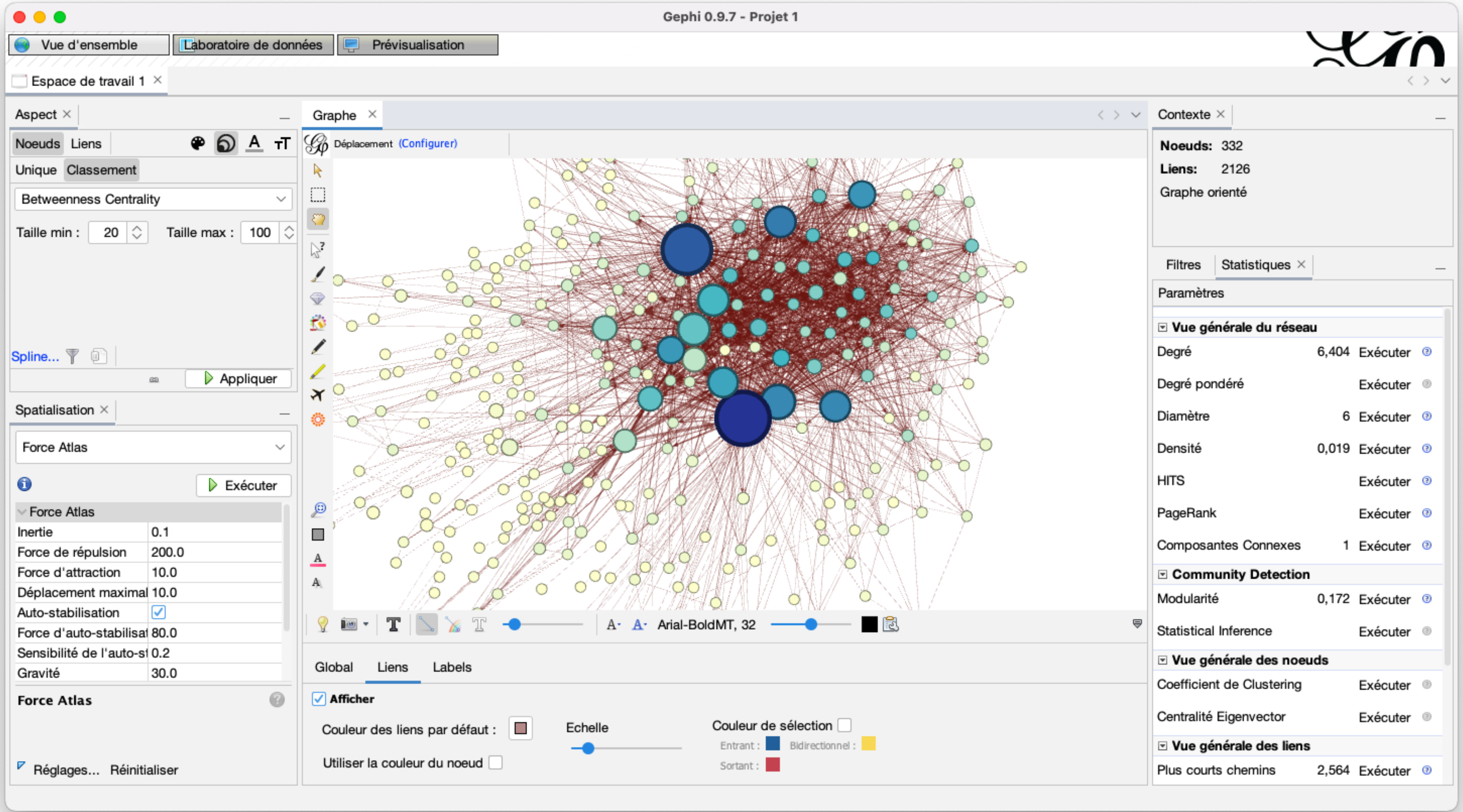
Part V

Tools

GEPHI

- **Gephi** is an open-source software for network analysis and visualization, developed in Java.
- Its main advantage lies in mapping data and performing various graph theory calculations through an intuitive graphical interface.
- This enables the visualization of key aspects of a network, such as identifying the most central elements, the most distant nodes, or the best-connected components.
- Gephi supports several network file formats, including .csv, .xls, .net, .gml, .dot, and .gdf.

GEPHI



Aspect — Noeuds — Classement: Betweenness Centrality

Taille min : 20 Taille max : 100

Spatialisation: Force Atlas

Inertie	0.1
Force de répulsion	200.0
Force d'attraction	10.0
Déplacement maximal	10.0
Auto-stabilisation	✓
Force d'auto-stabilisat	80.0
Sensibilité de l'auto-st	0.2
Gravité	30.0

Force Atlas

Réglages... Réinitialiser

Liens: ✓ Afficher

Couleur des liens par défaut : 🟫 Echelle

Utiliser la couleur du noeud ☐

Couleur de sélection ☐ Entrant : 🟦 Bidirectionnel : 🟨 Sortant : 🟥

Contexte

Noeuds: 332

Liens: 2126

Graphe orienté

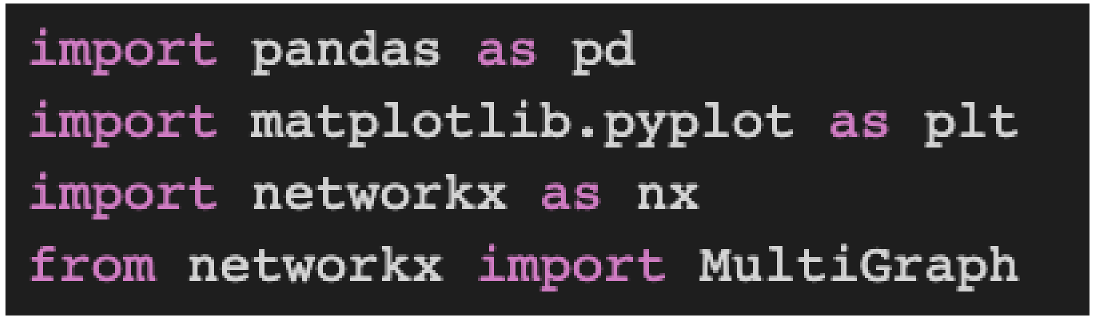
Statistiques

Vue générale du réseau

Degré	6,404	Exécuter
Degré pondéré		Exécuter
Diamètre	6	Exécuter
Densité	0,019	Exécuter
HITS		Exécuter
PageRank		Exécuter
Composantes Connexes	1	Exécuter
Community Detection		
Modularité	0,172	Exécuter
Statistical Inference		Exécuter
Vue générale des noeuds		
Coefficient de Clustering		Exécuter
Centralité Eigenvector		Exécuter
Vue générale des liens		
Plus courts chemins	2,564	Exécuter

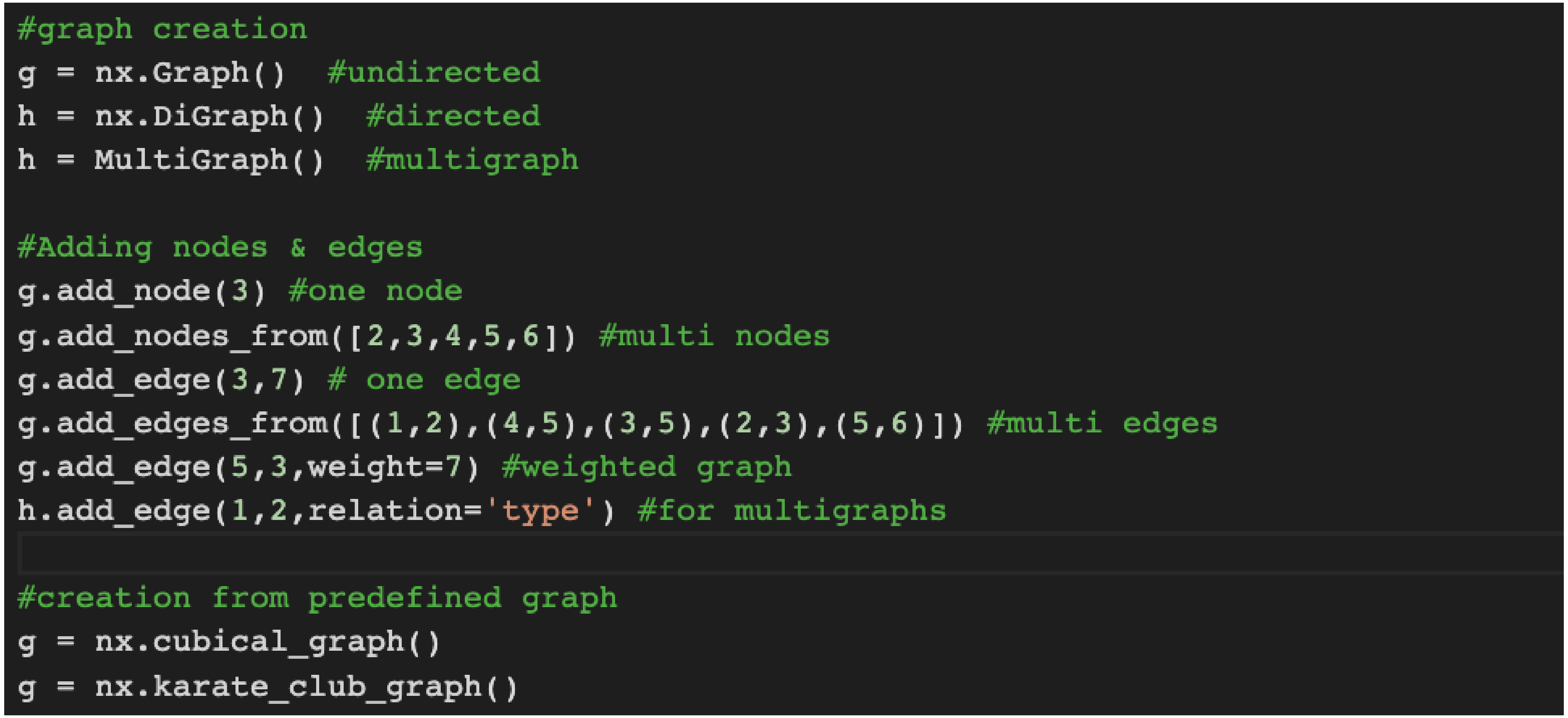
Networkx

- NetworkX, a Python library for studying the structure, dynamics, and functions of complex networks.
- Installation
  - pip install networkx #or
  - conda install -c anaconda networkx #in anaconda
  - Already Exist #in google colab
- Required Libraries



```python
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
from networkx import MultiGraph
```

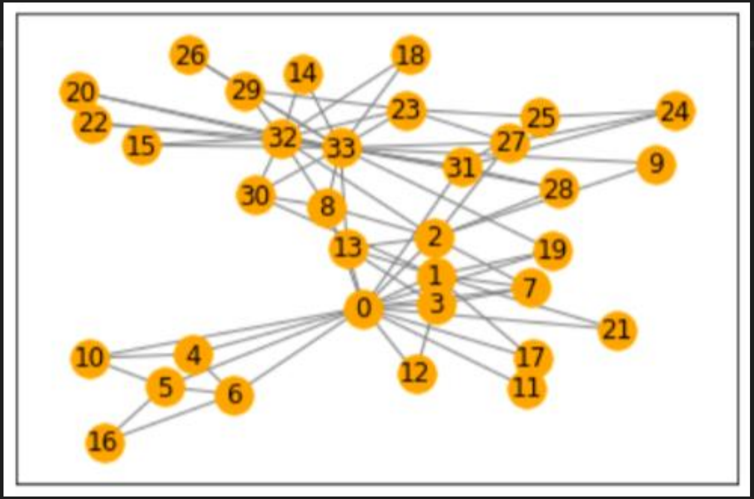
Networkx



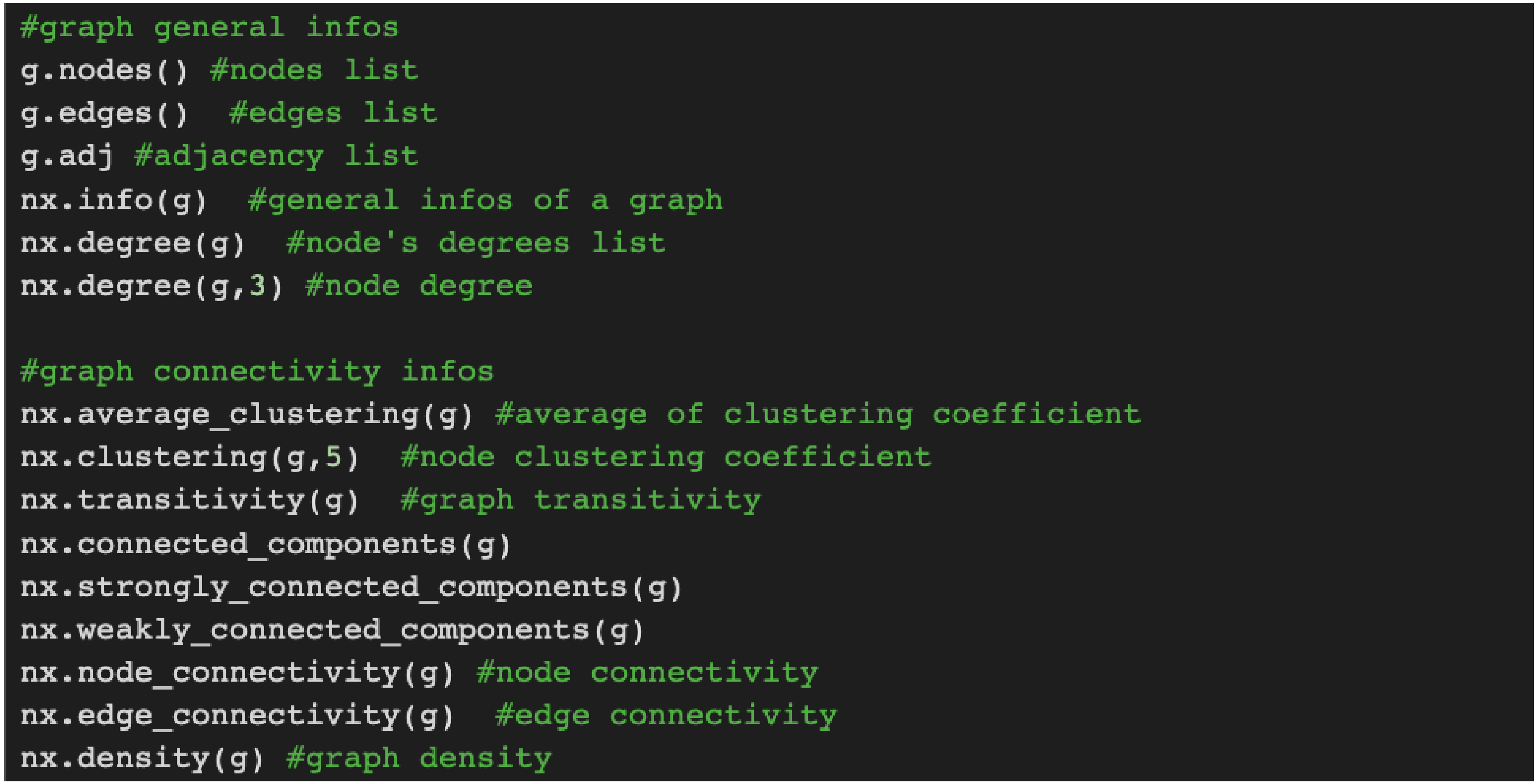
```
#graph creation
g = nx.Graph()  #undirected
h = nx.DiGraph()  #directed
h = MultiGraph()  #multigraph

#Adding nodes & edges
g.add_node(3) #one node
g.add_nodes_from([2,3,4,5,6]) #multi nodes
g.add_edge(3,7) # one edge
g.add_edges_from([(1,2),(4,5),(3,5),(2,3),(5,6)]) #multi edges
g.add_edge(5,3,weight=7) #weighted graph
h.add_edge(1,2,relation='type') #for multigraphs

#creation from predefined graph
g = nx.cubical_graph()
g = nx.karate_club_graph()
```

Networkx

```
#creation from txt file
g = nx.read_edgelist('path/graph.txt', create_using=nx.Graph(), nodetype=int)

#creation from csv file
df = pd.read_csv("path/graph.csv")
g = nx.from_pandas_edgelist(df, source="col1_name", target="col2_name")

#creation from gml file
g = nx.read_gml('path/graph.gml')

#Removing nodes & edges
g.remove_node(0)  #node
g.remove_edge(5,3)  #edge

#drawing a graph
nx.draw(g)  #use layout by pos param: nx.draw(g,pos=nx.circular_layout(g))
nx.draw_networkx(g) #node_color, edge_color params for coloring: node_color='r'
```

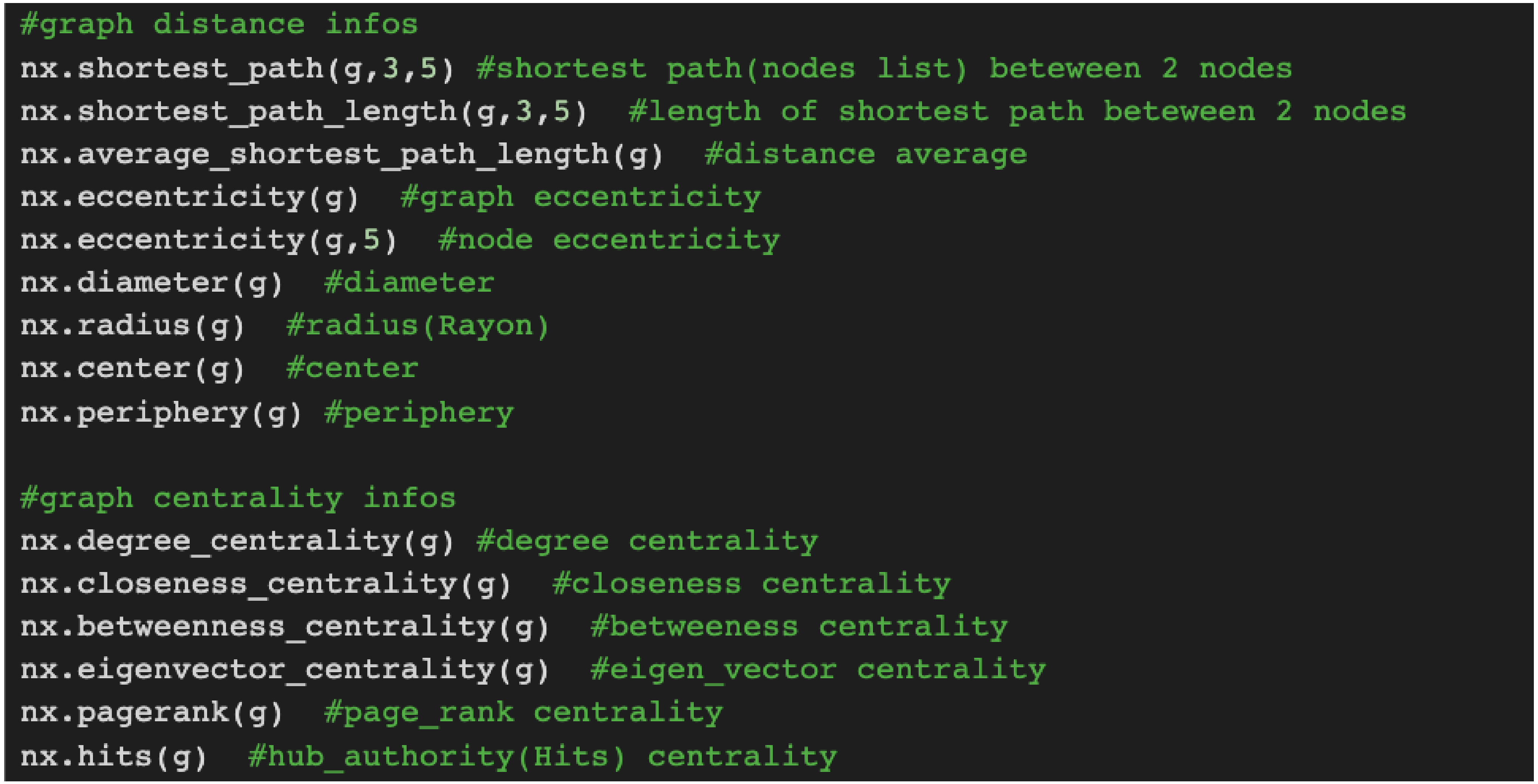


Networkx



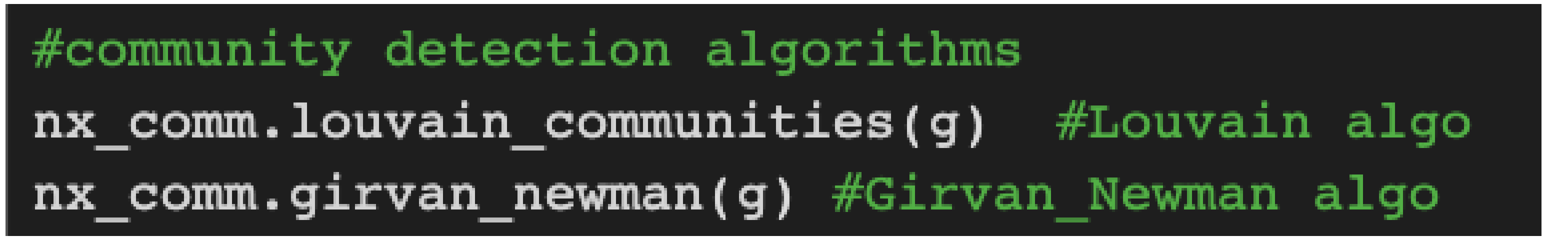
```
#graph general infos
g.nodes() #nodes list
g.edges()  #edges list
g.adj #adjacency list
nx.info(g)  #general infos of a graph
nx.degree(g)  #node's degrees list
nx.degree(g,3) #node degree

#graph connectivity infos
nx.average_clustering(g) #average of clustering coefficient
nx.clustering(g,5)  #node clustering coefficient
nx.transitivity(g)  #graph transitivity
nx.connected_components(g)
nx.strongly_connected_components(g)
nx.weakly_connected_components(g)
nx.node_connectivity(g) #node connectivity
nx.edge_connectivity(g)  #edge connectivity
nx.density(g) #graph density
```

Networkx



```
#graph distance infos
nx.shortest_path(g,3,5) #shortest path(nodes list) beteween 2 nodes
nx.shortest_path_length(g,3,5)  #length of shortest path beteween 2 nodes
nx.average_shortest_path_length(g)  #distance average
nx.eccentricity(g)  #graph eccentricity
nx.eccentricity(g,5)  #node eccentricity
nx.diameter(g)  #diameter
nx.radius(g)  #radius(Rayon)
nx.center(g)  #center
nx.periphery(g) #periphery

#graph centrality infos
nx.degree_centrality(g) #degree centrality
nx.closeness_centrality(g)  #closeness centrality
nx.betweenness_centrality(g)  #betweeness centrality
nx.eigenvector_centrality(g)  #eigen_vector centrality
nx.pagerank(g)  #page_rank centrality
nx.hits(g)  #hub_authority(Hits) centrality
```

Networkx



```
#community detection algorithms
nx_comm.louvain_communities(g)  #Louvain algo
nx_comm.girvan_newman(g) #Girvan_Newman algo
```

End …