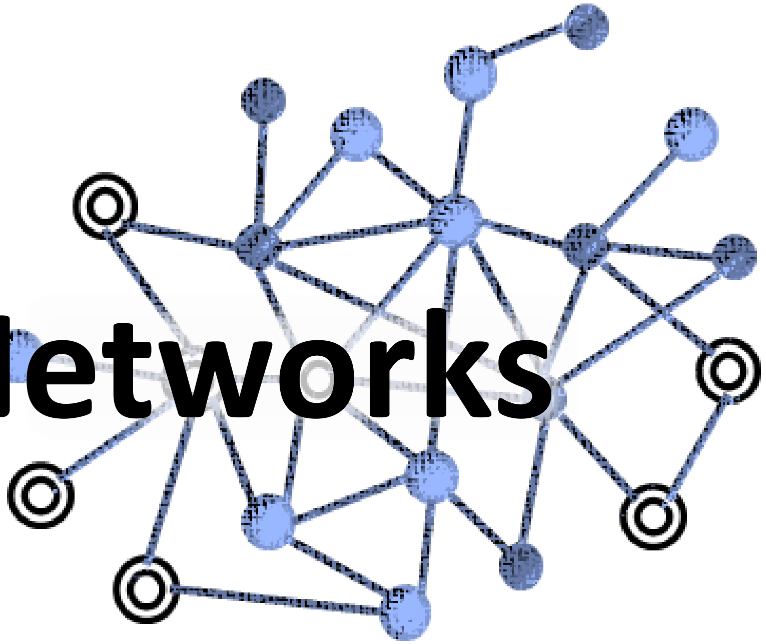
선문대학교
SUN MOON UNIVERSITY

Lab#01 – NetworkX

Social & Information Networks



김 민 경

AI소프트웨어학과

https://www.minkyoung.kim/

Outline

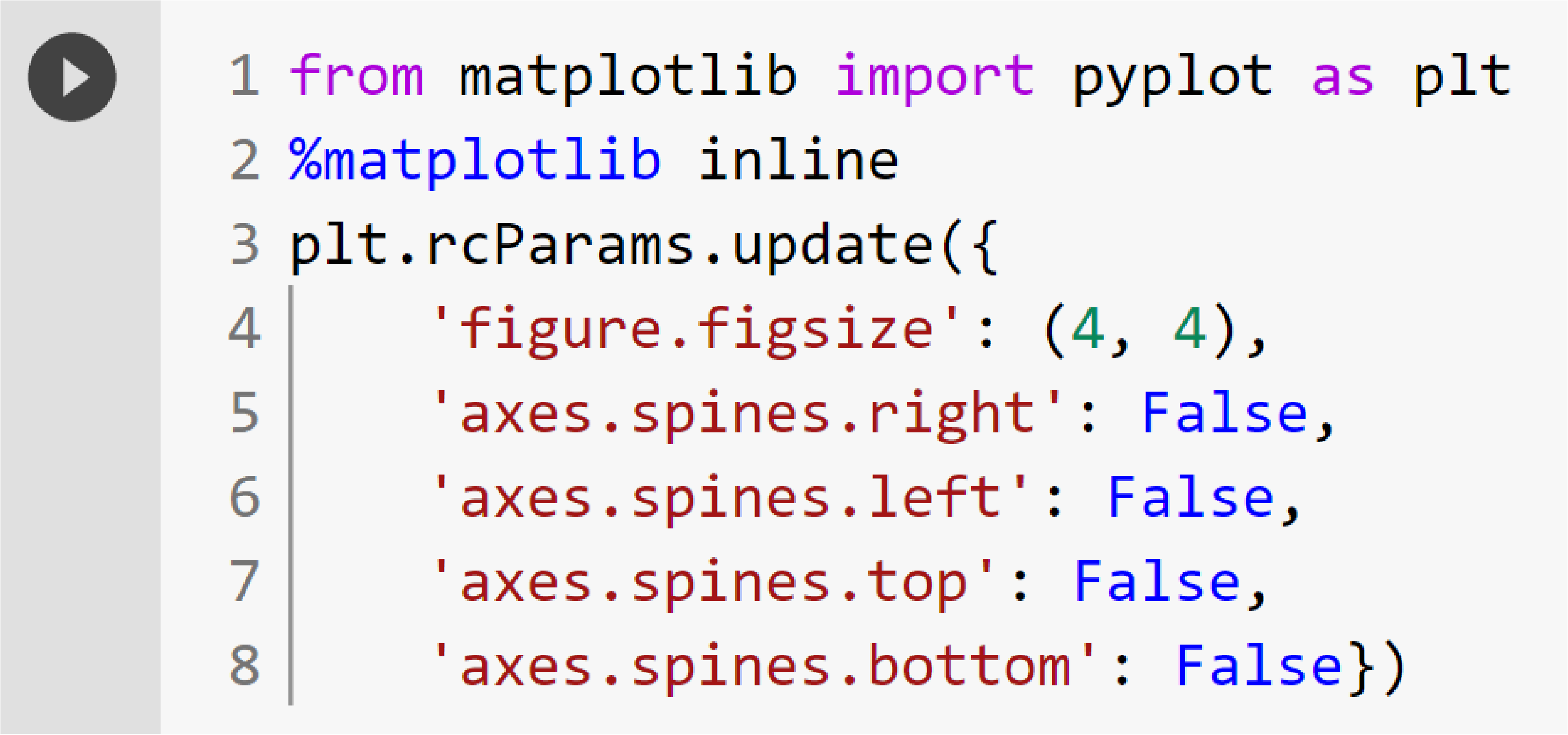
- **NetworkX**
  - Introduction
  - Step-by-Step Fundamentals with Python
    - Basic Classes
    - Graph Construction
    - Graph Information
    - Plot

What is NetworkX?

NetworkX
Network Analysis in Python

- Python package for the creation, manipulation and study of the structure, dynamics and functions of complex networks
  - **Data structures** for representing many types of networks
  - **Nodes** can be any (hashable) Python object
  - **Edges** can contain arbitrary data
  - Focus on **computational network modeling** not software tool development
- Online documentation
  - **https://networkx.github.io/documentation/stable/tutorial.html**

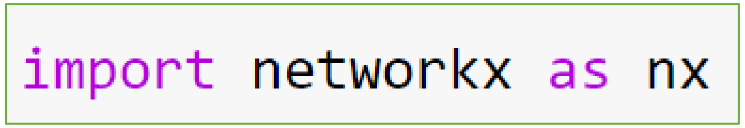
Configure plotting in Jupyter Environment



```python
from matplotlib import pyplot as plt
%matplotlib inline
plt.rcParams.update({
    'figure.figsize': (4, 4),
    'axes.spines.right': False,
    'axes.spines.left': False,
    'axes.spines.top': False,
    'axes.spines.bottom': False})
```

Getting Started

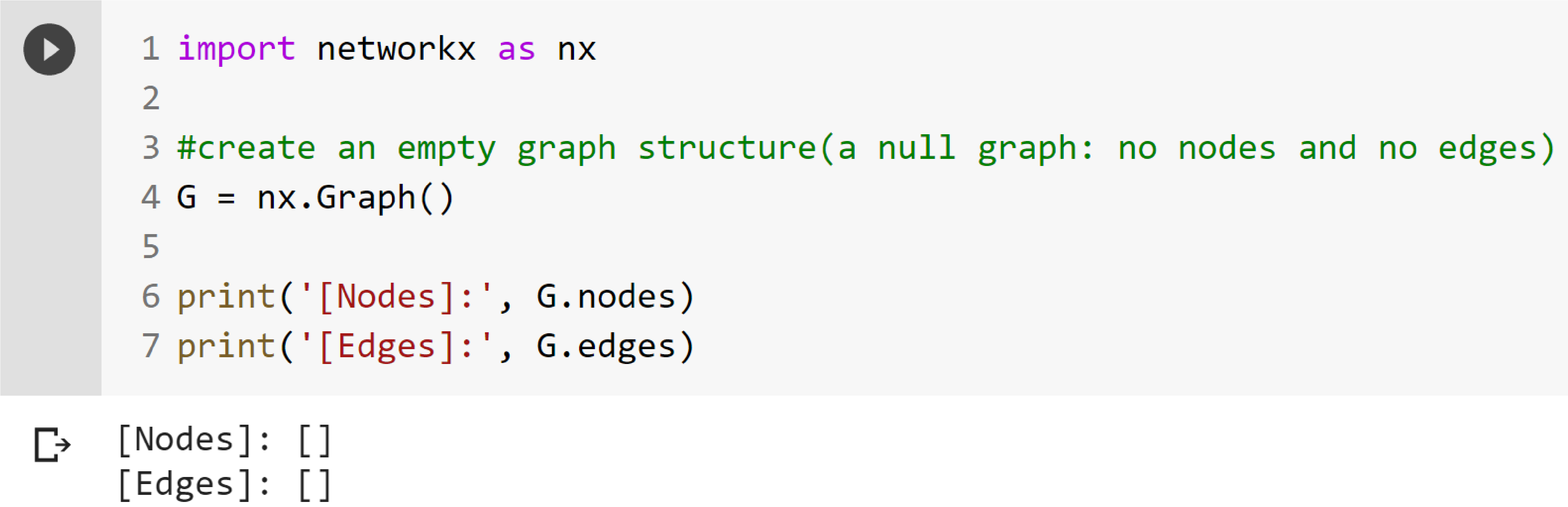
- Start Python and import NetworkX:



```
import networkx as nx
```

- Diverse graph classes for undirected and directed networks:
  - ***Graph***: undirected simple (allows self loops) → a basic graph class
  - ***DiGraph***: directed simple (allows self loops)
  - ***MultiGraph***: undirected with multiedges
  - ***MultiDiGraph***: directed with multiedges
- A graph instance *G* can be grown in different ways further.

Getting Started – Basic Graph Class



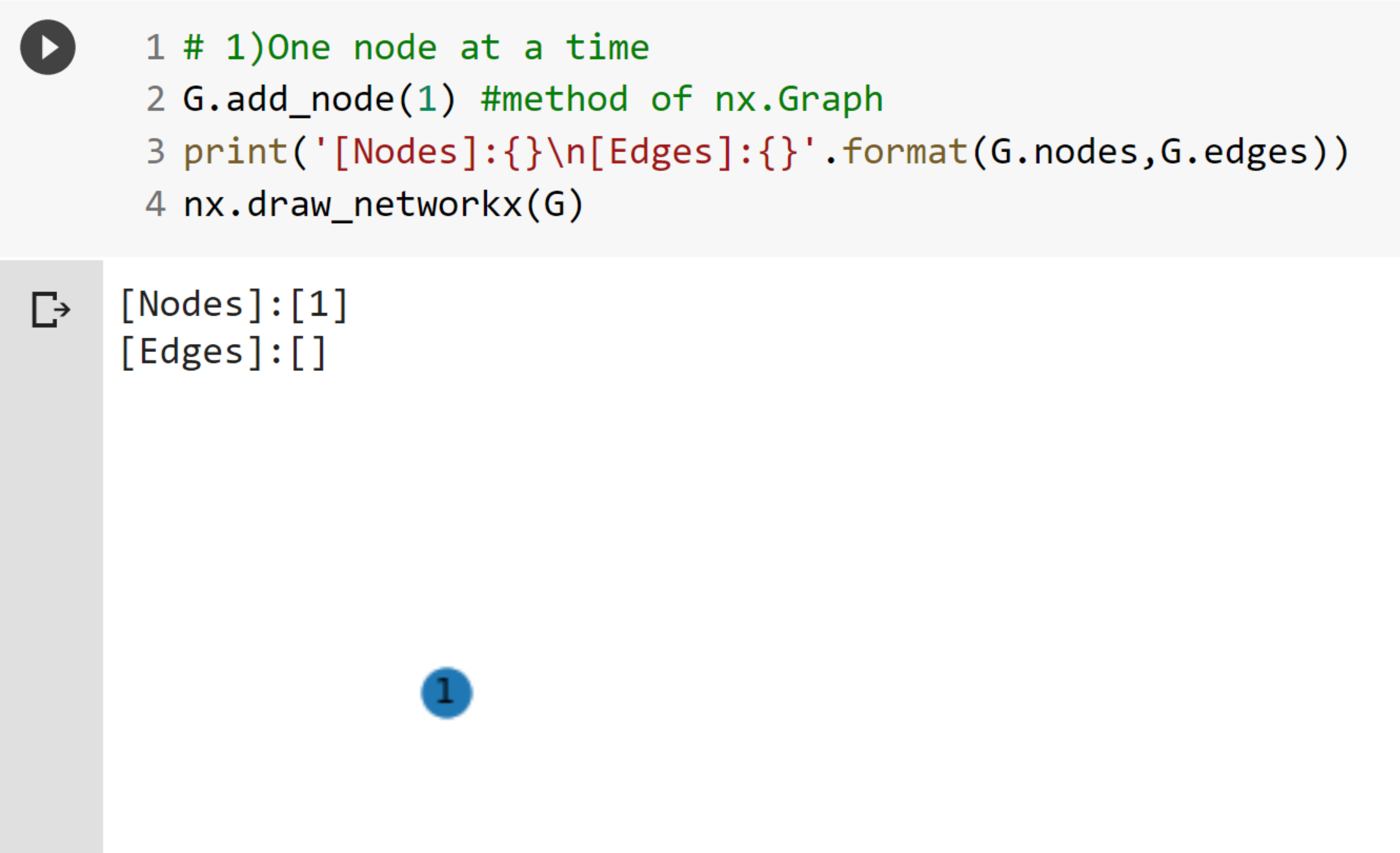
```python
import networkx as nx

#create an empty graph structure(a null graph: no nodes and no edges)
G = nx.Graph()

print('[Nodes]:', G.nodes)
print('[Edges]:', G.edges)
```

```
[Nodes]: []
[Edges]: []
```

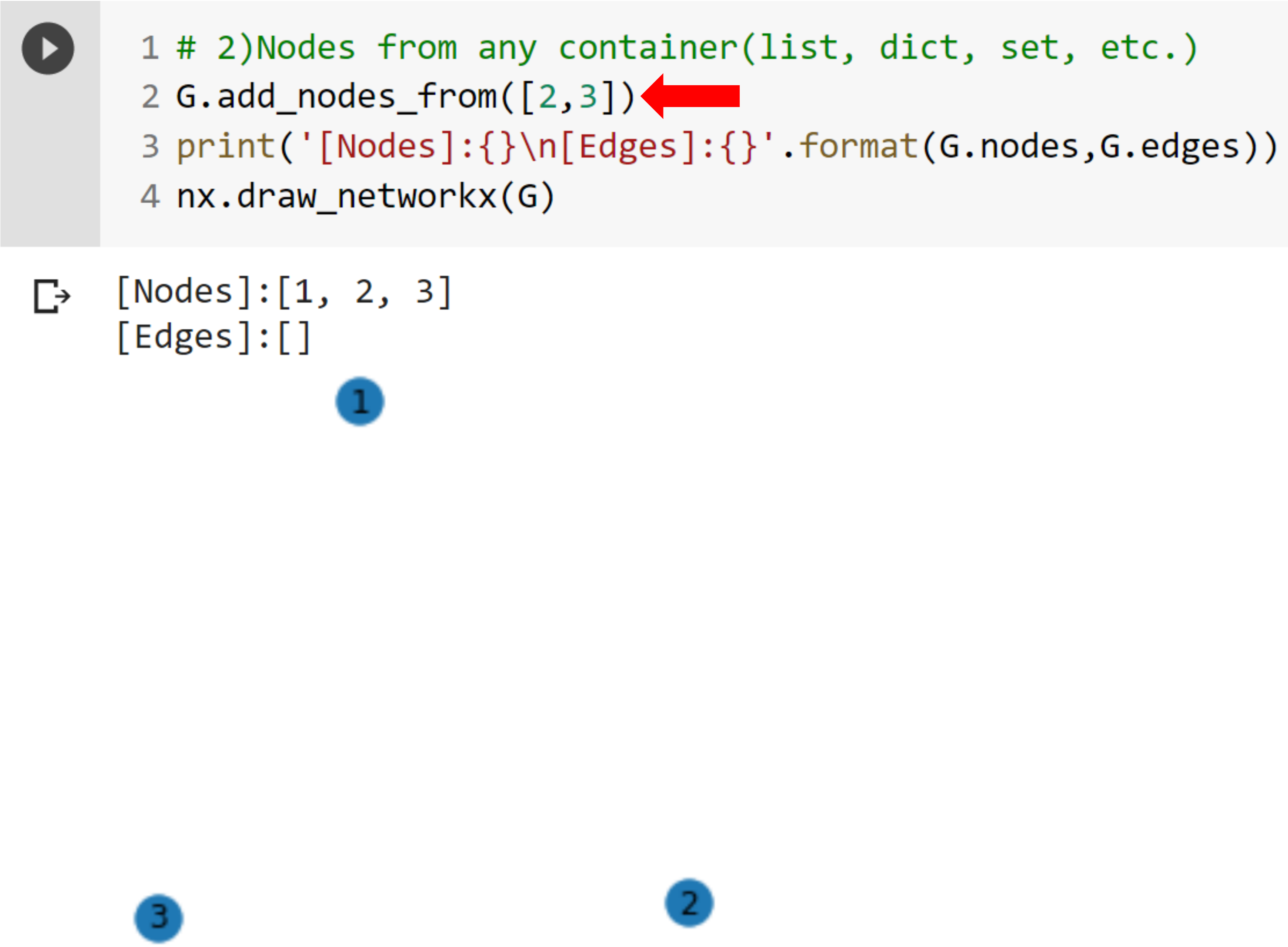
Add Nodes – Single Node



```
# 1)One node at a time
G.add_node(1) #method of nx.Graph
print('[Nodes]:{}\n[Edges]:{}'.format(G.nodes,G.edges))
nx.draw_networkx(G)
```

```
[Nodes]:[1]
[Edges]:[]
```

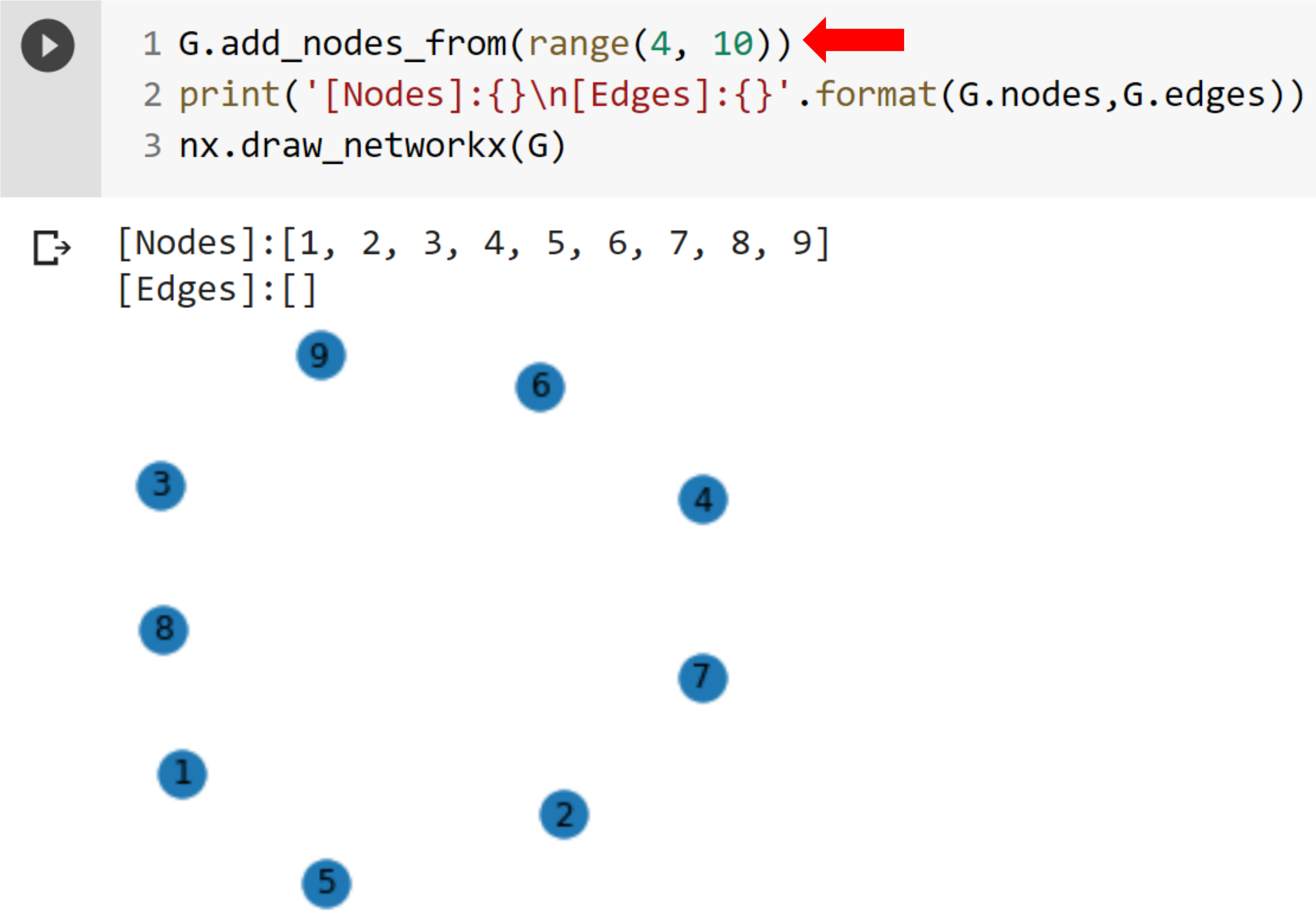
Add Nodes – Multiple Nodes(1)



```python
# 2)Nodes from any container(list, dict, set, etc.)
G.add_nodes_from([2,3])
print('[Nodes]:{}\n[Edges]:{}'.format(G.nodes,G.edges))
nx.draw_networkx(G)
```

```
[Nodes]:[1, 2, 3]
[Edges]:[]
```

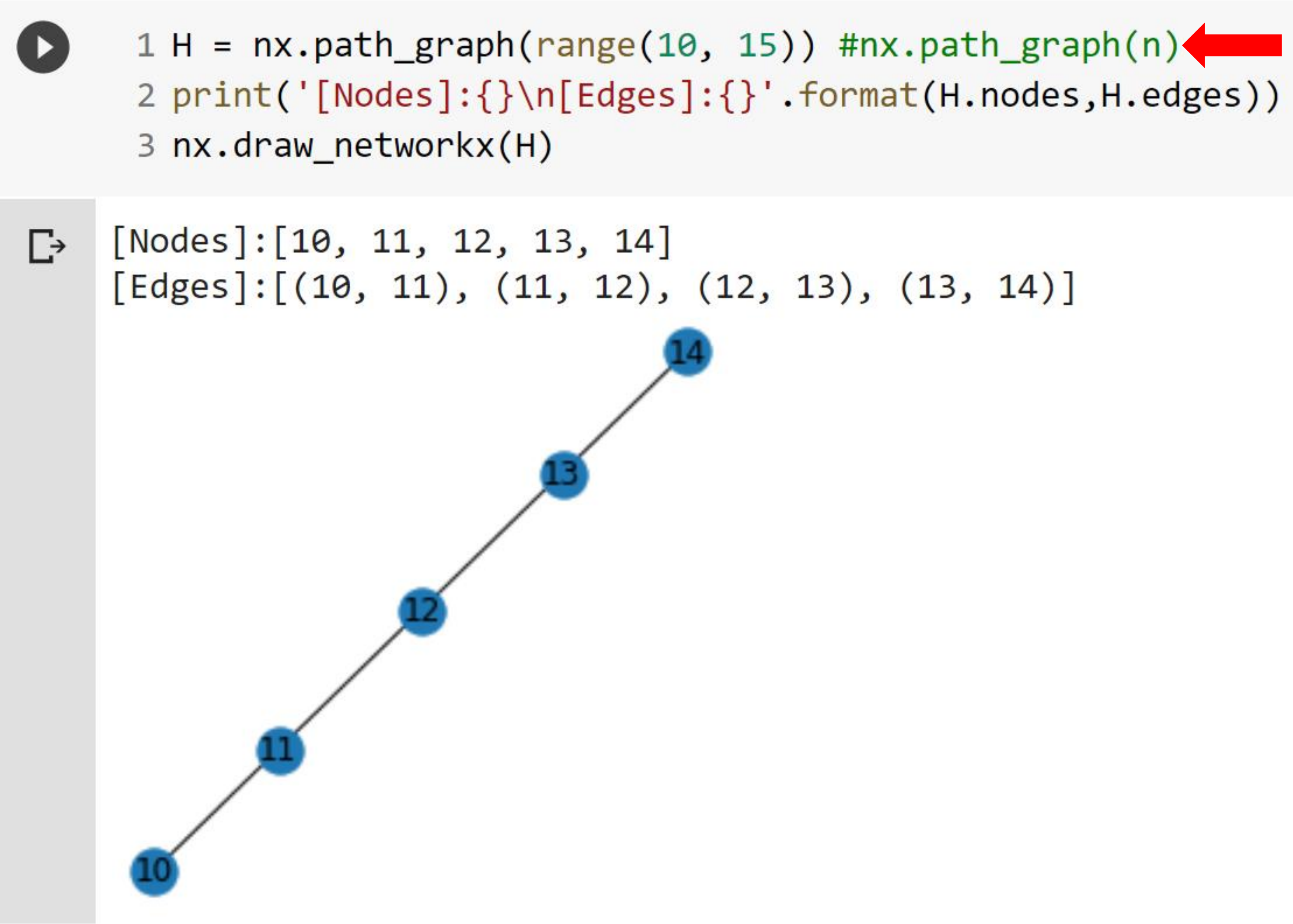
Add Nodes – Multiple Nodes(2)



```
G.add_nodes_from(range(4, 10))
print('[Nodes]:{}\n[Edges]:{}'.format(G.nodes,G.edges))
nx.draw_networkx(G)
```

```
[Nodes]:[1, 2, 3, 4, 5, 6, 7, 8, 9]
[Edges]:[]
```

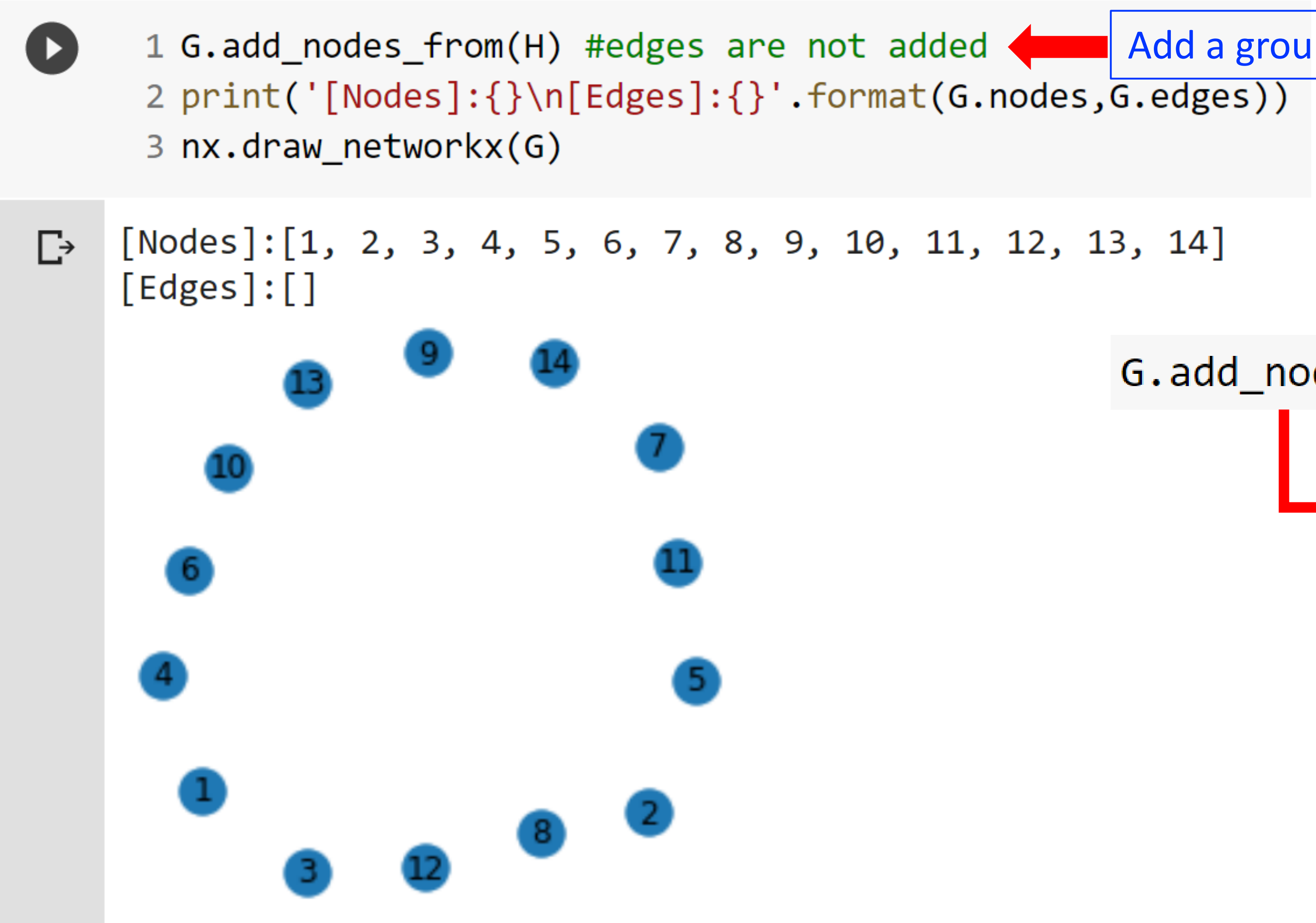
Add Nodes – Create a path graph for addition



```
H = nx.path_graph(range(10, 15)) #nx.path_graph(n)
print('[Nodes]:{}\n[Edges]:{}'.format(H.nodes,H.edges))
nx.draw_networkx(H)
```

```
[Nodes]:[10, 11, 12, 13, 14]
[Edges]:[(10, 11), (11, 12), (12, 13), (13, 14)]
```

Add Nodes – Multiple Nodes(3)



```python
G.add_nodes_from(H) #edges are not added
print('[Nodes]:{}\n[Edges]:{}'.format(G.nodes,G.edges))
nx.draw_networkx(G)
```

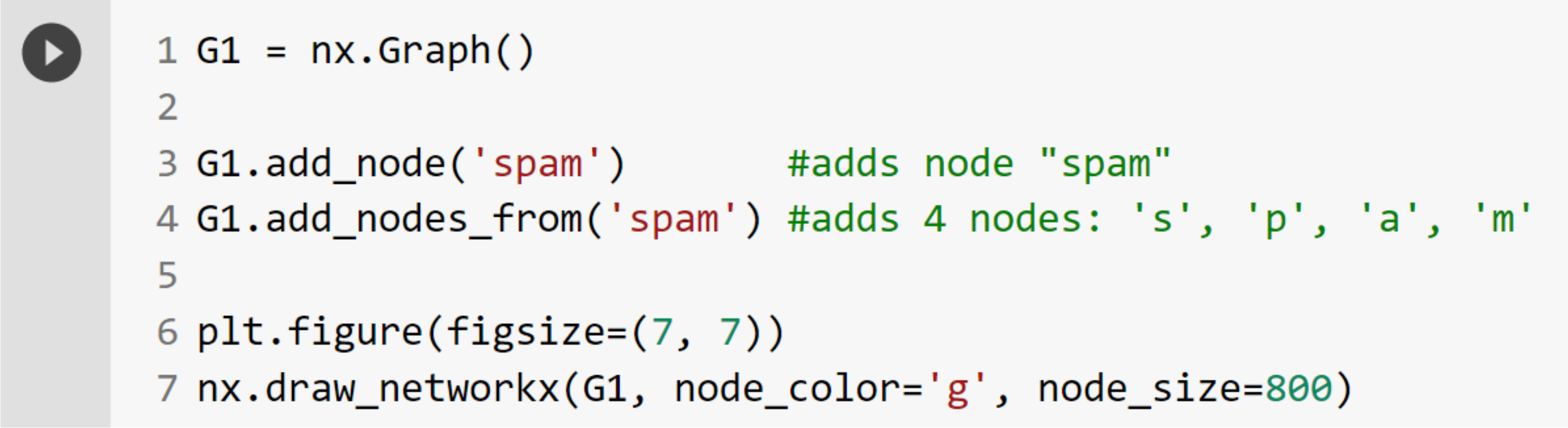
```
[Nodes]:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
[Edges]:[]
```

Add a group of nodes at once

```python
G.add_node(H)
```

Add another graph as a node

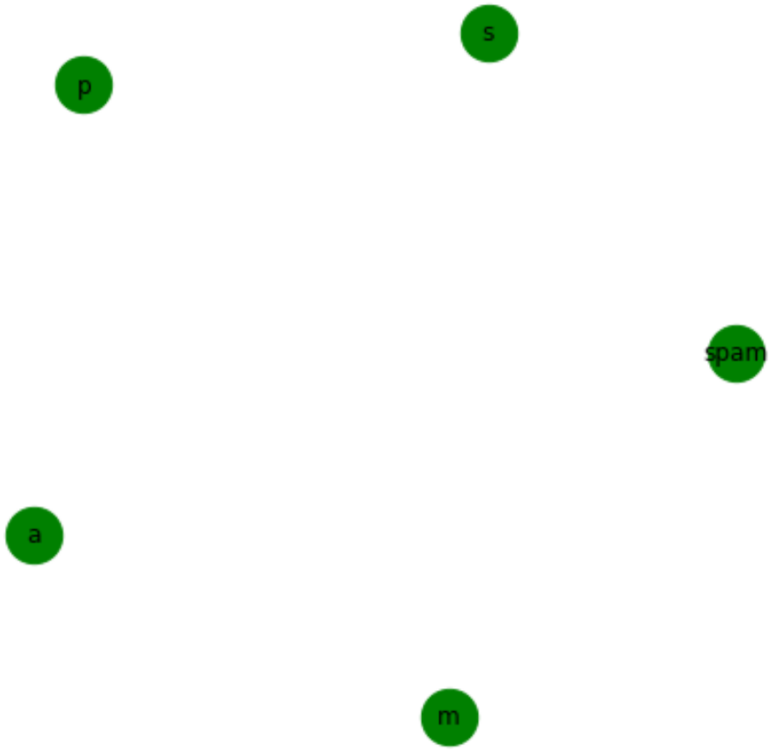
Add Nodes – Multiple Nodes(4)



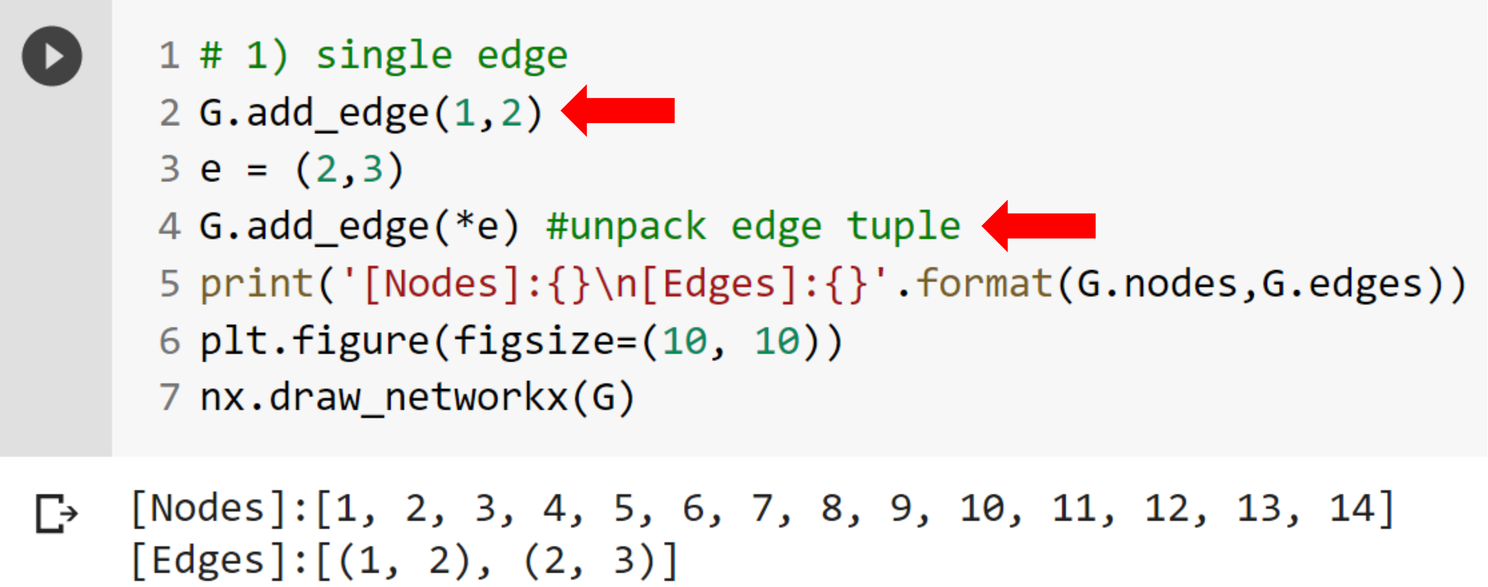
```
G1 = nx.Graph()

G1.add_node('spam')       #adds node "spam"
G1.add_nodes_from('spam') #adds 4 nodes: 's', 'p', 'a', 'm'

plt.figure(figsize=(7, 7))
nx.draw_networkx(G1, node_color='g', node_size=800)
```

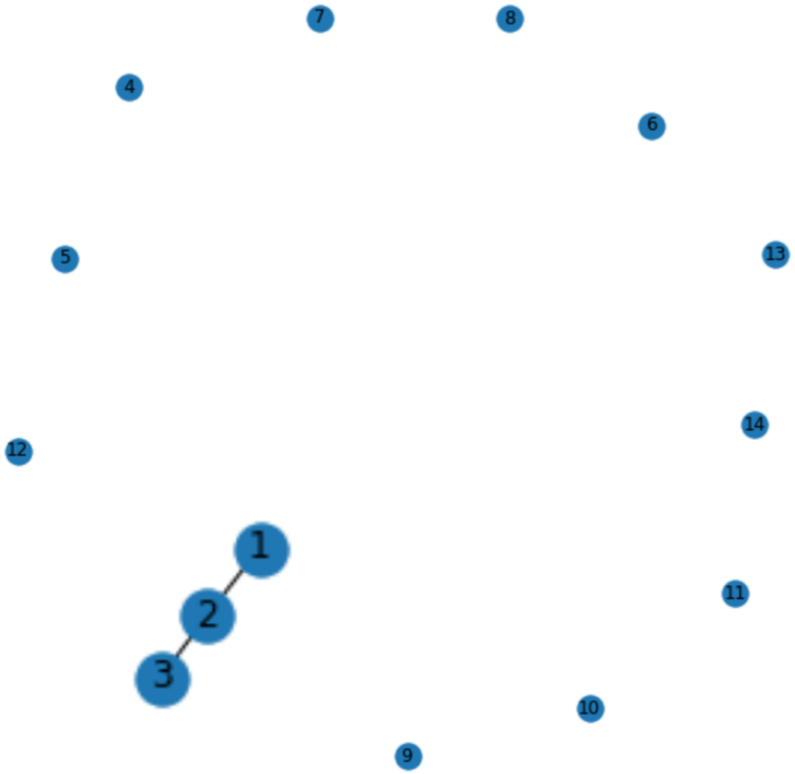


Add Edges – Single Edge

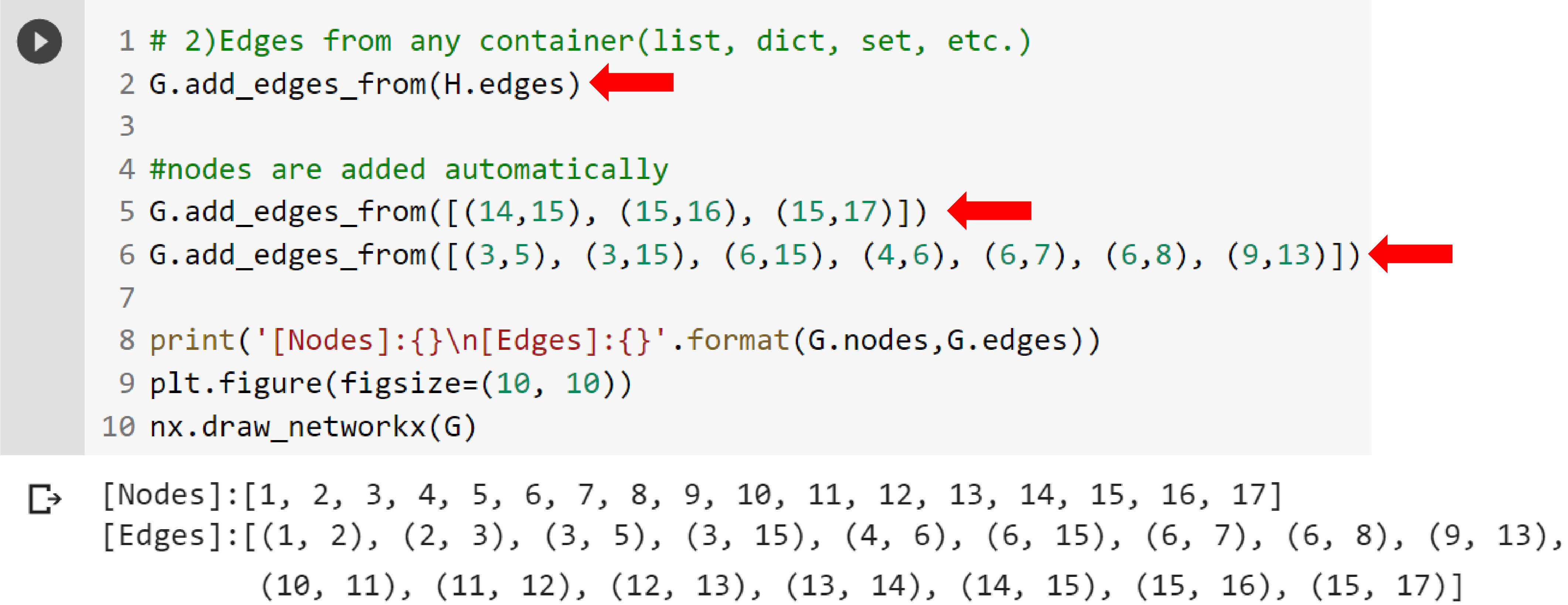


```
# 1) single edge
G.add_edge(1,2)
e = (2,3)
G.add_edge(*e) #unpack edge tuple
print('[Nodes]:{}\n[Edges]:{}'.format(G.nodes,G.edges))
plt.figure(figsize=(10, 10))
nx.draw_networkx(G)
```

```
[Nodes]:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
[Edges]:[(1, 2), (2, 3)]
```



Add Edges – Multiple Edges



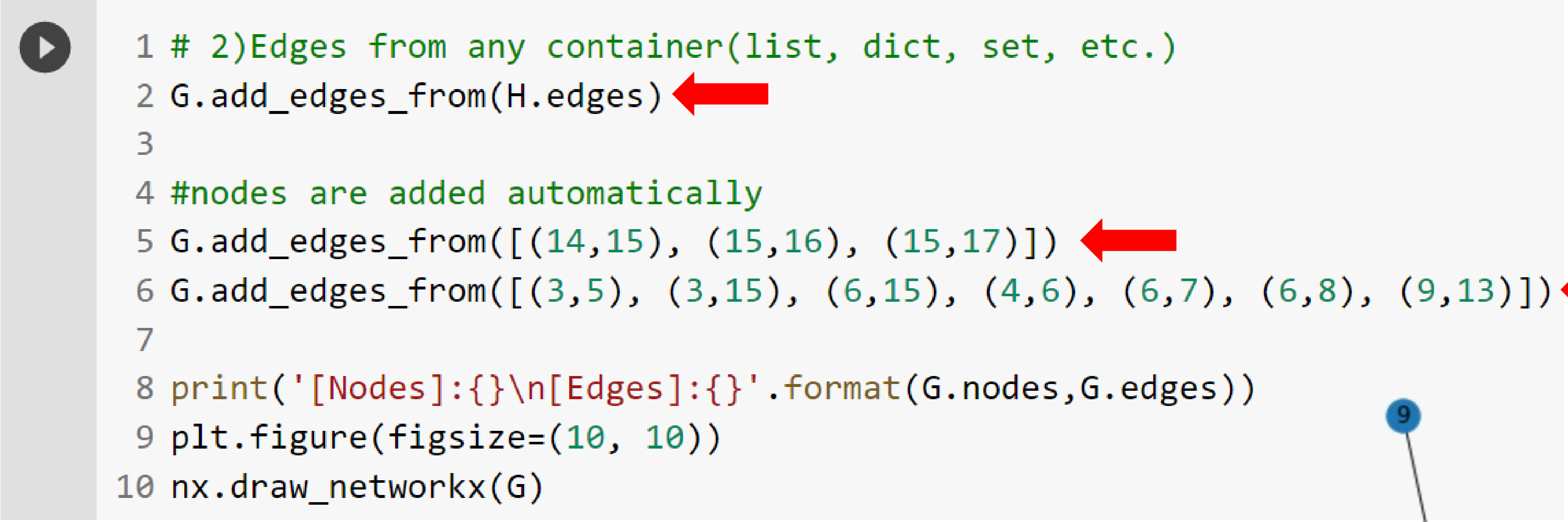
```
# 2)Edges from any container(list, dict, set, etc.)
G.add_edges_from(H.edges)

#nodes are added automatically
G.add_edges_from([(14,15), (15,16), (15,17)])
G.add_edges_from([(3,5), (3,15), (6,15), (4,6), (6,7), (6,8), (9,13)])

print('[Nodes]:{}\n[Edges]:{}'.format(G.nodes,G.edges))
plt.figure(figsize=(10, 10))
nx.draw_networkx(G)
```

```
[Nodes]:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
[Edges]:[(1, 2), (2, 3), (3, 5), (3, 15), (4, 6), (6, 15), (6, 7), (6, 8), (9, 13),
         (10, 11), (11, 12), (12, 13), (13, 14), (14, 15), (15, 16), (15, 17)]
```

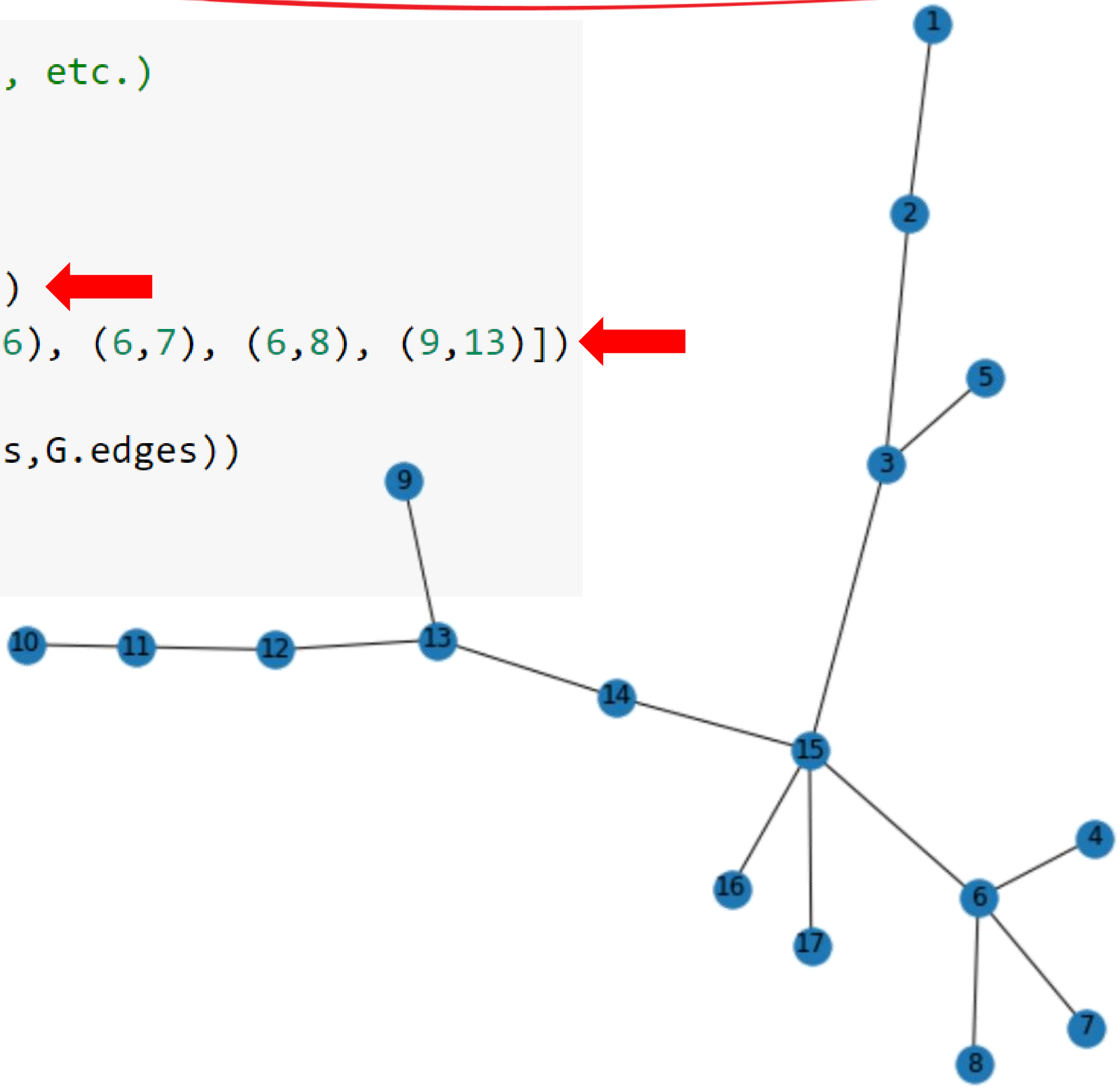
Add Edges – Multiple Edges



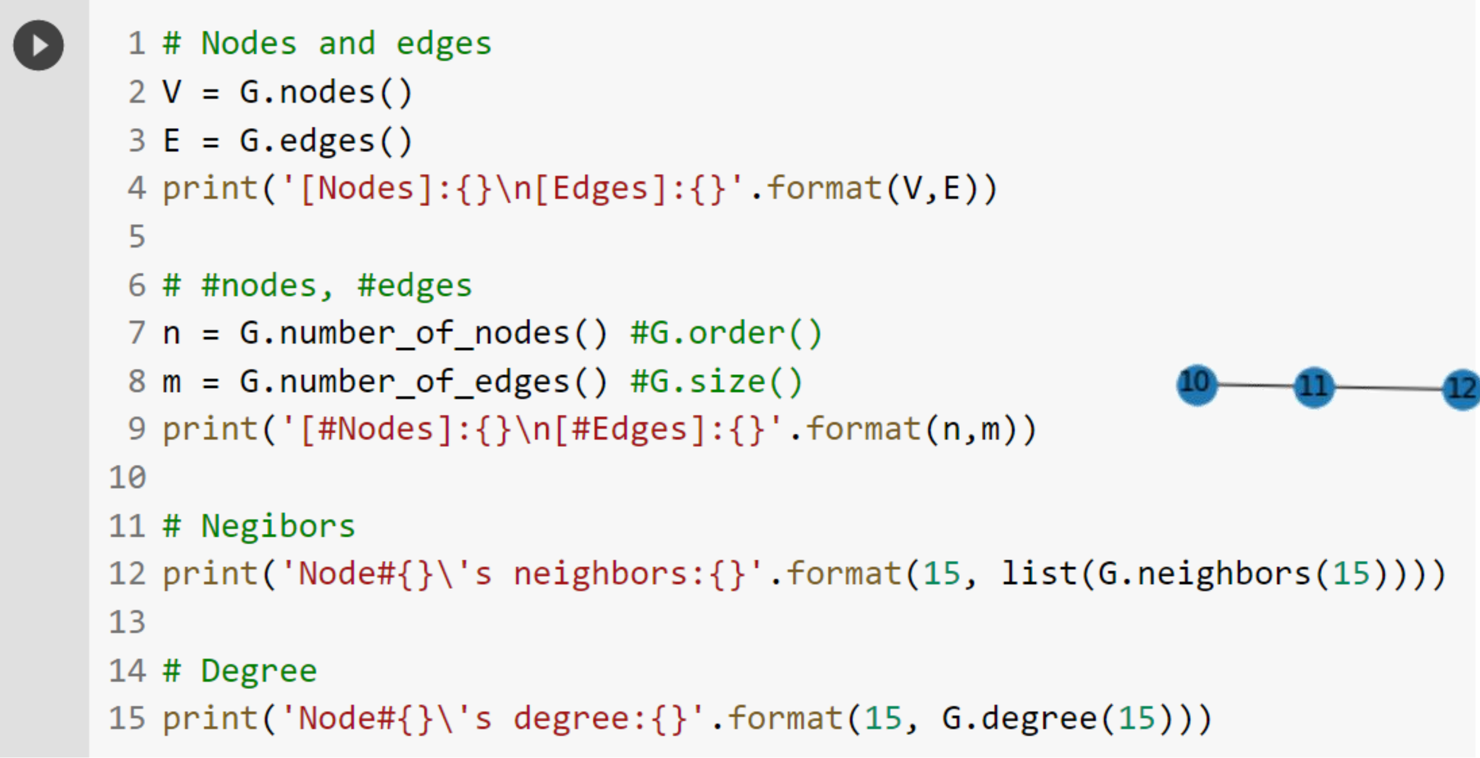
```
# 2)Edges from any container(list, dict, set, etc.)
G.add_edges_from(H.edges)

#nodes are added automatically
G.add_edges_from([(14,15), (15,16), (15,17)])
G.add_edges_from([(3,5), (3,15), (6,15), (4,6), (6,7), (6,8), (9,13)])

print('[Nodes]:{}\n[Edges]:{}'.format(G.nodes,G.edges))
plt.figure(figsize=(10, 10))
nx.draw_networkx(G)
```



Access Nodes and Edges



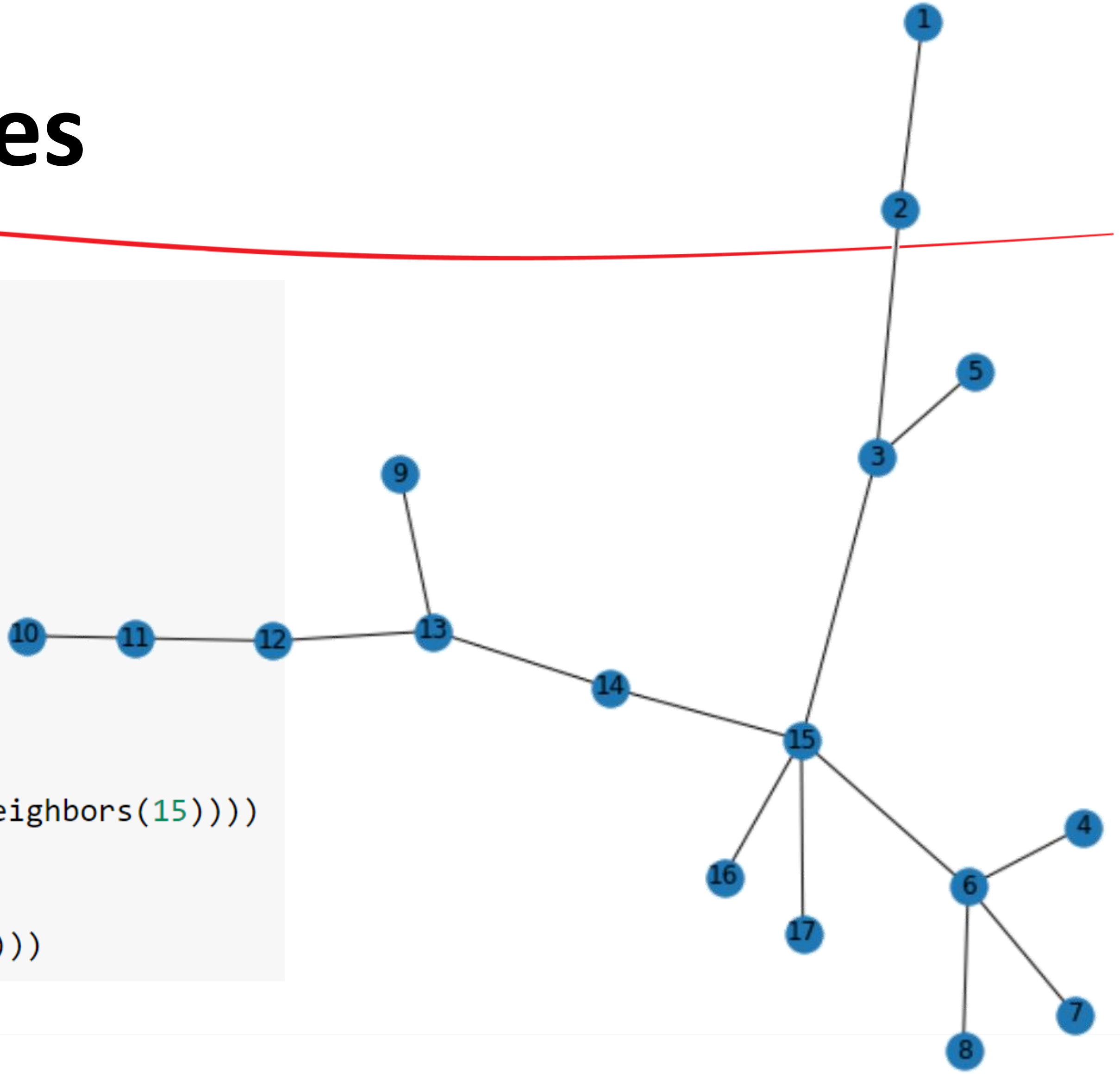
```python
# Nodes and edges
V = G.nodes()
E = G.edges()
print('[Nodes]:{}\n[Edges]:{}'.format(V,E))

# #nodes, #edges
n = G.number_of_nodes() #G.order()
m = G.number_of_edges() #G.size()
print('[#Nodes]:{}\n[#Edges]:{}'.format(n,m))

# Negibors
print('Node#{}\'s neighbors:{}'.format(15, list(G.neighbors(15))))

# Degree
print('Node#{}\'s degree:{}'.format(15, G.degree(15)))
```

```
[Nodes]:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
[Edges]:[(1, 2), (2, 3), (3, 5), (3, 15), (4, 6), (6, 15), (6, 7), (6, 8), (9, 13), (10, 11), (11, 12), (12, 13), (13, 14), (14, 15), (15, 16), (15, 17)]
[#Nodes]:17
[#Edges]:16
Node#15's neighbors:[14, 16, 17, 3, 6]
Node#15's degree:5
```


1
2
5
3
9
10
11
12
13
14
15
4
16
6
17
7
8