Operator Functions Set 1 & Set 2
Data Types
Introduction
Arrays Set 1, Set 2
String Methods Set 1, Set 2, Set 3
String Template Class & String Formatting using %
List Methods Set 1, Set 2, Set 3
Tuples & Sets
Dictionary Methods Set 1, Set 2
ChainMap
Explore More...
Control Flow
Loops and Control Statements
Counters & Accessing Counters
Iterators & Iterator Functions Set 1, Set 2
Generators
Explore More...
Functions
Function Decorators
Returning Multiple Values
Yield instead of Return
Python Closures & Coroutine
Explore More...
Modules
Introduction

Numeric Functions & Logarithmic and Power functions
Calender Functions Set 1, Set 2
Complex Numbers Introduction & Important functions
Explore More...
Object Oriented Concepts
Class, Object and Members
Data Hiding and Object Printing
Inheritance, Subclass and super
Class method vs static method & Class or Static Variables
Explore More...
Exception Handling
Exception Handling
User-Defined Exceptions
Built-in Exceptions
Libraries and Functions
Timeit
Numpy Set 1, Set 2
Get and Post
import module & reload module
Collection Modules Deque, Namedtuple & Heap
Explore More...
Machine Learning with Python
Classifying data using Support Vector Machines(SVMs) in Python

K means Clustering
How to get synonyms/antonyms from NLTK WordNet in Python?
Explore More...
Misc
Sql using Python & MongoDB and Python
Json formatting & Python Virtual environment
Metaprogramming with Metaclasses in Python
Python Input Methods for Competitive Programming
Explore More...
Applications and Projects
Creating a proxy webserver Set 1, Set 2
Send Messsage to FB friend
Twitter Sentiment Analysis & Whatsapp using Python
Desktop Notifier & Junk File Organizer
Explore More...

NetworkX : Python software package for study of complex networks

NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and function of complex networks. It is used to study large complex networks represented in form of graphs with nodes and edges. Using networkx we can load and store complex networks. We can generate many types of random and classic networks, analyze network structure, build network models, design new network algorithms and draw networks.

Installation of the package:

```
pip install networkx
```

Creating Nodes

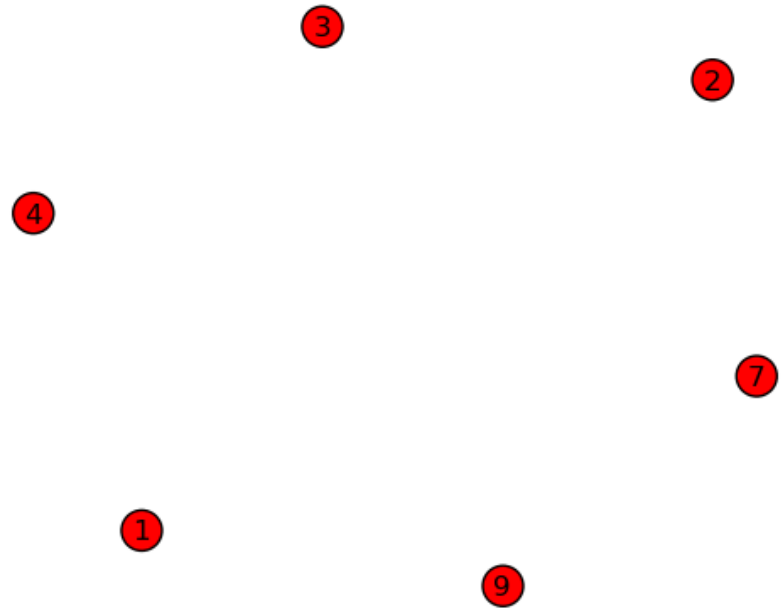
Add one node at a time:

```
G.add_node(1)
```

Add a list of nodes:

```
G.add_nodes_from([2,3])
```

Let us create nodes in the graph G. After adding nodes 1, 2, 3, 4, 7, 9



Creating Edges

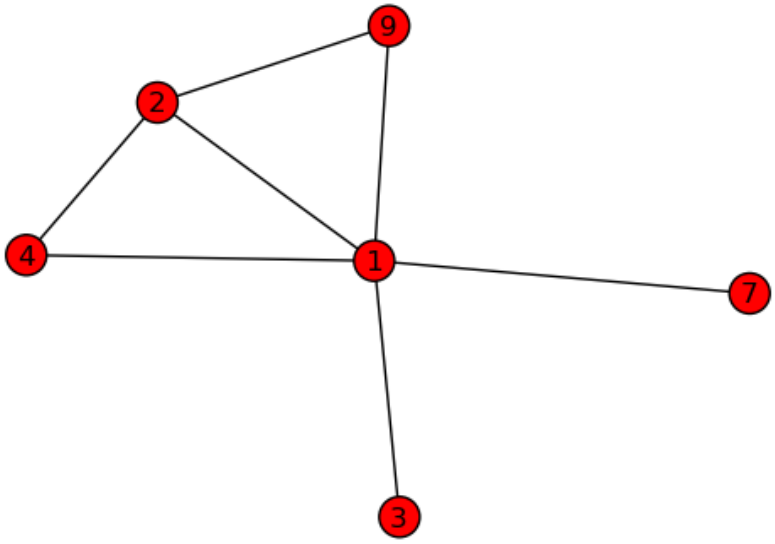
Adding one edge at a time:

```
G.add_edge(1,2)
G.add_edge(3,1)
G.add_edge(2,4)
G.add_edge(4,1)
G.add_edge(9,1)
```

Adding a list of edges:

```
G.add_edges_from([(1,2),(1,3)])
```

After adding edges (1,2), (3,1), (2,4), (4,1), (9,1), (1,7), (2,9)

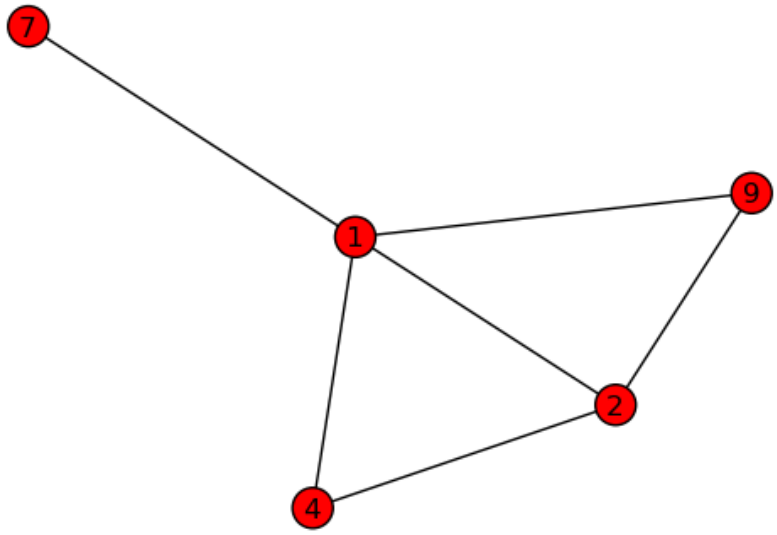


Removing Nodes and Edges

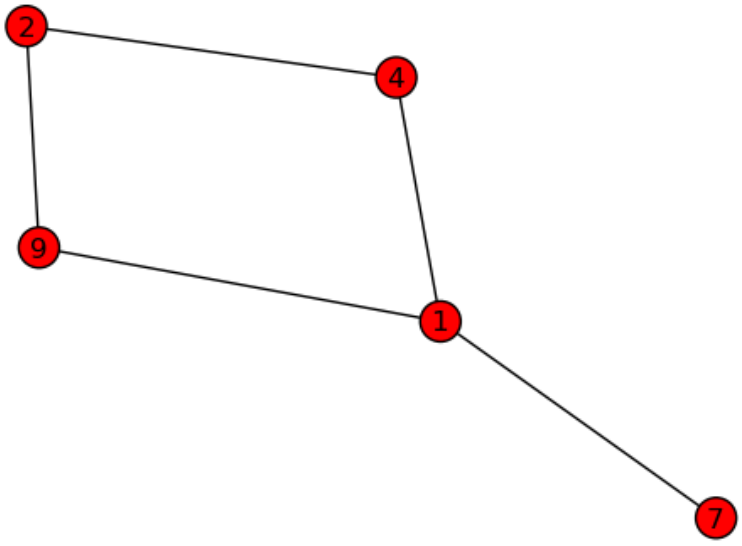
One can demolish the graph using any of these functions:

```
Graph.remove_node(), Graph.remove_nodes_from(),
Graph.remove_edge() and Graph.remove_edges_from()
```

After removing node 3



After removing edge (1,2)




```
# Python program to create an undirected
# graph and add nodes and edges to a graph

# To import package
import networkx

# To create an empty undirected graph
G = networkx.Graph()

# To add a node
G.add_node(1)
```

```
G.add_node(2)
G.add_node(3)
G.add_node(4)
G.add_node(7)
G.add_node(9)

# To add an edge
# Note graph is undirected
# Hence order of nodes in edge doesn't matter
G.add_edge(1,2)
G.add_edge(3,1)
G.add_edge(2,4)
G.add_edge(4,1)
G.add_edge(9,1)
G.add_edge(1,7)
G.add_edge(2,9)

# To get all the nodes of a graph
node_list = G.nodes()
print("#1")
print(node_list)

# To get all the edges of a graph
edge_list = G.edges()
print("#2")
print(edge_list)

# To remove a node of a graph
G.remove_node(3)
node_list = G.nodes()
print("#3")
print(node_list)

# To remove an edge of a graph
G.remove_edge(1,2)
edge_list = G.edges()
print("#4")
print(edge_list)

# To find number of nodes
n = G.number_of_nodes()
print("#5")
print(n)

# To find number of edges
m = G.number_of_edges()
print("#6")
print(m)

# To find degree of a node
# d will store degree of node 2
d = G.degree(2)
print("#7")
print(d)

# To find all the neighbor of a node
neighbor_list = G.neighbors(2)
print("#8")
print(neighbor_list)

#To delete all the nodes and edges
G.clear()
```

Run on IDE

Output :

```
#1
```

```
[1, 2, 3, 4, 7, 9]
#2
[(1, 9), (1, 2), (1, 3), (1, 4), (1, 7), (2, 4), (2, 9)]
#3
[1, 2, 4, 7, 9]
#4
[(1, 9), (1, 4), (1, 7), (2, 4), (2, 9)]
#5
5
#6
5
#7
2
#8
[4, 9]
```

In the next post, we'll be discussing how to create weighted graphs, directed graphs, multi graphs. How to draw graphs. In later posts we'll see how to use inbuilt functions like Depth fist search aka dfs, breadth first search aka BFS, dijkstra's shortest path algorithm.

Reference : Networxx at Github

This article is contributed by **Pratik Chhajer**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner Company Wise Coding Practice

Graph Python

Recommended Posts:

getpass() and getuser() in Python (Password without echo)
pickle — Python object serialization
Word Ladder (Length of shortest chain to reach a target word)
Print all paths from a given source to a destination
Python GUI – tkinter

(Login to Rate and Mark)

2.5 Average Difficulty : **2.5/5.0**
Based on **2** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!

 Contact Us! About Us! Careers! Privacy Policy