NetworkX is a powerful Python library for creating, analyzing, and visualizing complex networks (graphs). This guide covers installation, basic usage, and common operations.

1. Installation

Before importing NetworkX, ensure it's installed:

Using pip (Recommended)



Using conda (for Anaconda users)



Verify Installation

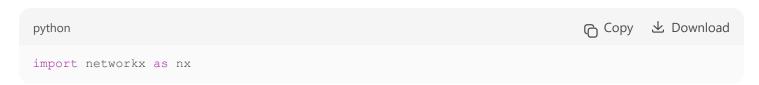
```
python

import networkx as nx

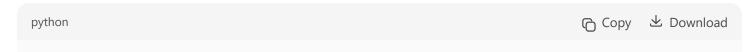
print(nx.__version__) # Should print the installed version
```

2. Importing NetworkX

The standard convention is to import it as <code>nx</code>:



Optional: Import Common Submodules



```
import matplotlib.pyplot as plt # For graph visualization
```

3. Basic Graph Creation

Create an Empty Graph

```
python

G = nx.Graph() # Undirected graph

DG = nx.DiGraph() # Directed graph

MG = nx.MultiGraph() # Undirected multigraph (multiple edges)

MDG = nx.MultiDiGraph() # Directed multigraph
```

Add Nodes

```
python

G.add_node(1) # Single node

G.add_nodes_from([2, 3, 4]) # Multiple nodes

G.add_node("A", weight=5) # Node with attributes
```

Add Edges

```
python

G.add_edge(1, 2) # Single edge

G.add_edges_from([(1, 3), (2, 4)]) # Multiple edges

G.add_edge("A", "B", weight=2.5) # Edge with attributes
```

4. Graph Analysis

Basic Properties

```
python

print("Nodes:", G.nodes())
print("Edges:", G.edges())
print("Number of nodes:", G.number_of_nodes())
```

```
print("Number of edges:", G.number_of_edges())
print("Degree of node 1:", G.degree(1)) # Number of connections
```

Shortest Path

```
python

path = nx.shortest_path(G, source=1, target=4)
print("Shortest path from 1 to 4:", path)
```

Centrality Measures

```
python

degree_centrality = nx.degree_centrality(G)

betweenness_centrality = nx.betweenness_centrality(G)

closeness_centrality = nx.closeness_centrality(G)

print("Degree centrality:", degree_centrality)
```

5. Visualization

Draw a Simple Graph

```
python

nx.draw(G, with_labels=True, node_color='lightblue', edge_color='gray')
plt.show() # Requires matplotlib
```

Customized Plot

```
python

pos = nx.spring_layout(G)  # Layout algorithm

nx.draw(G, pos, with_labels=True, node_size=500, node_color='skyblue', font_weight='bold')

plt.title("NetworkX Graph Visualization")

plt.show()
```

6. Importing/Exporting Graphs

Save & Load (Adjacency List)

```
python

nx.write_adjlist(G, "graph.adjlist") # Save

G = nx.read_adjlist("graph.adjlist") # Load
```

Export to Pandas (for DataFrames)

```
python

import pandas as pd

df_edges = nx.to_pandas_edgelist(G)

df_nodes = pd.DataFrame.from_dict(dict(G.nodes(data=True)), orient='index')
```

7. Common Algorithms

Algorithm	Usage
BFS / DFS Traversal	<pre>list(nx.bfs_tree(G, source=1))</pre>
Connected Components	<pre>list(nx.connected_components(G))</pre>
PageRank	nx.pagerank(G)
Minimum Spanning Tree	<pre>nx.minimum_spanning_tree(G)</pre>
Community Detection	<pre>nx.algorithms.community.greedy_modularity_communities(G)</pre>

8. Advanced: Random Graphs

```
python

# Erdős-Rényi random graph

G_random = nx.erdos_renyi_graph(20, 0.1) # 20 nodes, 10% edge probability
```

```
# Barabási-Albert (scale-free) graph
G_ba = nx.barabasi_albert_graph(50, 2) # 50 nodes, 2 edges per new node
```

9. Troubleshooting

- ImportError: No module named 'networkx' → Reinstall using pip install networkx.
- **Graph not displaying?** → Ensure matplotlib is installed (pip install matplotlib).
- Slow performance? → Use nx.convert_node_labels_to_integers(G) for large graphs.

Conclusion

NetworkX (nx) is a versatile library for graph analysis. This guide covers:

- ✓ Installation & import
- Graph creation & modification
- ✓ Basic analysis & visualization
- Common algorithms

For more details, check the official NetworkX documentation.