2022/9/26 Mon.

問題解決

Google Colaboratoryを用い pythonとnetworkx でgraph 描画

堀田 敬介

python, networkx でグラフを描画

- グラフ最適化の最適解をグラフ $G = (V, E)$ で描画したい
  - Google の Colaboratory を利用し，python, networkx を使う

  - 利用方法（初回）

    (1) google アカウントにログインし，google drive へ移動

    (2) 「新規」ー「その他」ー「アプリを追加」を選択

    (3) 「Google Colaboratory」 を追加
  - 利用方法（2回目以降）

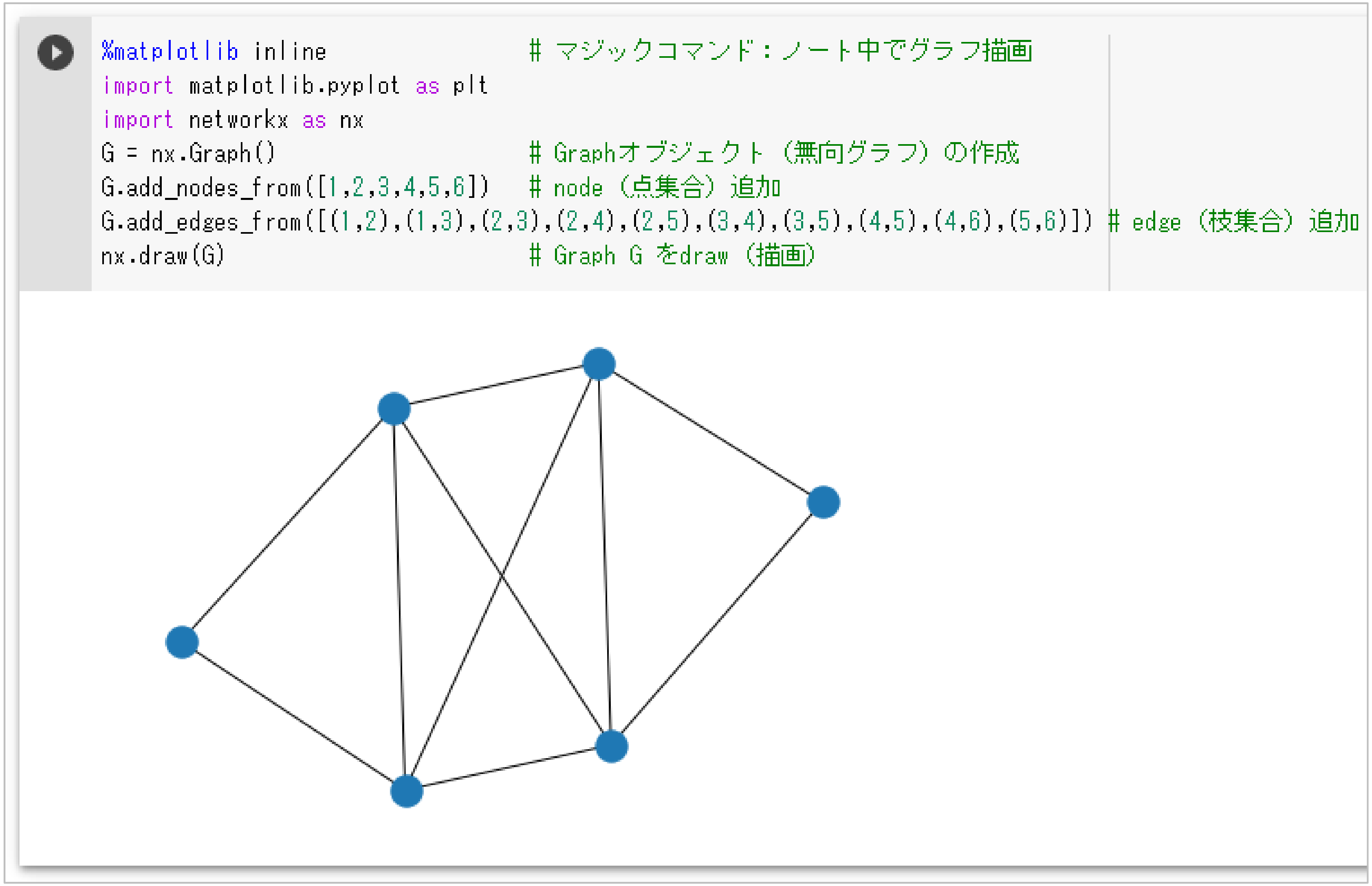
    (1) google アカウントにログインし，google drive へ移動

    (2) 「新規」ー「その他」ー「Google Colaboratory」を選択

    ※
  - ファイルは Google Drive に保存．一度作成したら，2度目からは，google drive 内のファイル [***.ipynb] を選択して，開くことができる
  - Jupyter Notebook と同様に使える
  - networkx, matplotlib などの pythonライブラリは default で使用可能

python, networkx でグラフを描画

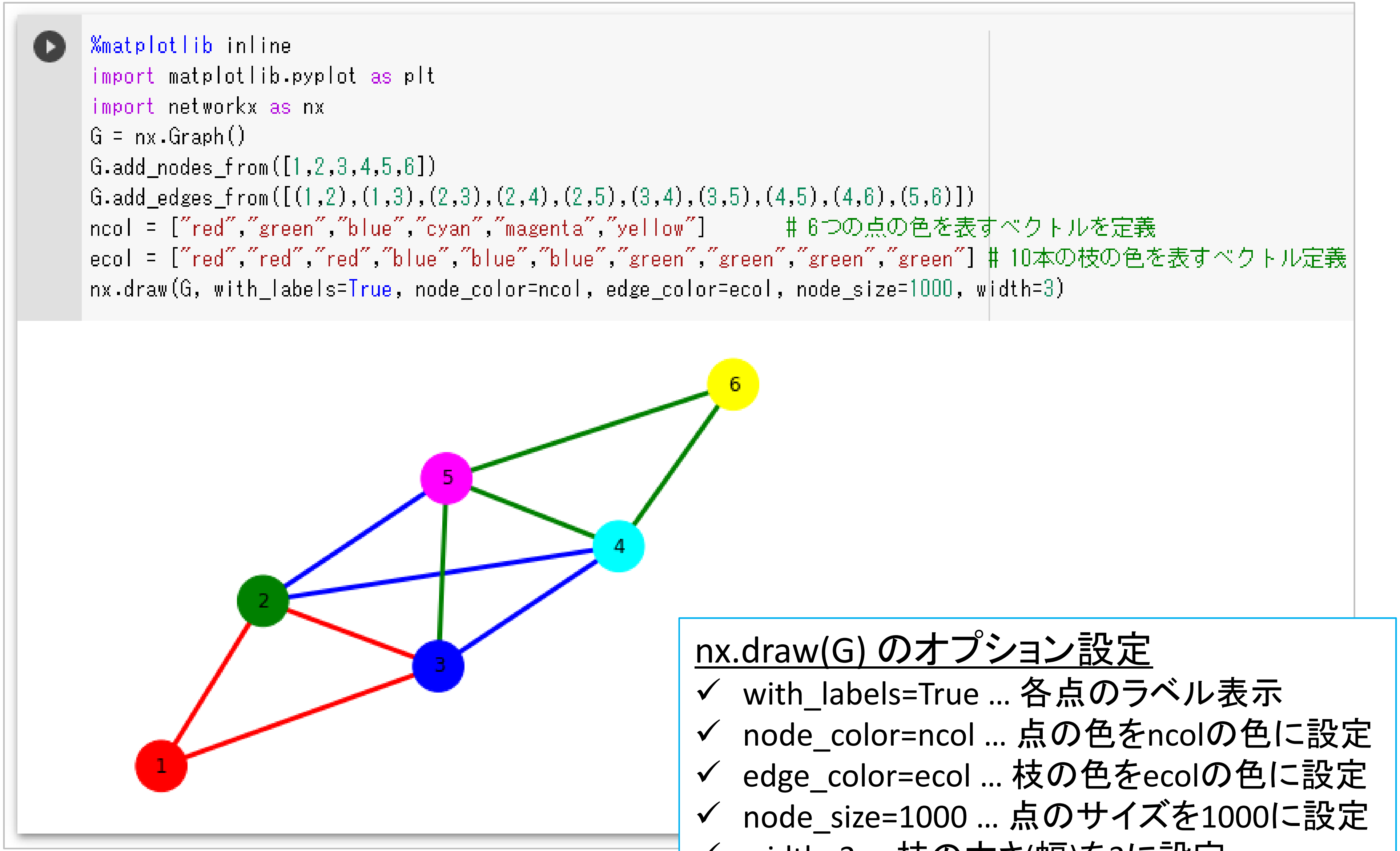
- 無向グラフ $G = (V, E)$ を描画
  - V={1,2,3,4,5,6}　　※|V|=6
  - E={(1,2),(1,3),(2,3),(2,4),(2,5),(3,4),(4,5),(4,6),(5,6)}　　※|E|=10



```
%matplotlib inline                       # マジックコマンド：ノート中でグラフ描画
import matplotlib.pyplot as plt
import networkx as nx
G = nx.Graph()                           # Graphオブジェクト（無向グラフ）の作成
G.add_nodes_from([1,2,3,4,5,6])          # node（点集合）追加
G.add_edges_from([(1,2),(1,3),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5),(4,6),(5,6)]) # edge（枝集合）追加
nx.draw(G)                               # Graph G をdraw（描画）
```

python, networkx でグラフを描画

- 無向グラフ $G = (V, E)$ を描画し装飾



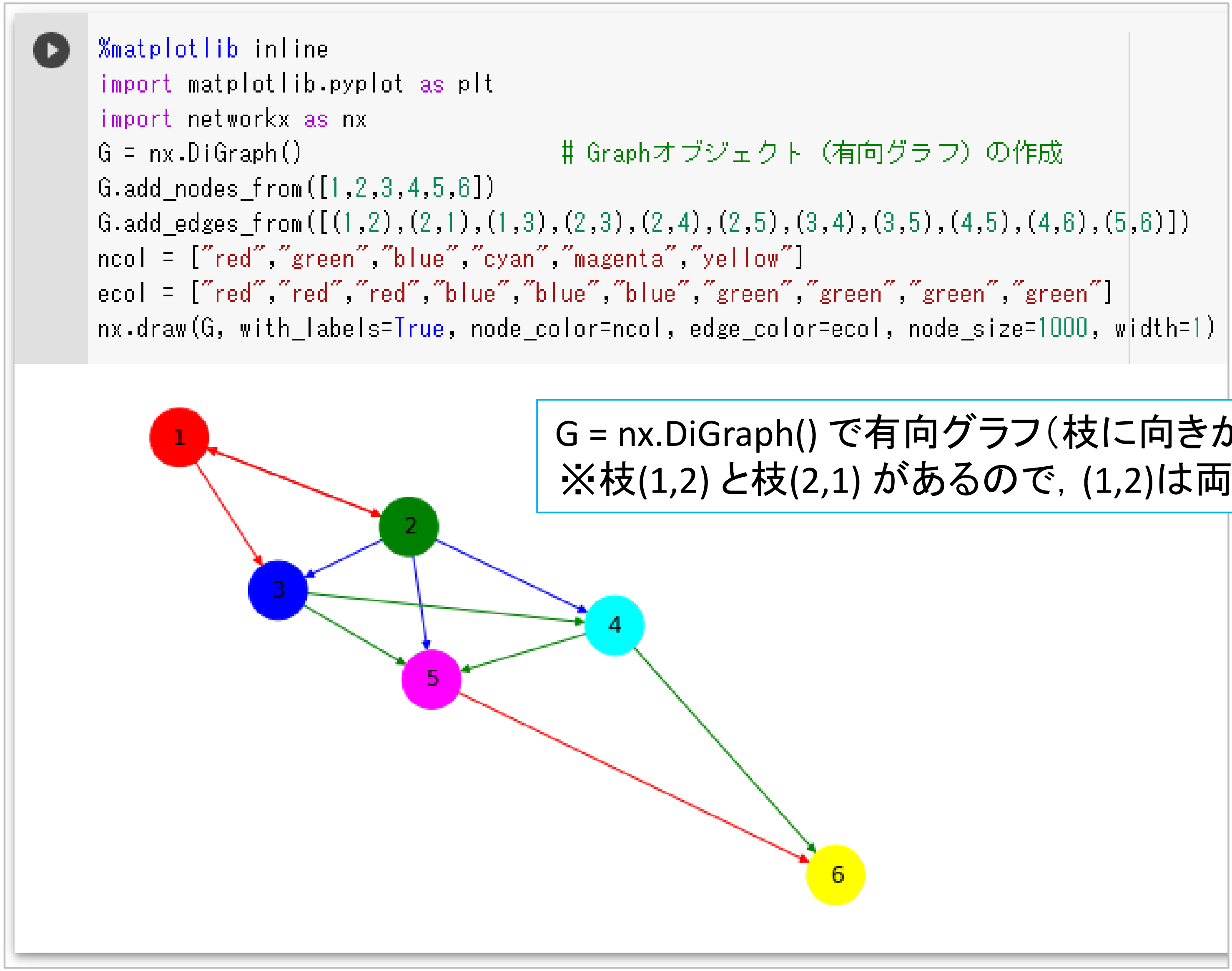
```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
G = nx.Graph()
G.add_nodes_from([1,2,3,4,5,6])
G.add_edges_from([(1,2),(1,3),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5),(4,6),(5,6)])
ncol = ["red","green","blue","cyan","magenta","yellow"]       # 6つの点の色を表すベクトルを定義
ecol = ["red","red","red","blue","blue","blue","green","green","green","green"] # 10本の枝の色を表すベクトル定義
nx.draw(G, with_labels=True, node_color=ncol, edge_color=ecol, node_size=1000, width=3)
```

nx.draw(G) のオプション設定

- ✓ with_labels=True … 各点のラベル表示
- ✓ node_color=ncol … 点の色をncolの色に設定
- ✓ edge_color=ecol … 枝の色をecolの色に設定
- ✓ node_size=1000 … 点のサイズを1000に設定
- ✓ width=3 … 枝の太さ(幅)を3に設定

python, networkx でグラフを描画

- 有向グラフ $G = (V, E)$ を描画し装飾

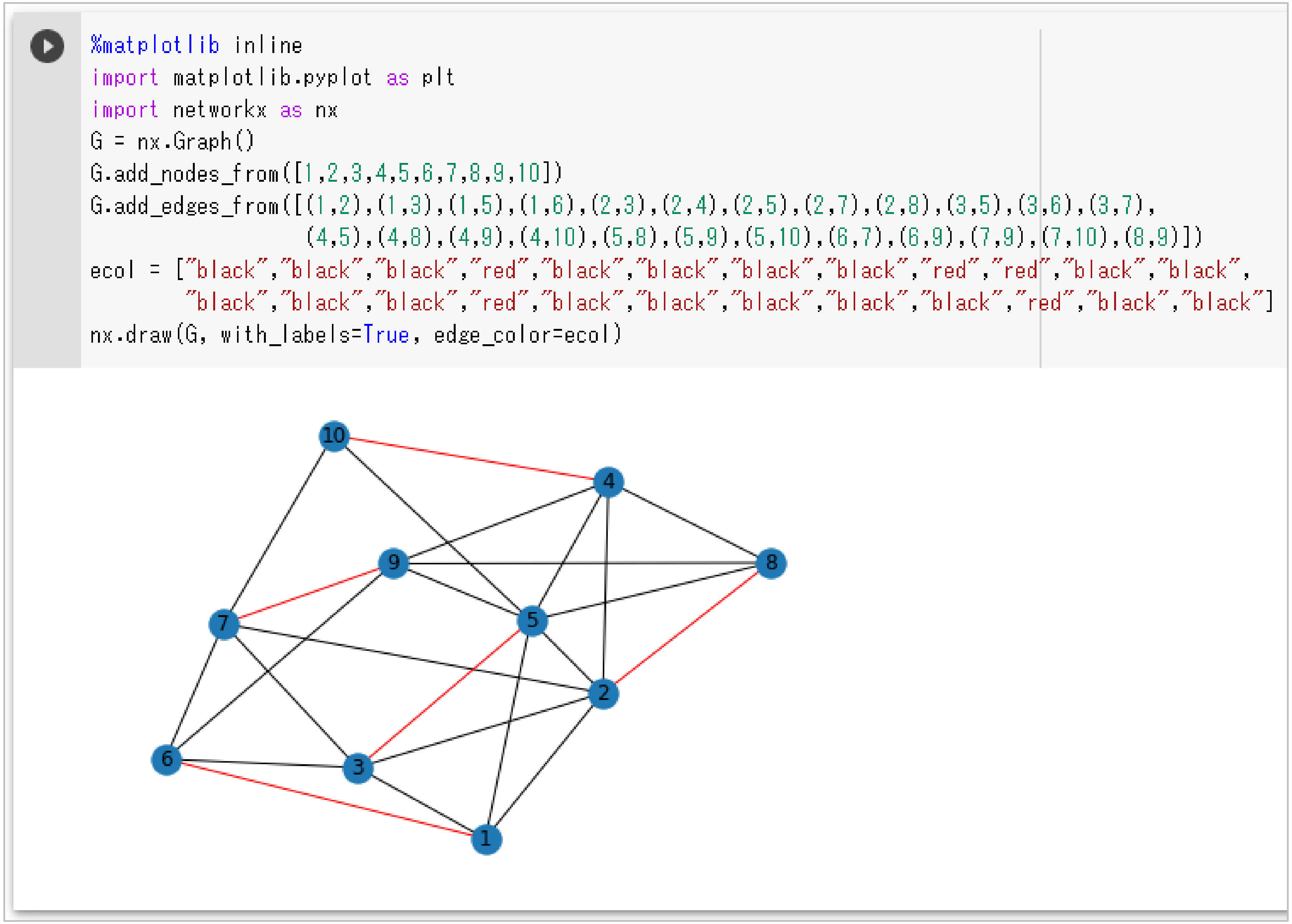


```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
G = nx.DiGraph()                              # Graphオブジェクト（有向グラフ）の作成
G.add_nodes_from([1,2,3,4,5,6])
G.add_edges_from([(1,2),(2,1),(1,3),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5),(4,6),(5,6)])
ncol = ["red","green","blue","cyan","magenta","yellow"]
ecol = ["red","red","red","blue","blue","blue","green","green","green","green"]
nx.draw(G, with_labels=True, node_color=ncol, edge_color=ecol, node_size=1000, width=1)
```

G = nx.DiGraph() で有向グラフ（枝に向きがある）を作成
※枝(1,2) と枝(2,1) があるので，(1,2)は両向き矢印⇔に

python, networkx でグラフを描画

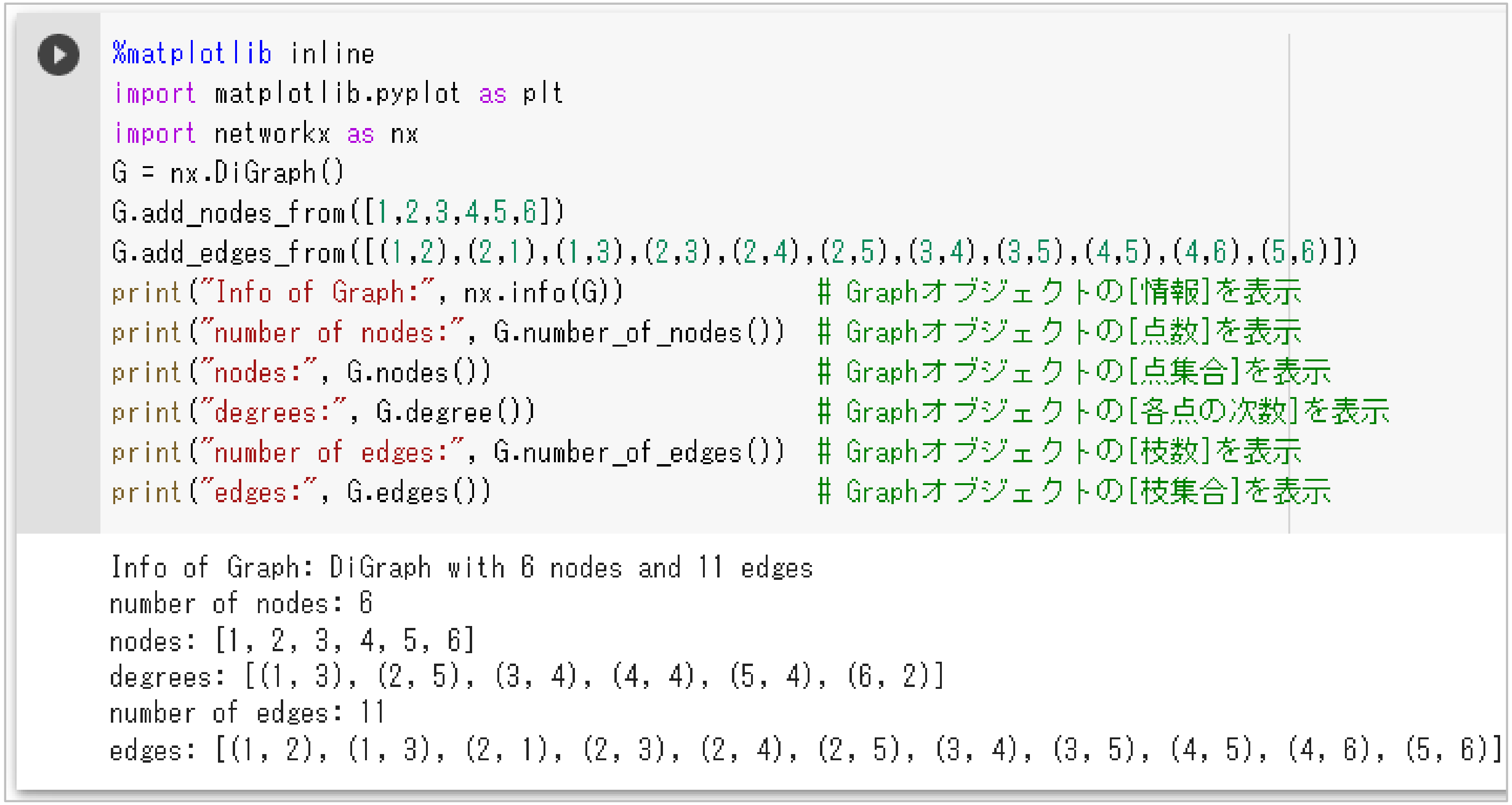
- 無向グラフ $G = (V, E)$ の最大マッチングを描画
  - 最大マッチング M={(1,6), (2,8), (3,5), (4,10), (7,9)}　※$|M|$=5　赤色の枝



```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
G = nx.Graph()
G.add_nodes_from([1,2,3,4,5,6,7,8,9,10])
G.add_edges_from([(1,2),(1,3),(1,5),(1,6),(2,3),(2,4),(2,5),(2,7),(2,8),(3,5),(3,6),(3,7),
                  (4,5),(4,8),(4,9),(4,10),(5,8),(5,9),(5,10),(6,7),(6,9),(7,9),(7,10),(8,9)])
ecol = ["black","black","black","red","black","black","black","black","red","red","black","black",
        "black","black","black","red","black","black","black","black","black","red","black","black"]
nx.draw(G, with_labels=True, edge_color=ecol)
```

python, networkx でグラフを描画

- グラフ $G = (V, E)$ オブジェクトを生成後，各種情報を取得し表示



```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
G = nx.DiGraph()
G.add_nodes_from([1,2,3,4,5,6])
G.add_edges_from([(1,2),(2,1),(1,3),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5),(4,6),(5,6)])
print("Info of Graph:", nx.info(G))            # Graphオブジェクトの[情報]を表示
print("number of nodes:", G.number_of_nodes()) # Graphオブジェクトの[点数]を表示
print("nodes:", G.nodes())                     # Graphオブジェクトの[点集合]を表示
print("degrees:", G.degree())                  # Graphオブジェクトの[各点の次数]を表示
print("number of edges:", G.number_of_edges()) # Graphオブジェクトの[枝数]を表示
print("edges:", G.edges())                     # Graphオブジェクトの[枝集合]を表示
```

```
Info of Graph: DiGraph with 6 nodes and 11 edges
number of nodes: 6
nodes: [1, 2, 3, 4, 5, 6]
degrees: [(1, 3), (2, 5), (3, 4), (4, 4), (5, 4), (6, 2)]
number of edges: 11
edges: [(1, 2), (1, 3), (2, 1), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5), (4, 6), (5, 6)]
```

print( ) 関数は，python の命令文で，括弧内( )のものを画面に表示する

✓ ダブル・クォーテーション(” “)で囲まれた部分は「文字列」で，そのまま画面に表示

✓ 複数のものを表示する場合は，コンマ(,)で区切る

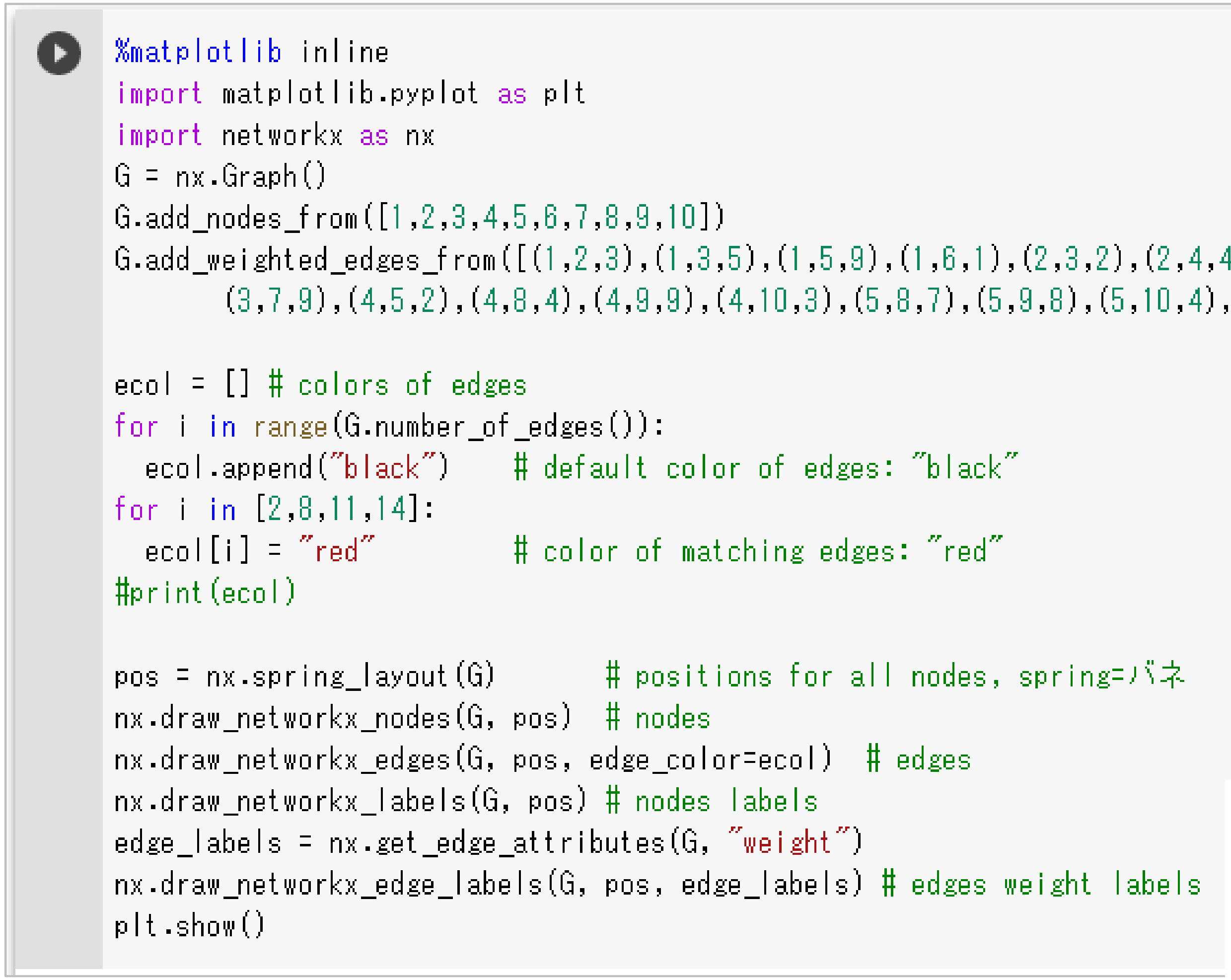
✓ 例えば，nx.info(G) は，グラフオブジェクト G の情報(6点11枝の有向グラフ)を取得

※print( ) 文の内部に書かれているので取得した情報を画面に表示

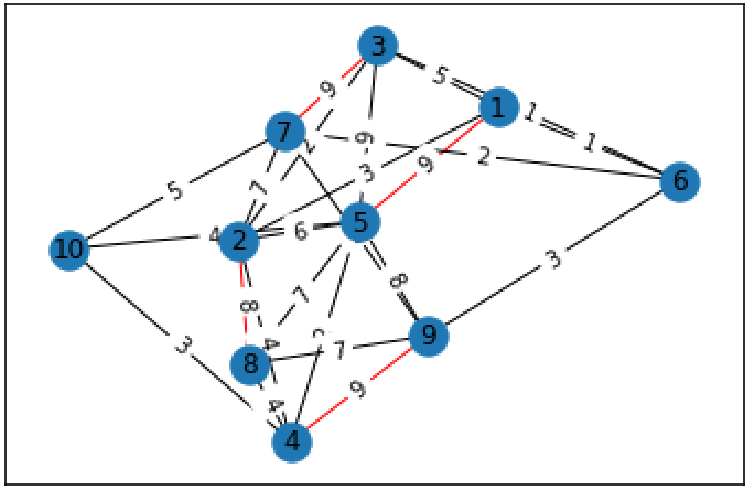
✓ 例えば，degrees: [(1,3), (2,5),…] は「点1の次数が3で，点2の次数が5で，…」という意味

python, networkx でグラフを描画

➢ 枝に重みがあるグラフ（最大重みマッチング）の描画



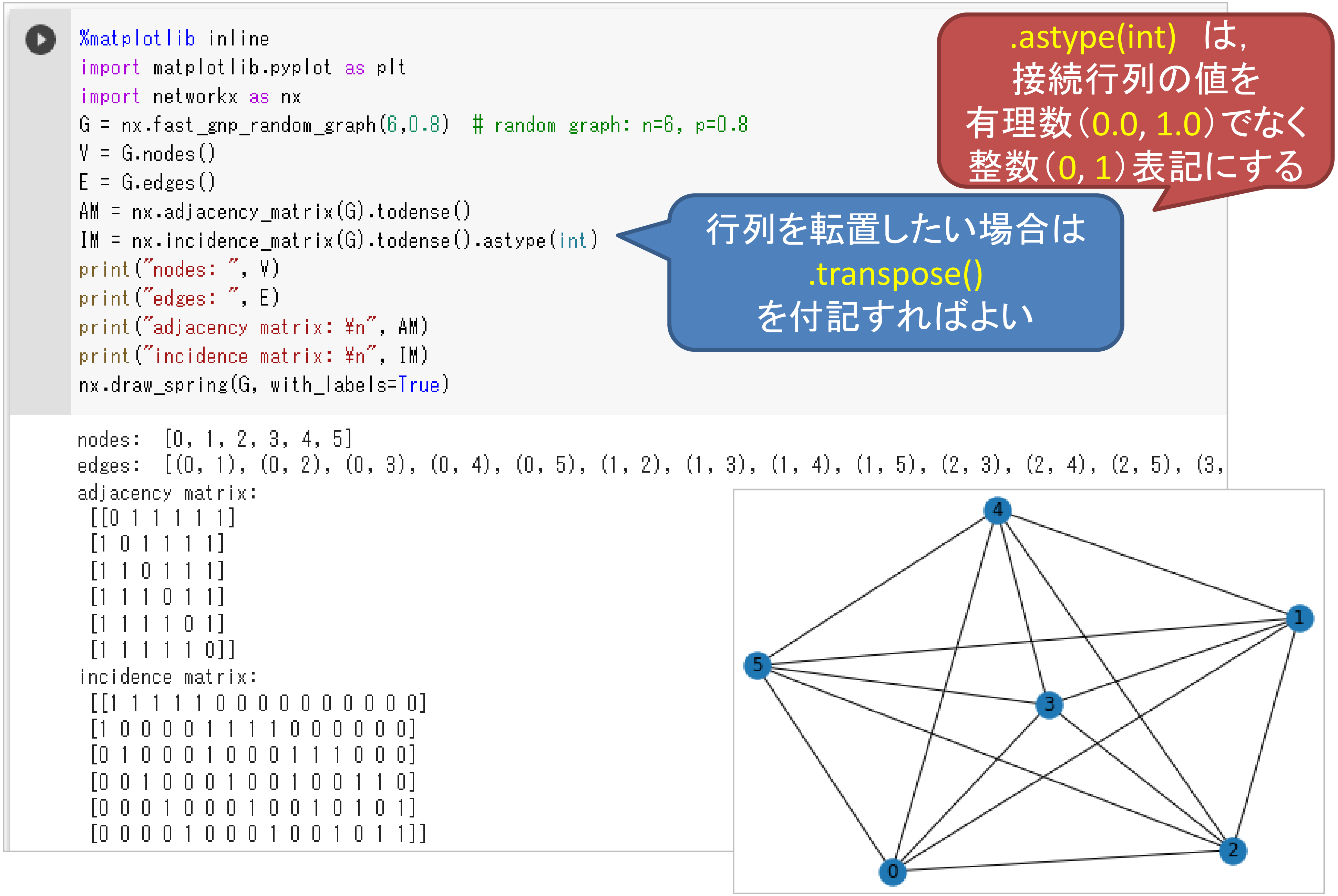
```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
G = nx.Graph()
G.add_nodes_from([1,2,3,4,5,6,7,8,9,10])
G.add_weighted_edges_from([(1,2,3),(1,3,5),(1,5,9),(1,6,1),(2,3,2),(2,4,4),(2,5,6),(2,7,7),(2,8,8),(3,5,9),(3,6,1),
       (3,7,9),(4,5,2),(4,8,4),(4,9,9),(4,10,3),(5,8,7),(5,9,8),(5,10,4),(6,7,2),(6,9,3),(7,9,5),(7,10,5),(8,9,7)])

ecol = [] # colors of edges
for i in range(G.number_of_edges()):
  ecol.append("black")    # default color of edges: "black"
for i in [2,8,11,14]:
  ecol[i] = "red"         # color of matching edges: "red"
#print(ecol)

pos = nx.spring_layout(G)      # positions for all nodes, spring=バネ
nx.draw_networkx_nodes(G, pos)  # nodes
nx.draw_networkx_edges(G, pos, edge_color=ecol)  # edges
nx.draw_networkx_labels(G, pos) # nodes labels
edge_labels = nx.get_edge_attributes(G, "weight")
nx.draw_networkx_edge_labels(G, pos, edge_labels) # edges weight labels
plt.show()
```



python, networkx でグラフを描画

- ランダムグラフの生成から，隣接行列/接続行列の取得まで

.todense() は，疎行列ではなく密行列にする



```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
G = nx.fast_gnp_random_graph(6,0.8)  # random graph: n=6, p=0.8
V = G.nodes()
E = G.edges()
AM = nx.adjacency_matrix(G).todense()
IM = nx.incidence_matrix(G).todense().astype(int)
print("nodes: ", V)
print("edges: ", E)
print("adjacency matrix: ¥n", AM)
print("incidence matrix: ¥n", IM)
nx.draw_spring(G, with_labels=True)
```

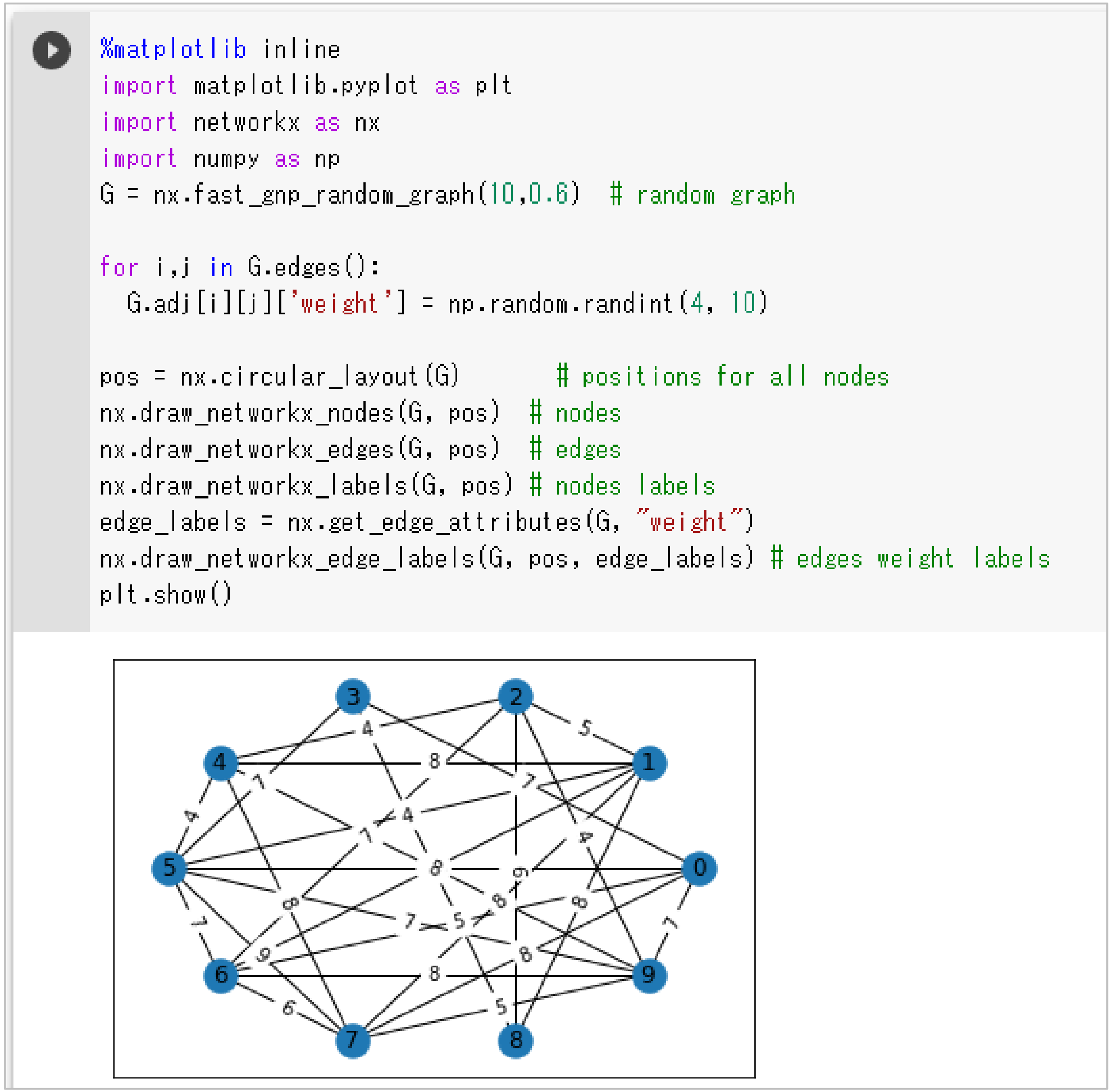
```
nodes:  [0, 1, 2, 3, 4, 5]
edges:  [(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (1, 2), (1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (3,
adjacency matrix:
 [[0 1 1 1 1 1]
 [1 0 1 1 1 1]
 [1 1 0 1 1 1]
 [1 1 1 0 1 1]
 [1 1 1 1 0 1]
 [1 1 1 1 1 0]]
incidence matrix:
 [[1 1 1 1 1 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 1 1 1 1 0 0 0 0 0 0]
 [0 1 0 0 0 1 0 0 0 1 1 1 0 0 0]
 [0 0 1 0 0 0 1 0 0 1 0 0 1 1 0]
 [0 0 0 1 0 0 0 1 0 0 1 0 1 0 1]
 [0 0 0 0 1 0 0 0 1 0 0 1 0 1 1]]
```

.astype(int) は，接続行列の値を有理数（0.0, 1.0）でなく整数（0, 1）表記にする

行列を転置したい場合は .transpose() を付記すればよい

python, networkx でグラフを描画

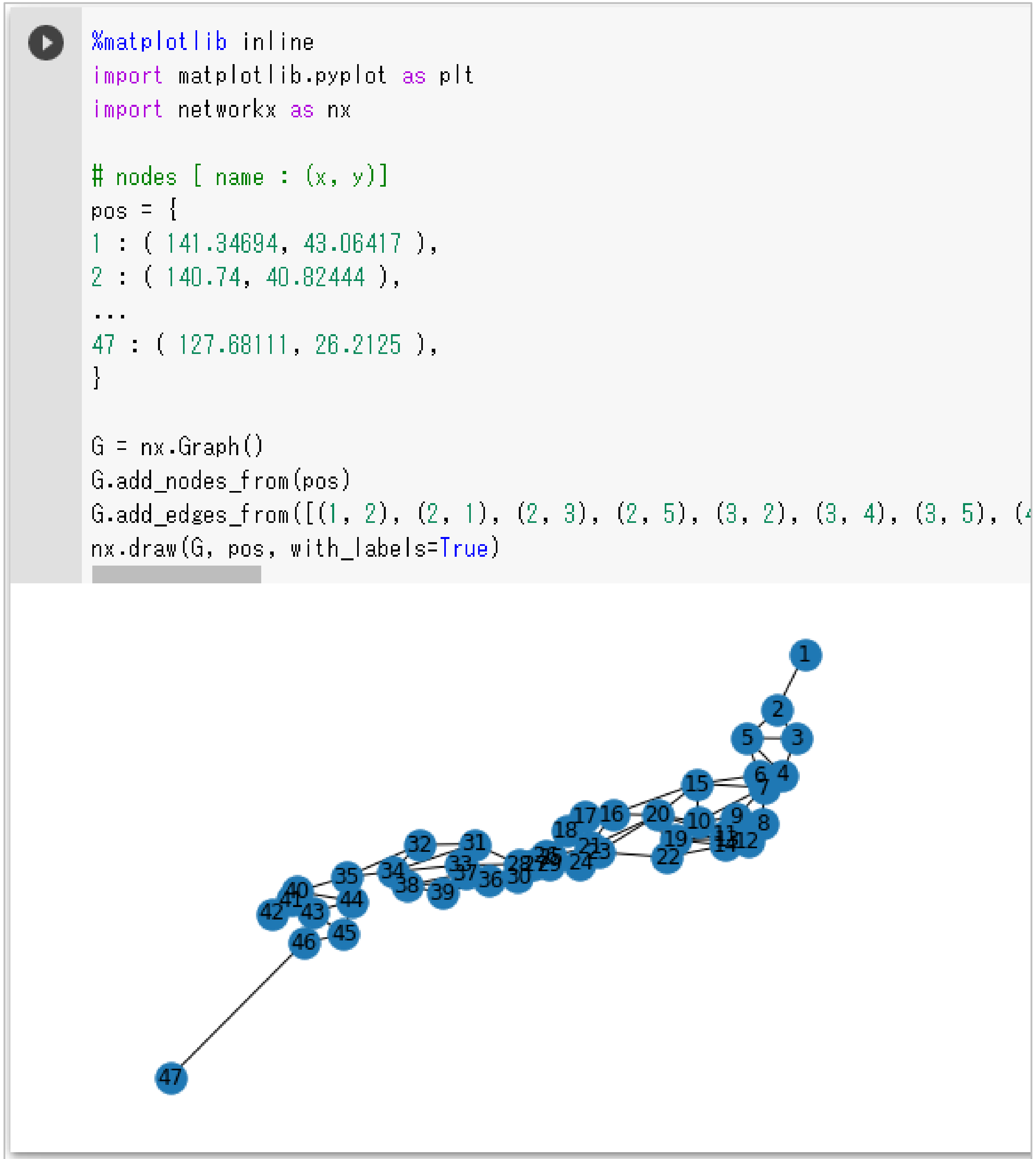
- 重み付きランダムグラフの生成



```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
G = nx.fast_gnp_random_graph(10,0.6)  # random graph

for i,j in G.edges():
  G.adj[i][j]['weight'] = np.random.randint(4, 10)

pos = nx.circular_layout(G)       # positions for all nodes
nx.draw_networkx_nodes(G, pos)  # nodes
nx.draw_networkx_edges(G, pos)  # edges
nx.draw_networkx_labels(G, pos) # nodes labels
edge_labels = nx.get_edge_attributes(G, "weight")
nx.draw_networkx_edge_labels(G, pos, edge_labels) # edges weight labels
plt.show()
```

python, networkx でグラフを描画

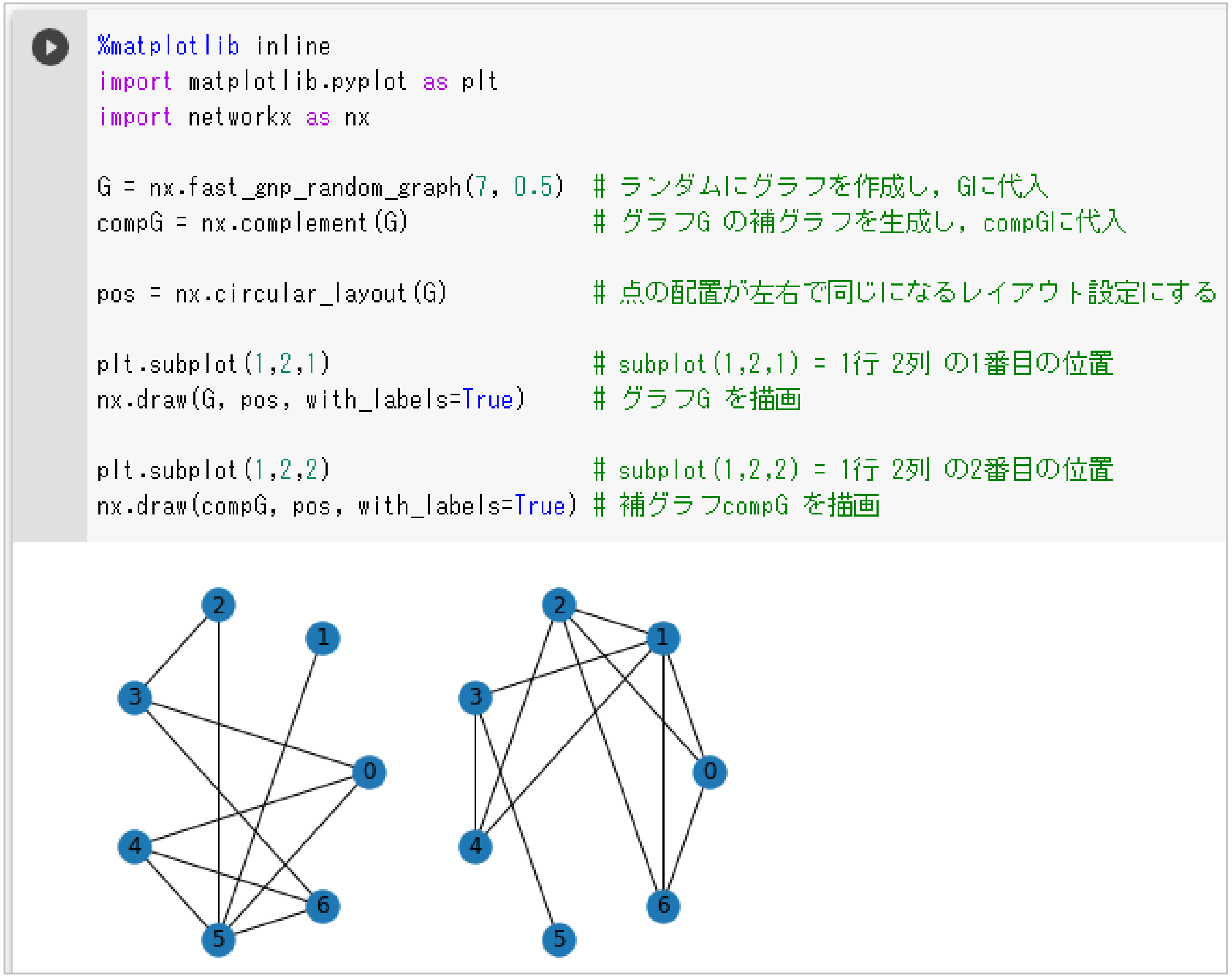
- 位置情報（(*x*, *y*)座標）付きグラフの描画



```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx

# nodes [ name : (x, y)]
pos = {
1 : ( 141.34694, 43.06417 ),
2 : ( 140.74, 40.82444 ),
...
47 : ( 127.68111, 26.2125 ),
}

G = nx.Graph()
G.add_nodes_from(pos)
G.add_edges_from([(1, 2), (2, 1), (2, 3), (2, 5), (3, 2), (3, 4), (3, 5), (
nx.draw(G, pos, with_labels=True)
```

python, networkx でグラフを描画

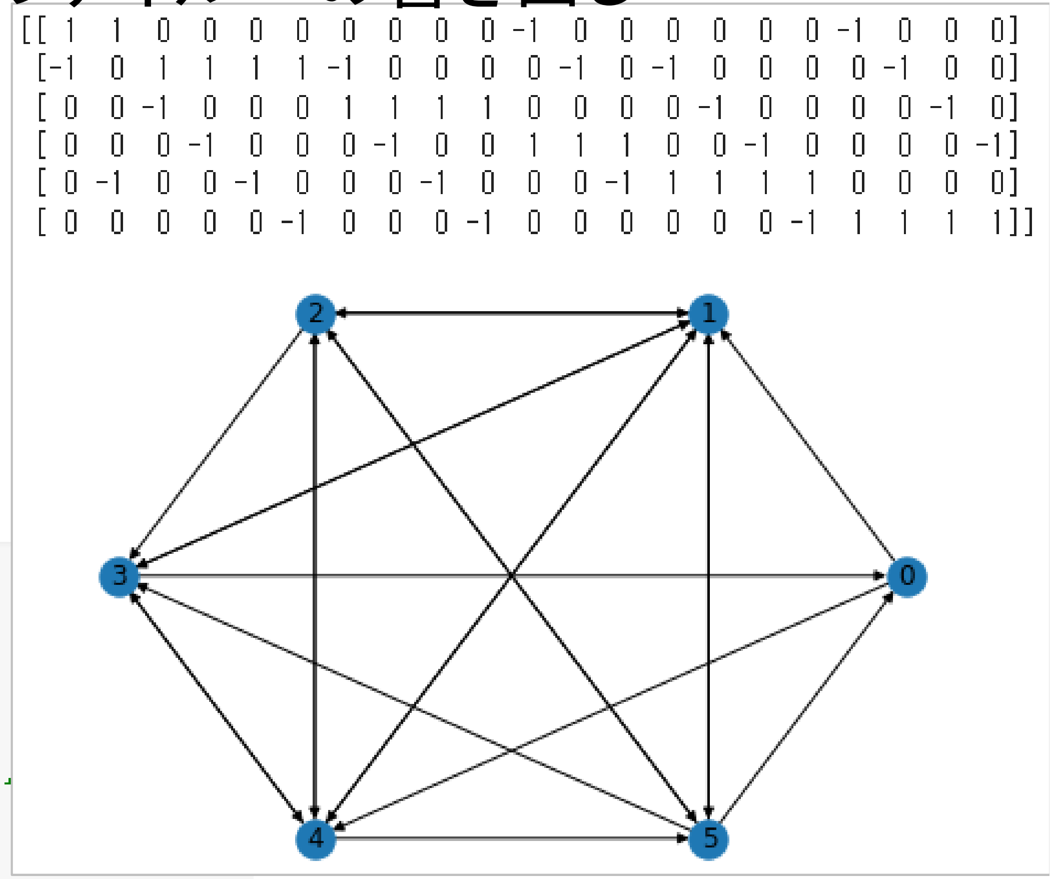
- 無向グラフと補グラフの生成，左右に並べて描画



```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx

G = nx.fast_gnp_random_graph(7, 0.5)  # ランダムにグラフを作成し，Gに代入
compG = nx.complement(G)              # グラフG の補グラフを生成し，compGに代入

pos = nx.circular_layout(G)           # 点の配置が左右で同じになるレイアウト設定にする

plt.subplot(1,2,1)                    # subplot(1,2,1) = 1行 2列 の1番目の位置
nx.draw(G, pos, with_labels=True)     # グラフG を描画

plt.subplot(1,2,2)                    # subplot(1,2,2) = 1行 2列 の2番目の位置
nx.draw(compG, pos, with_labels=True) # 補グラフcompG を描画
```

python, networkx でグラフを描画

- ➢ 有向グラフと接続行列，テキストファイルへの書き出し



```
[[ 1  1  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0 -1  0  0  0]
 [-1  0  1  1  1  1 -1  0  0  0  0 -1  0 -1  0  0  0  0 -1  0  0]
 [ 0  0 -1  0  0  0  1  1  1  1  0  0  0  0 -1  0  0  0  0 -1  0]
 [ 0  0  0 -1  0  0  0 -1  0  0  1  1  1  0  0 -1  0  0  0  0 -1]
 [ 0 -1  0  0 -1  0  0  0 -1  0  0  0 -1  1  1  1  1  0  0  0  0]
 [ 0  0  0  0  0 -1  0  0  0 -1  0  0  0  0  0  0 -1  1  1  1  1]]
```

有向グラフを指定

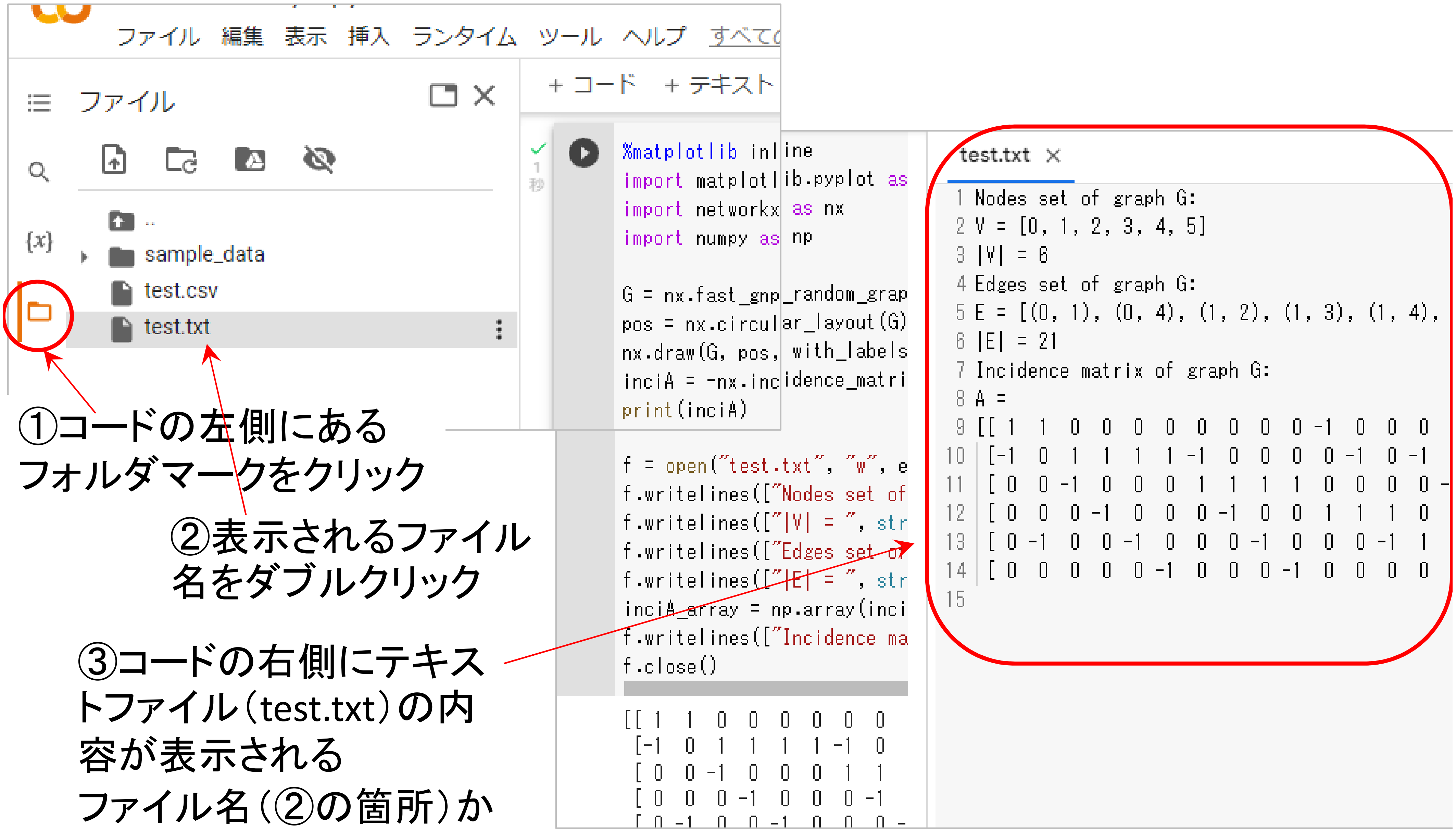
```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np

G = nx.fast_gnp_random_graph(6, 0.6, directed='True') # ランダ
pos = nx.circular_layout(G)
nx.draw(G, pos, with_labels=True)            # グラフG 描画
inciA = -nx.incidence_matrix(G, oriented=True).todense().astype(int) # 接続行列
print(inciA)

f = open("test.txt", "w", encoding="Shift-JIS")
f.writelines(["Nodes set of graph G:¥nV = ", str(G.nodes()), "¥n"])
f.writelines(["|V| = ", str(G.number_of_nodes()), "¥n"])
f.writelines(["Edges set of graph G:¥nE = ", str(G.edges()), "¥n"])
f.writelines(["|E| = ", str(G.number_of_edges()), "¥n"])
inciA_array = np.array(inciA)
f.writelines(["Incidence matrix of graph G:¥nA = ¥n", str(inciA_array), "¥n"])
f.close()
```

接続行列を有向グラフ用に設定
最初に負の符号（ー）も忘れずに
※(-1)倍しないと通常と逆になる

numpy も利用

python, networkx でグラフを描画

- 書き出したテキストファイル（test.txt）の確認



```
%matplotlib inline
import matplotlib.pyplot as
import networkx as nx
import numpy as np

G = nx.fast_gnp_random_grap
pos = nx.circular_layout(G)
nx.draw(G, pos, with_labels
inciA = -nx.incidence_matri
print(inciA)

f = open("test.txt", "w", e
f.writelines(["Nodes set of
f.writelines(["|V| = ", str
f.writelines(["Edges set of
f.writelines(["|E| = ", str
inciA_array = np.array(inci
f.writelines(["Incidence ma
f.close()
```

```
[[ 1  1  0  0  0  0  0  0
 [-1  0  1  1  1  1 -1  0
 [ 0  0 -1  0  0  0  1  1
 [ 0  0  0 -1  0  0  0 -1
 [ 0 -1  0  0 -1  0  0  0 -
```

```
Nodes set of graph G:
V = [0, 1, 2, 3, 4, 5]
|V| = 6
Edges set of graph G:
E = [(0, 1), (0, 4), (1, 2), (1, 3), (1, 4),
|E| = 21
Incidence matrix of graph G:
A =
[[ 1  1  0  0  0  0  0  0  0  0 -1  0  0  0
 [-1  0  1  1  1  1 -1  0  0  0  0 -1  0 -1
 [ 0  0 -1  0  0  0  1  1  1  1  0  0  0  0 -
 [ 0  0  0 -1  0  0  0 -1  0  0  1  1  1  0
 [ 0 -1  0  0 -1  0  0  0 -1  0  0  0 -1  1
 [ 0  0  0  0  0 -1  0  0  0 -1  0  0  0  0

```

①コードの左側にあるフォルダマークをクリック

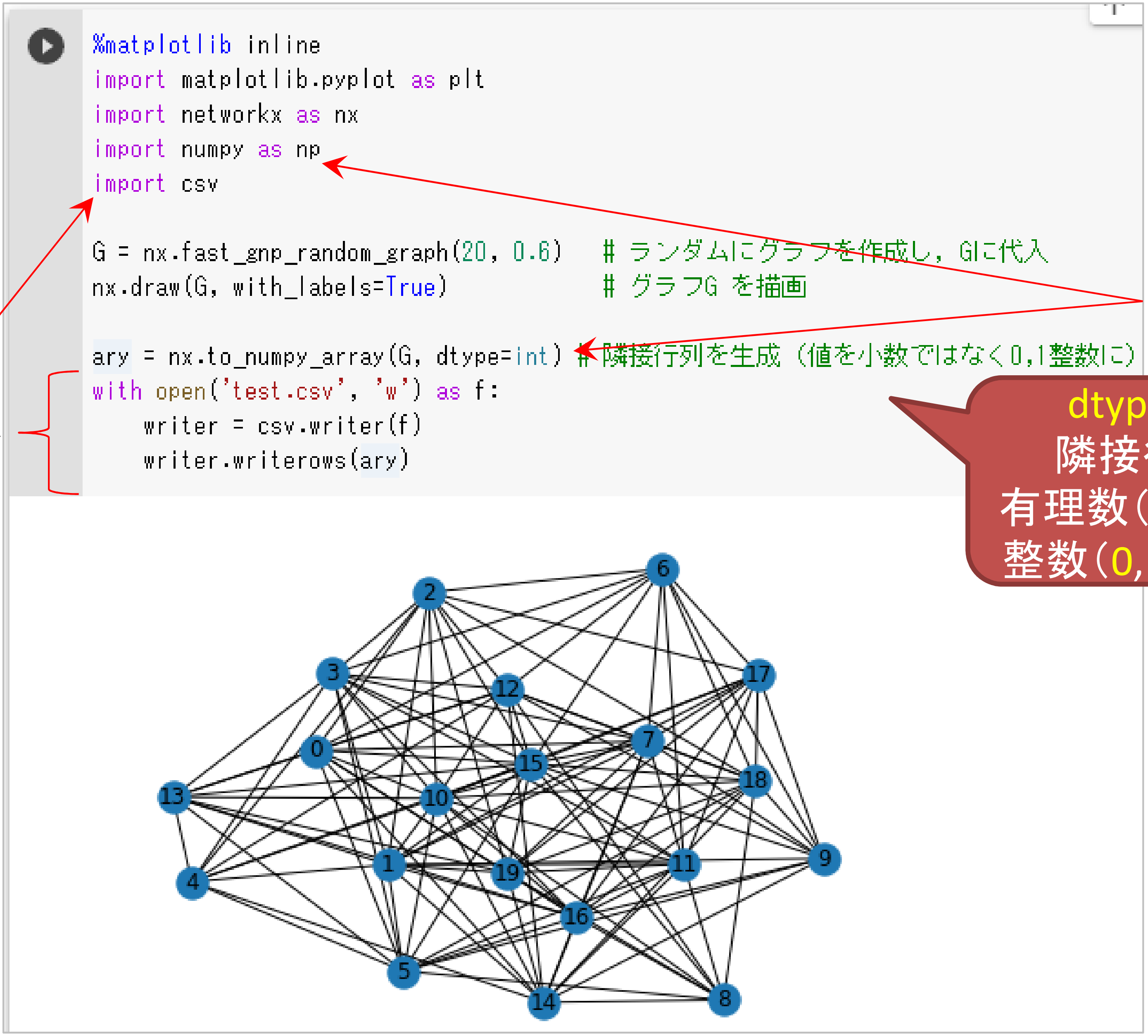
②表示されるファイル名をダブルクリック

③コードの右側にテキストファイル（test.txt）の内容が表示される

ファイル名（②の箇所）からダウンロードも可

python, networkx でグラフを描画

- 無向グラフと隣接行列，csvファイルへの書き出し



```
%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
import csv

G = nx.fast_gnp_random_graph(20, 0.6)   # ランダムにグラフを作成し，Gに代入
nx.draw(G, with_labels=True)            # グラフG を描画

ary = nx.to_numpy_array(G, dtype=int) # 隣接行列を生成 （値を小数ではなく0,1整数に)
with open('test.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerows(ary)
```

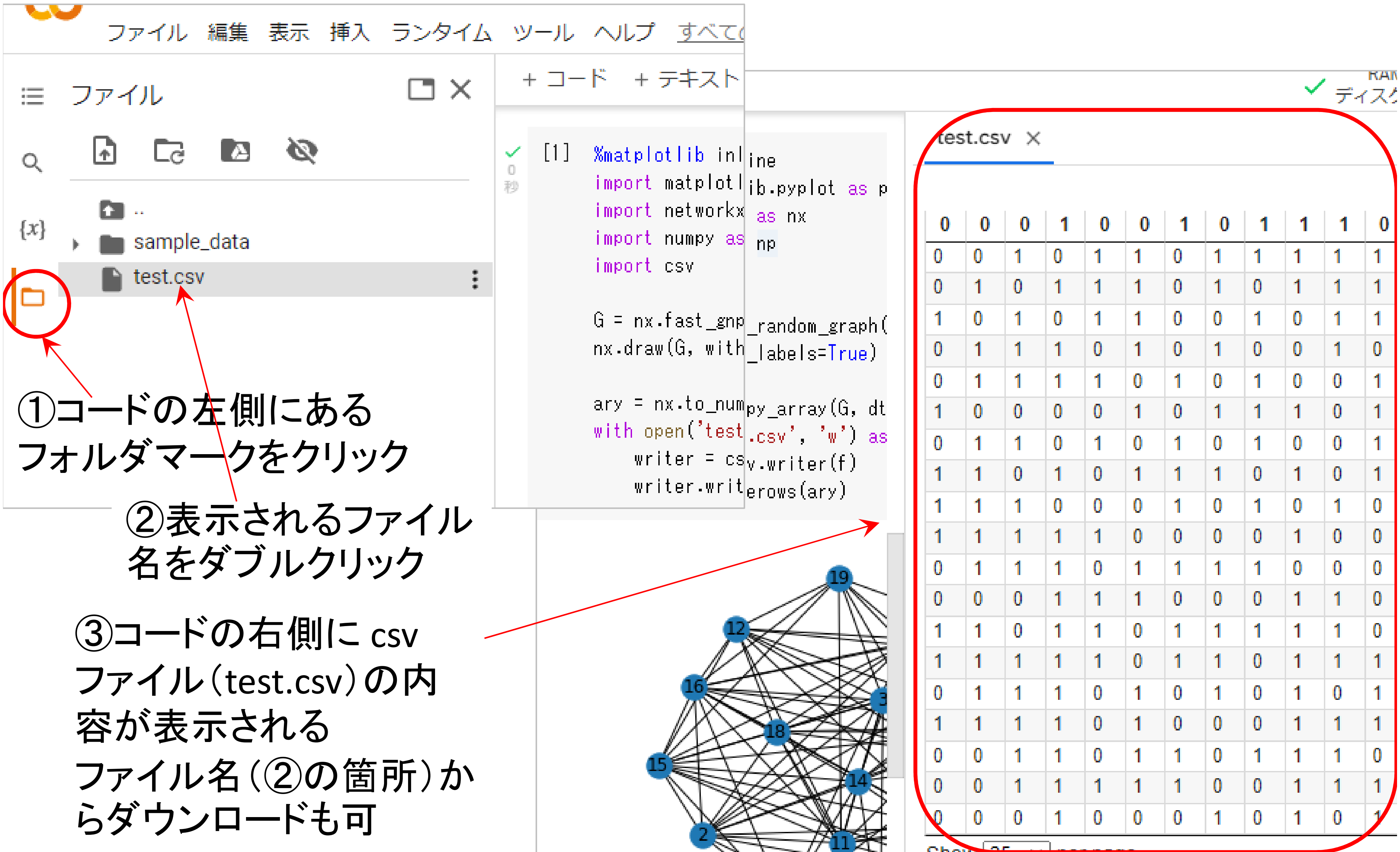
numpy も利用

csv へ書き込み

dtype=int　は，
隣接行列の値を
有理数（0.0, 1.0）でなく
整数（0, 1）表記にする

python, networkx でグラフを描画

- 書き出したcsvファイル（test.csv）の確認



```
%matplotlib inline
import matplotlib.pyplot as p
import networkx as nx
import numpy as np
import csv

G = nx.fast_gnp_random_graph(
nx.draw(G, with_labels=True)

ary = nx.to_numpy_array(G, dt
with open('test.csv', 'w') as
    writer = csv.writer(f)
    writer.writerows(ary)
```

0	0	0	1	0	0	1	0	1	1	1	0
0	0	1	0	1	1	0	1	1	1	1	1
0	1	0	1	1	1	0	1	0	1	1	1
1	0	1	0	1	1	0	0	1	0	1	1
0	1	1	1	0	1	0	1	0	0	1	0
0	1	1	1	1	0	1	0	1	0	0	1
1	0	0	0	0	1	0	1	1	1	0	1
0	1	1	0	1	0	1	0	1	0	0	1
1	1	0	1	0	1	1	1	0	1	0	1
1	1	1	0	0	0	1	0	1	0	1	0
1	1	1	1	1	0	0	0	0	1	0	0
0	1	1	1	0	1	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0	1	1	0
1	1	0	1	1	0	1	1	1	1	1	0
1	1	1	1	1	0	1	1	0	1	1	1
0	1	1	1	0	1	0	1	0	1	0	1
1	1	1	1	0	1	0	0	0	1	1	1
0	0	1	1	0	1	1	0	1	1	1	0
0	0	1	1	1	1	1	0	0	1	1	1
0	0	0	1	0	0	0	1	0	1	0	1

①コードの左側にある
フォルダマークをクリック

②表示されるファイル
名をダブルクリック

③コードの右側にcsv
ファイル（test.csv）の内
容が表示される
ファイル名（②の箇所）か
らダウンロードも可