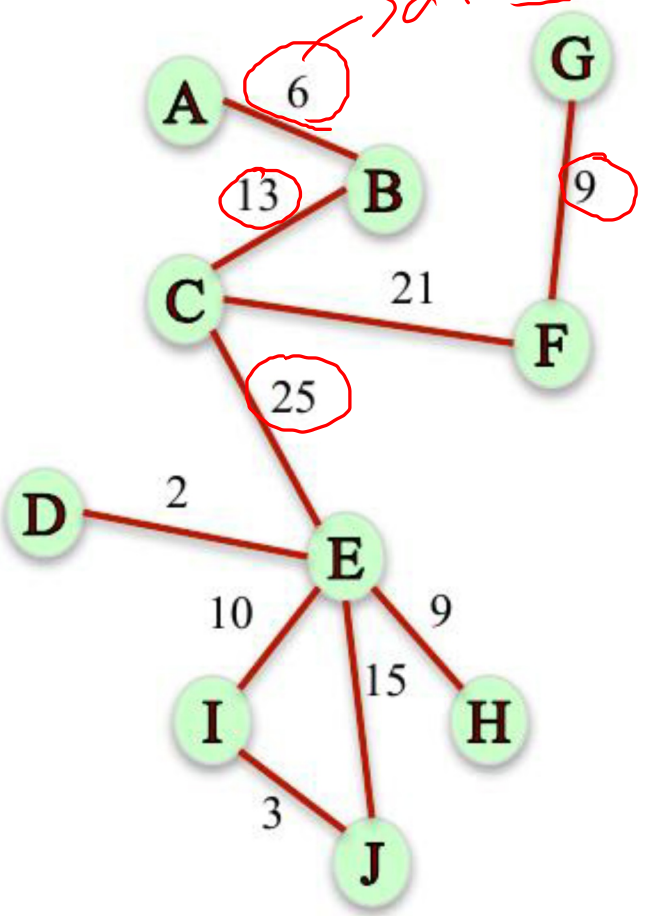
- Directed Edges
- Nodes: Birds
- Edge: What eats what
- Asymmetric Relationships




Chesapeake Bay Water bird Food Web

# Weighted Network

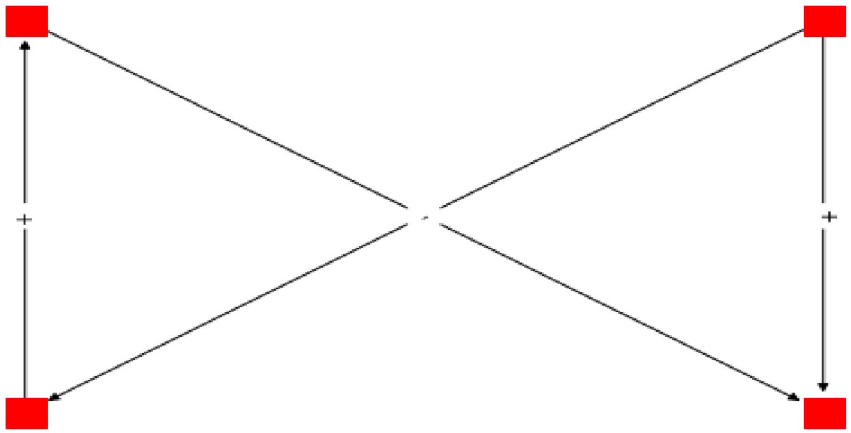
- Edges or links have weights assigned
- When relationships are not equal
- Typically a numerical weight
- Weights:
  - Delay
  - Distance
  - Density

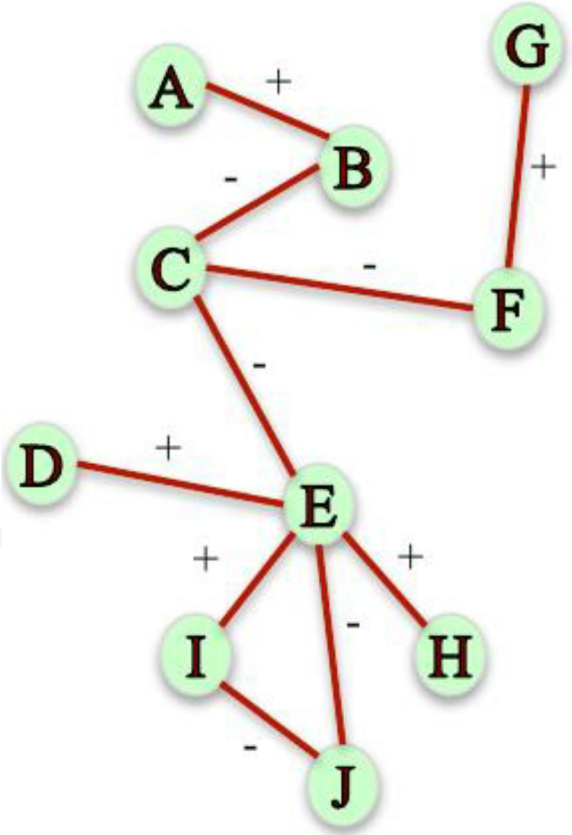



Number of times coworkers had lunch together in one year

# Signed Network

- Edges or links have sign assigned
- Sign could be positive or negative
- Typically to show agreement or conflict

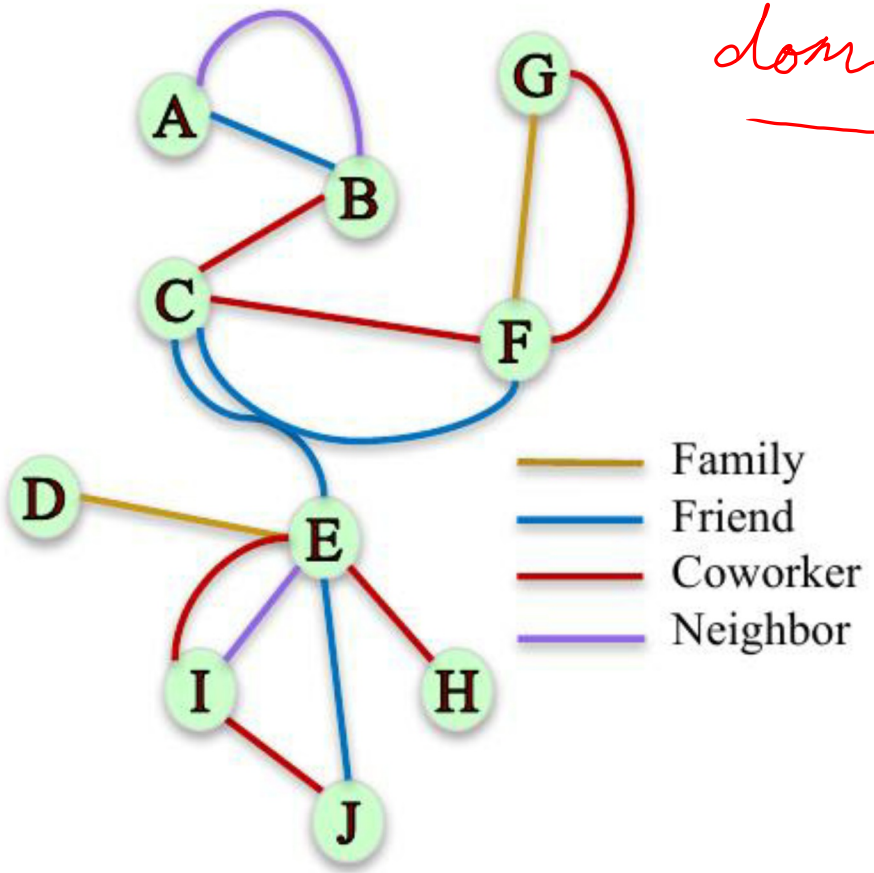





Friends and enemies

# Multiple Edged Graphs

- Nodes can engage in different relations simultaneously
- Two or more edges share a same pair of nodes
- Weights can also assigned to edges




Same domain

# Objectives

- Adjacency Matrix
- Adjacency List
- Edge List
- Incidence Matrix

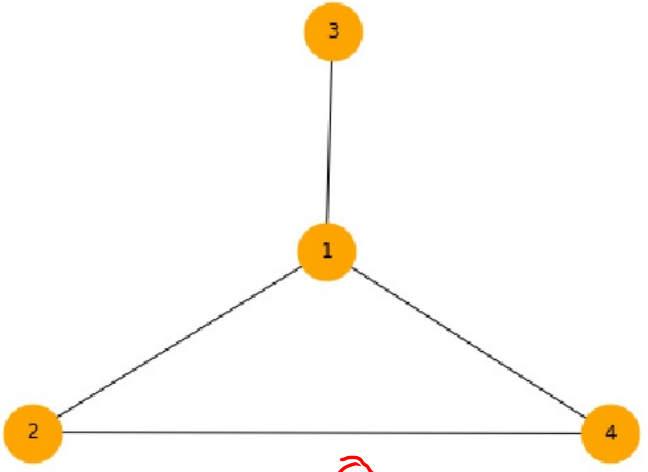

AMRITA VISHWA VIDYAPEETHAM | AHEAD Online

# Adjacency Matrix

## Undirected Graph

- Matrix of size $|V| \times |V|$
- $A_{ij} = \begin{pmatrix} a_{11} & \dots & \dots & a_{1n} \\ a_{21} & \dots & \dots & a_{2n} \\ a_{31} & \dots & \dots & a_{3n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & \dots & \dots & a_{nn} \end{pmatrix}$
- $A_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$

## Example




$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

- Note: $A = A^T$

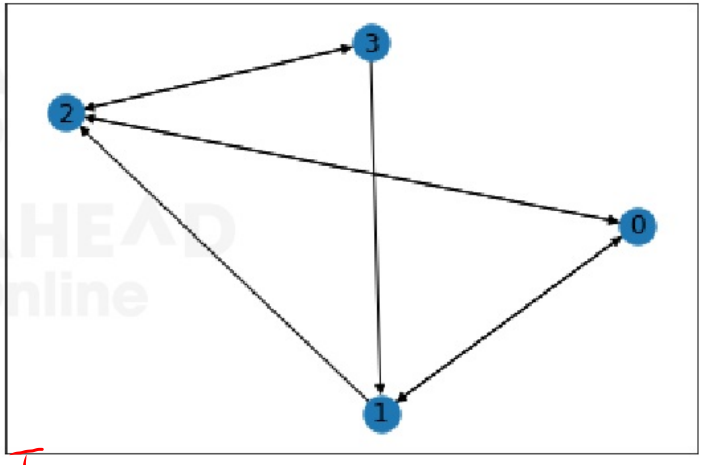
# Adjacency Matrix ..

## Directed Graph

- Matrix of size $|V| \times |V|$
- $A_{ij} = \begin{pmatrix} a_{11} & \dots & \dots & a_{1n} \\ a_{21} & \dots & \dots & a_{2n} \\ a_{31} & \dots & \dots & a_{3n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & \dots & \dots & a_{nn} \end{pmatrix}$
- $A_{ij} = \begin{cases} 1, & \text{if a link } i \rightarrow j \in E \\ 0, & \text{otherwise} \end{cases}$

### Example

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$




$A \neq A^T$

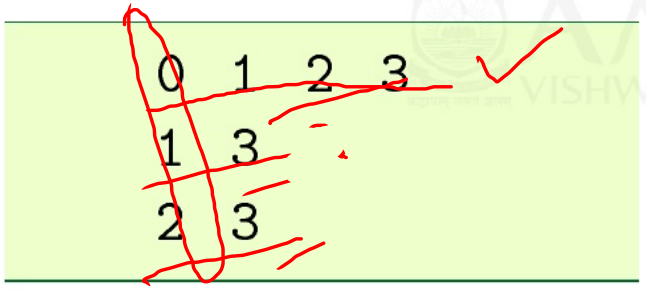
# Adjacency List

- An array of lists is used.
- The size of the array is equal to the number of vertices.

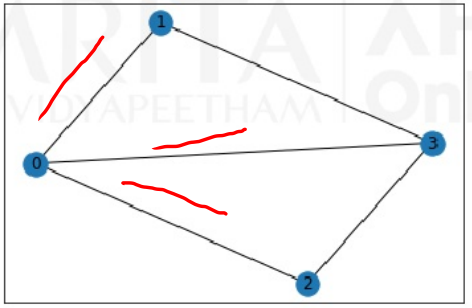
## Syntax

- Let $V = \{a, b, c, d, e\}$
- a b c d e;node a is adjacent to nodes b, c, d, e
- b c e; node b is (also) adjacent to nodes c, e
- e; node e is (also) adjacent to no new nodes

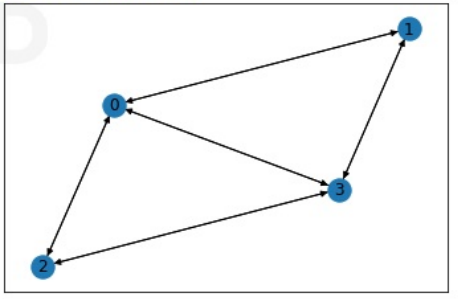
## Example




Undirected




Directed




Note the direction of links

# Edge List

- A list of edges (or an array) is used
- Simple representation

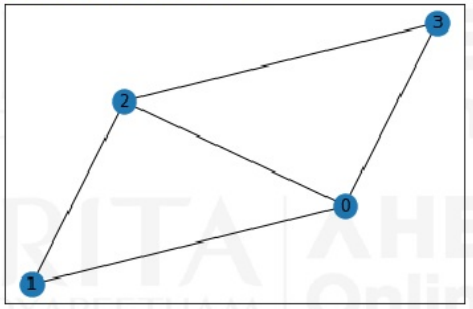
## Syntax

- Let $V = \{a, b, c, d\}$
- $E$=[(a,b), (b,c),(a,d)]

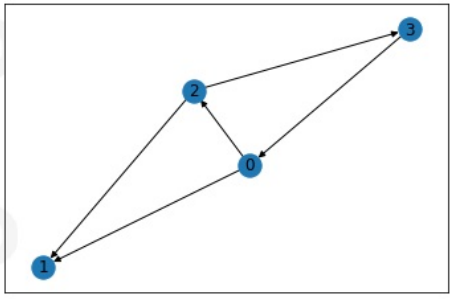
## Example

```
[(0,1),(2,3),(3,0),
(0, 2),(2, 1)]
```

Undirected




Directed



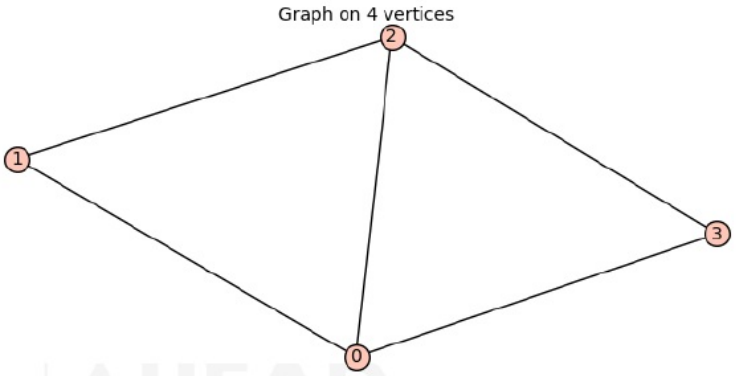

Note the direction of edges

# Incidence Matrix

- Each row represents a vertex and each column an edge:

## Undirected Graph

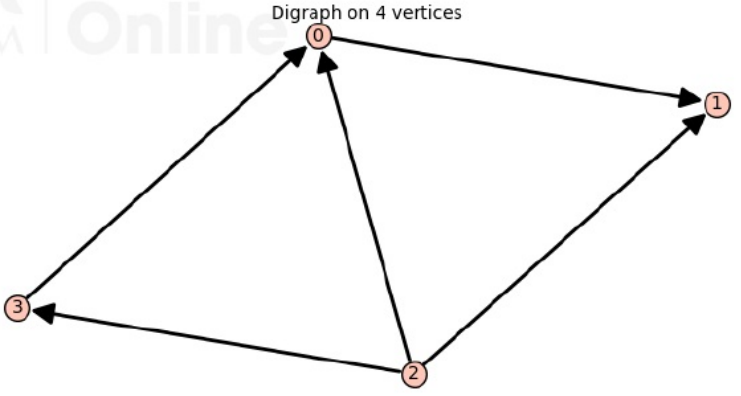
$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$




## Directed Graph

- Negative sign for out-links

$$\begin{pmatrix} -1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & -1 & 1 & 0 & 0 \end{pmatrix}$$



# Summary

## Graph Representations

- Adjacency Matrix
- Adjacency List
- Edge List
- Incidence Matrix


AMRITA VISHWA VIDYAPEETHAM | AHEAD Online

# Python

# Objectives

- Python World
- Programming in Python
- Basic Programs


AMRITA VISHWA VIDYAPEETHAM | AHEAD Online

# For our course

## Python Basics

- What is Python
- Python Language Basics
  - Syntax
  - Types, Operators
  - Control Flow, Functions
  - Classes
  - Tools

## Python Advanced

- More on Python
  - Modules
  - Standard Library
  - Virtual Environments and Packages
- Enhanced Consoles
- Anaconda
- Jupyter Notebook

# What is Python

## Python

- Authored by Guido van Rossum
- Multi-purpose (Web, GUI, Scripting, etc.)
- Object Oriented
- Interpreted
- Strongly typed and Dynamically typed
- Focus on readability and productivity

# Features of Python

## Language Features

- "very high-level language", with:
  - clean, spare syntax
- simple, regular, powerful semantics
- object-oriented, multi-paradigm
- focus on productivity via modularity, item uniformity, simplicity, pragmatism
- a rich standard library of modules
- lots of 3rd-party tools/add-ons
- several good implementations
  - CPython 2.5, pypy 1.0, IronPython 1.1 [.NET], (Jython 2.2 [JVM])

# Popularity & Releases

## Top Users

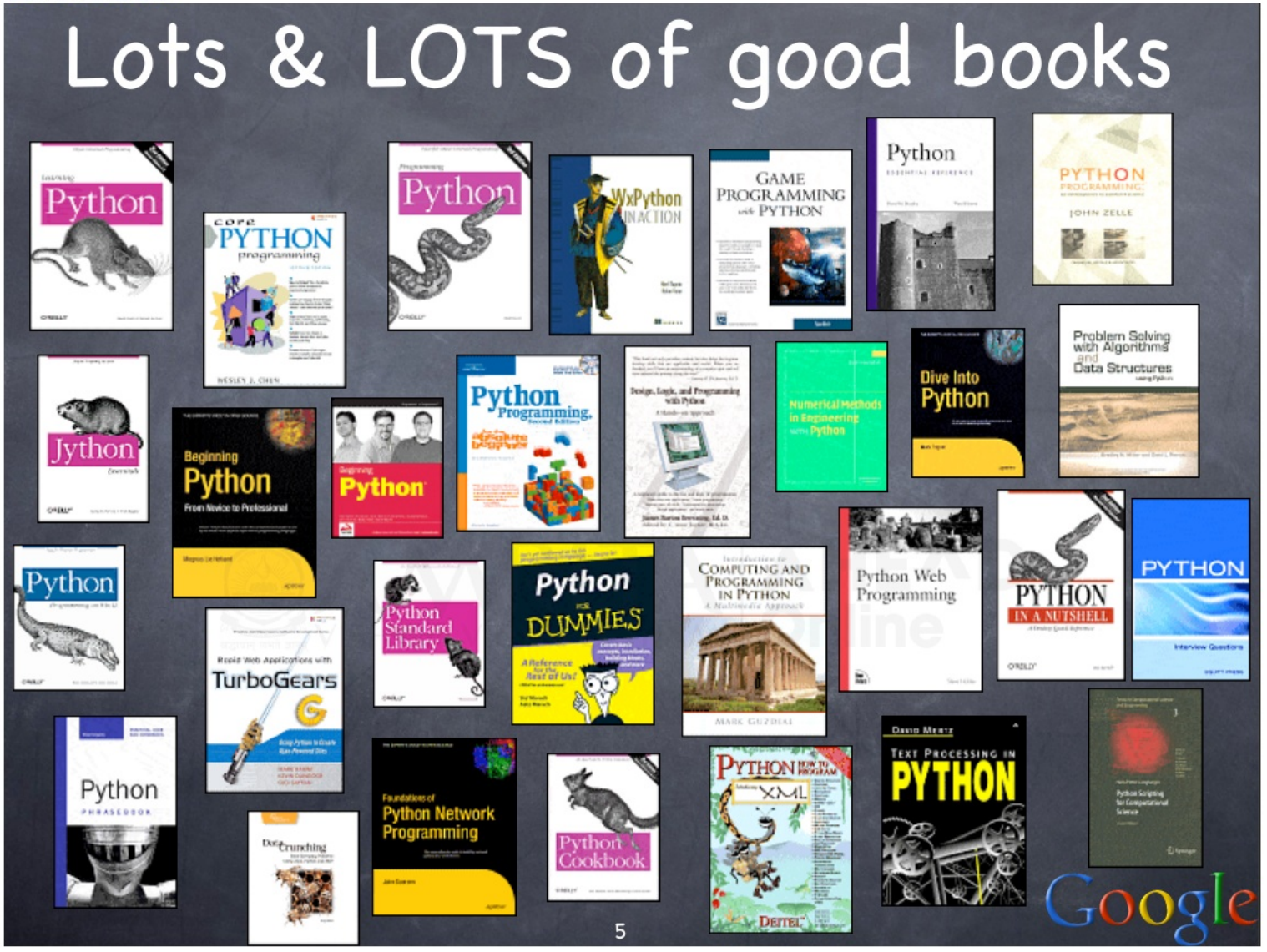
- Google
- National Weather Service
- NASA
- Library of Congress
- RedHat
- Walt Disney Feature Animation
- ...the list goes on...

## Releases

- Created in 1989 by Guido Van Rossum
- Python 1.0 released in 1994
- Python 2.0 released in 2000
- Python 3.0 released in 2008
- Python 2.7 released in 2010
- Python 3.10 latest

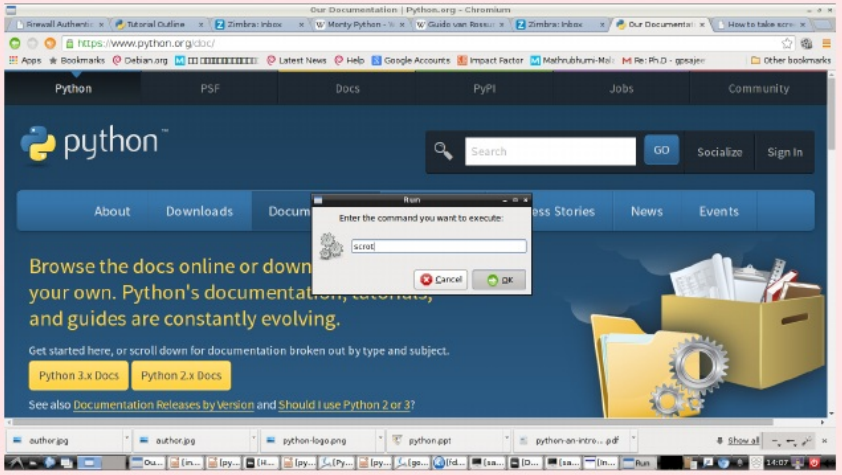
# Pleanty of Resources

## Books



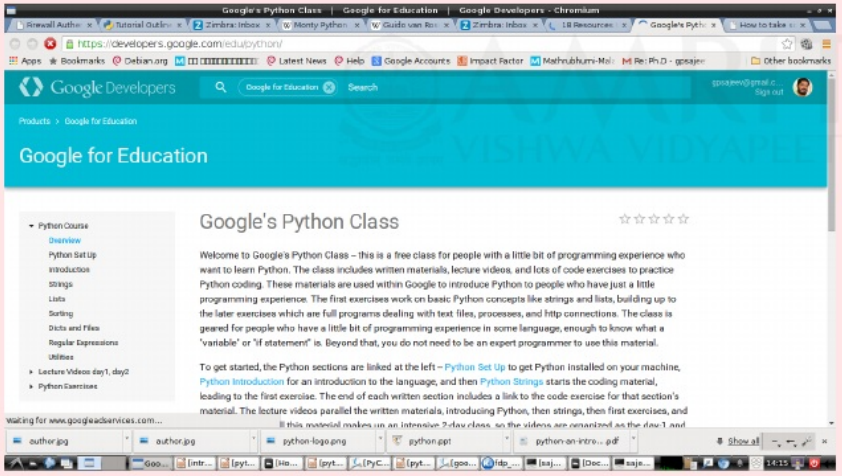
# Pleanty of Resources ...

## python.org

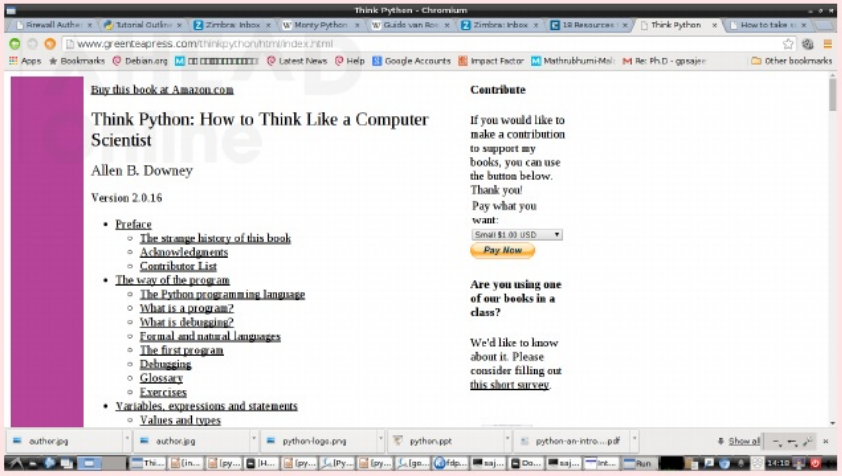



## Edgewell


## Google Python Classes

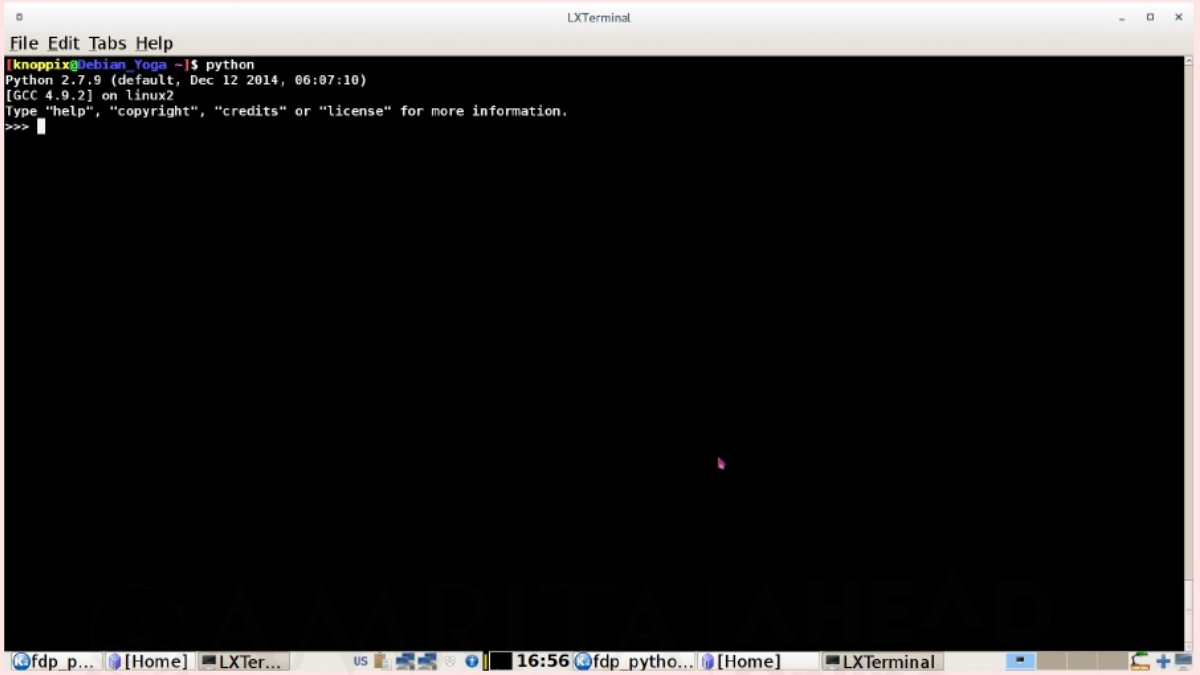



## Think Python



# Python Environment

## Python Shell

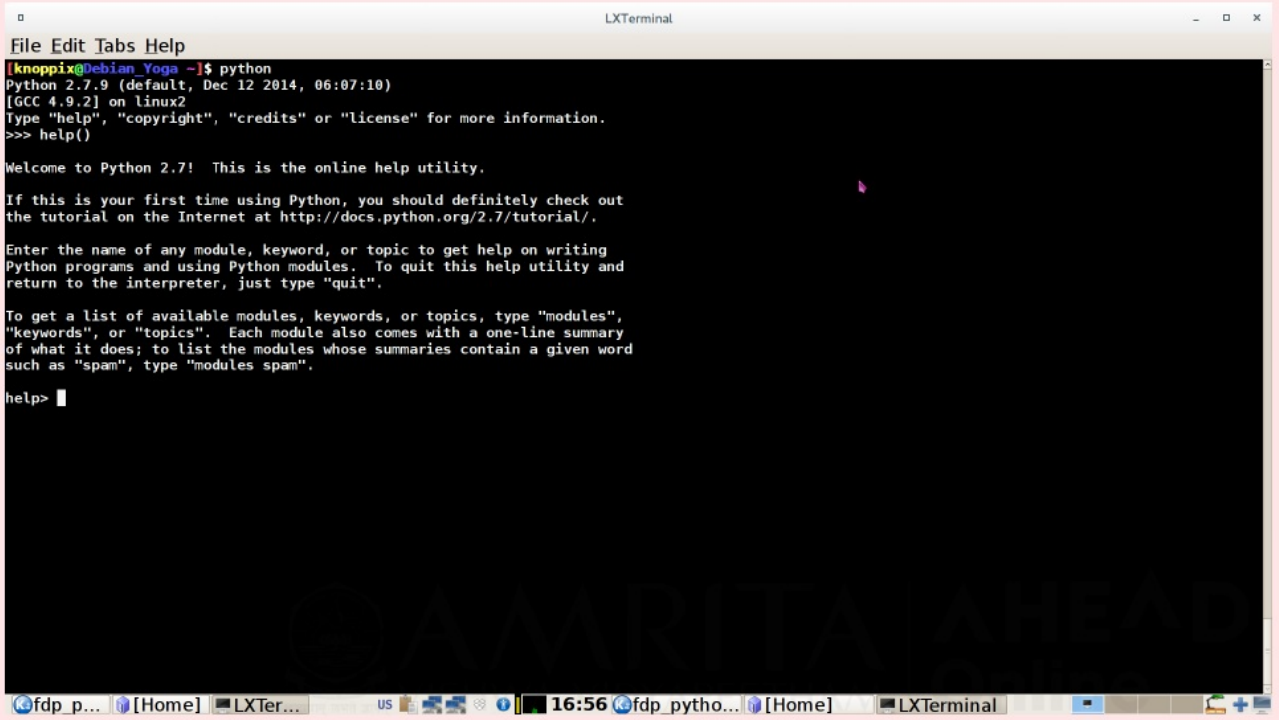



## Usage

- Type 'python' on a terminal
  - for entering the python shell
- Type 'quit()' from python shell
  - for exiting

# Python Help & Documentation

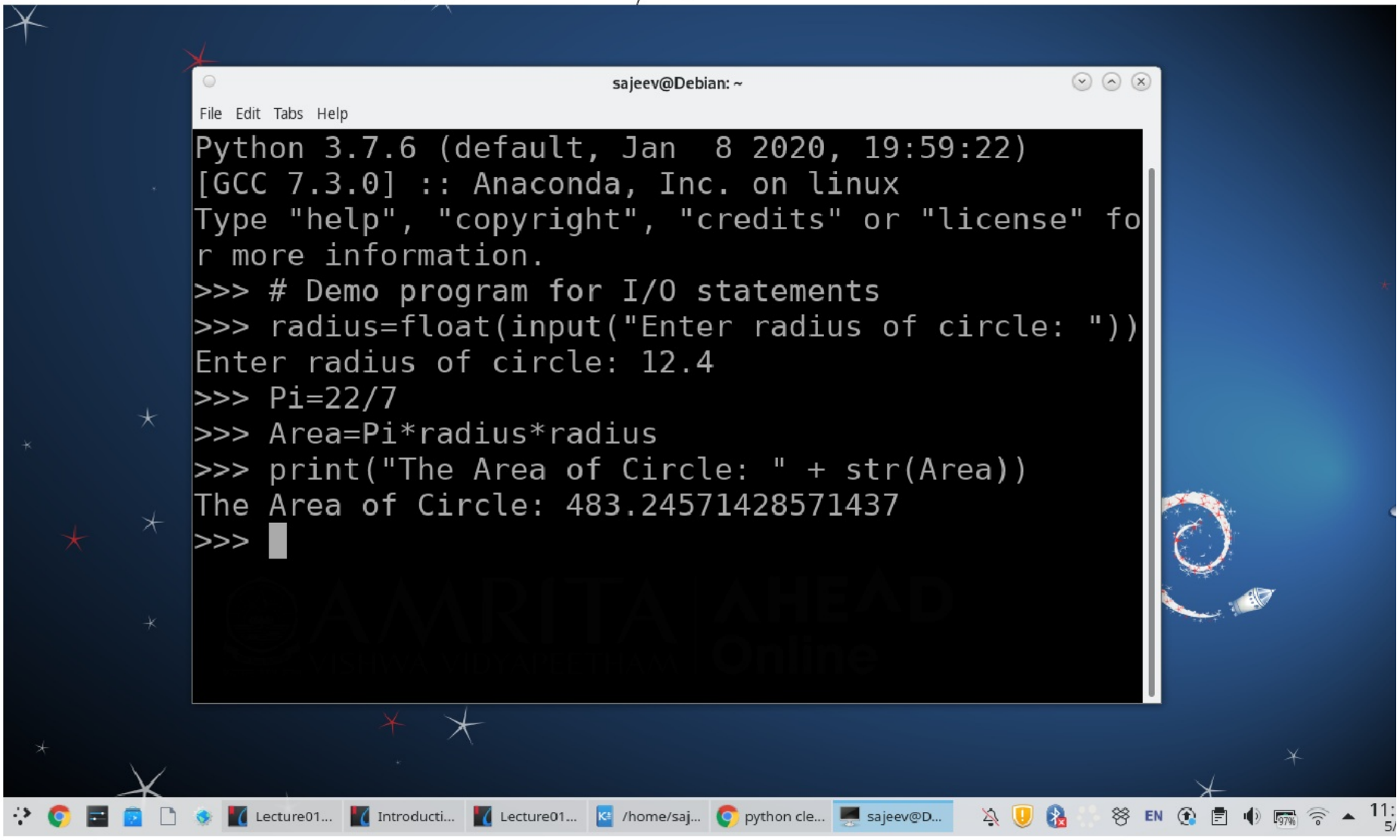
## Python help




## Usage

- Type 'help()' on a terminal
  - for entering the help mode
- Type 'q' from help system
  - for exiting

# Using Shell: I/O statements



# Basic Programs

```
# To check whether the number is even or odd
num = input("Enter a number: ")
mod = num % 2
if mod > 0:
    print("You picked an odd number.")
else:
    print("You picked an even number.")
```

```
# prints out all the elements of the list
# that are less than a num.
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
num = int(raw_input("Choose a number: "))
new_list = []
for i in a:
        if i < num:
            new_list.append(i)
print new_list
```

# Summary

## Python Language

- Introduction to Python
- Using Python Shell
- Basic Programs


AMRITA VISHWA VIDYAPEETHAM | AHEAD Online

# NetworkX

# Objectives

- *NetworkX*: classes for graph objects, generators to create standard graphs.
- Reading existing datasets
- Basic drawing tools.


AMRITA VISHWA VIDYAPEETHAM | AHEAD Online

# Basic Example

```
>>> import networkx as nx
>>> G = nx.Graph()
>>> G.add_node("spam")
>>> G.add_edge(1,2)
>>> print(G.nodes())
[1, 2, 'spam']
>>> print(G.edges())
[(1, 2)]
```

# Graph Types

- Graph : Undirected simple (allows self loops)
- DiGraph : Directed simple (allows self loops)
- MultiGraph : Undirected with parallel edges
- MultiDiGraph : Directed with parallel edges
- can convert to undirected: `g.to_undirected()`
- can convert to directed: `g.to_directed()`

To construct, use standard python syntax:

```
>>> g = nx.Graph()
>>> d = nx.DiGraph()
>>> m = nx.MultiGraph()
>>> h = nx.MultiDiGraph()
```

# Adding Nodes

- `add_nodes_from()` takes any iterable collection and any object

```
>>> g = nx.Graph()
>>> g.add_node('a')
>>> g.add_nodes_from( ['b','c','d'])
>>> g.add_nodes_from('xyz')
>>> h = nx.path_graph(5)
>>> g.add_nodes_from(h)
>>> g.nodes()
[0,1,'c','b',4,'d',2,3,5,'x','y','z']
```

# Adding Edges

- Adding an edge between nodes that don't exist will automatically add those nodes
- `add_nodes_from()` takes any iterable collection and any type (anything that has a `__iter__()` method)

```
>>> g = nx.Graph( [('a','b'),('b','c'),('c','a')] )
>>> g.add_edge('a', 'd')
>>> g.add_edges_from([('d', 'c'), ('d', 'b')])
```

# Node attributes

- Can add node attributes as optional arguments along with most add methods

```
>>> g = nx.Graph()
>>> g.add_node(1,name='Obrian')
>>> g.add_nodes_from([2],name='
    Quintana'])
>>> g[1]['name']
'Obrian'
```

# Edge attributes

- Can add edge attributes as optional arguments along with most add methods

```
>>> g.add_edge(1, 2, w=4.7 )
>>> g.add_edges_from([(3,4),(4,5)], w =3.0)
>>> g.add_edges_from([(1,2,{'val':2.0})])
# adds third value in tuple as 'weight' attr

>>> g.add_weighted_edges_from([(6,7,3.0)])
>>> g.get_edge_data(3,4){'w' : 3.0}
>>> g.add_edge(5,6)
```

# Simple Properties

- Number of nodes :

```
c len(g)
>>> g.number_of_nodes()
>>> g.order()
```

- Number of Edges

```
>>> g.number_of_edges()
```

- Check node membership

```
>>> g.has_node(1)
```

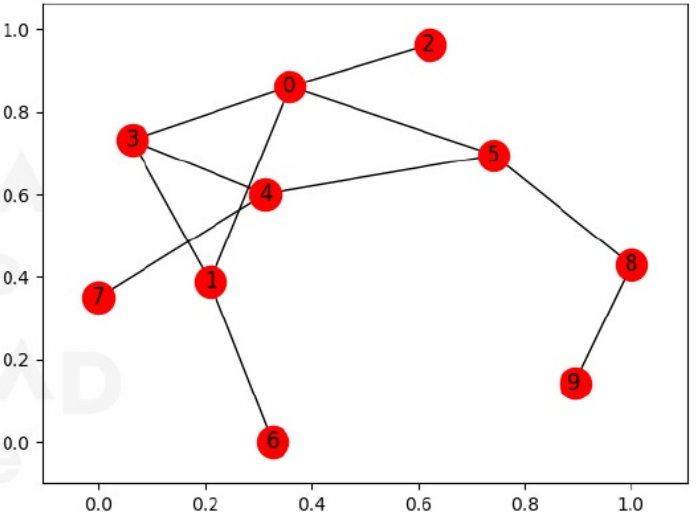
- Check edge presence

```
>>> g.has_edge(1)
```

# Loading & Plotting Graphs

- Adjacency List

```
import networkx as nx
G1 = nx.Graph()
# add node/edge pairs
G1.add_edges_from([(0, 1),
(0,2),(0,3),(0,5),(1,3),(1,6),
(3,4),(4,5),(4,7),(5,8),(8,9)
    ])
# draw the network G1
nx.draw_networkx(G1)
```



# Loading &Plotting Graphs ..

- Create a file for Adjacency List
- G_adjlist.txt is the adjaceny list representation of G1.

```
0 1 2 3 5; node 0 is adjacent to nodes 1, 2, 3, 5
1 3 6; node 1 is (also) adjacent to nodes 3, 6
2;  node 2 is (also) adjacent to no new nodes
3 4;node 3 is (also) adjacent to node 4
```

```
>>> !cat G_adjlist.txt
>>>G2 = nx.read_adjlist('G1.adjlist')
>>>G3=nx.Graph(G2)
>>>nx.draw_networkx(G2)
```

# Loading &Plotting Graphs ..

- Adjacency Matrix

```
import numpy as np
 G_mat =
 np.array([[0, 1, 1, 1, 0, 1, 0, 0, 0, 0],
           [1, 0, 0, 1, 0, 0, 1, 0, 0, 0],
           [1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
           [1, 1, 0, 0, 1, 0, 0, 0, 0, 0],
           [0, 0, 0, 1, 0, 1, 0, 1, 0, 0],
           [1, 0, 0, 0, 1, 0, 0, 0, 1, 0],
           [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
           [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
           [0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
           [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]])
G3 = nx.Graph(G_mat)
```

# Summary

- Creating Graphs
- Adding attributes
- Plotting graphs


AMRITA VISHWA VIDYAPEETHAM | AHEAD Online

# References


Hagberg, A., Swart, P., and S Chult, D.
Exploring network structure, dynamics, and function using networkx.
Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.