GRAFICAR REDES CON NETWORKX

Recordemos:

Bipartita:
Una red es bipartita cuando contiene dos tipos de nodos distintos y todos los bordes conectan un nodo del primer tipo con un nodo del segundo tipo.

Dirigida:
Una red está dirigida cuando cada borde tiene una orientación, es decir, cada borde va explícitamente de un nodo a otro.

Marcas de tiempo:
Cuando una red tiene marcas de tiempo, se conoce el tiempo de creación de cada borde.

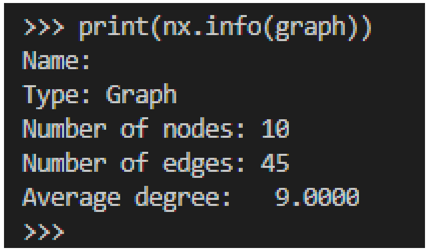
No dirigida:
Una red no está dirigida cuando sus bordes no tienen una orientación.

Unipartita:
Una red es unipartita cuando contiene un solo tipo de nodo.

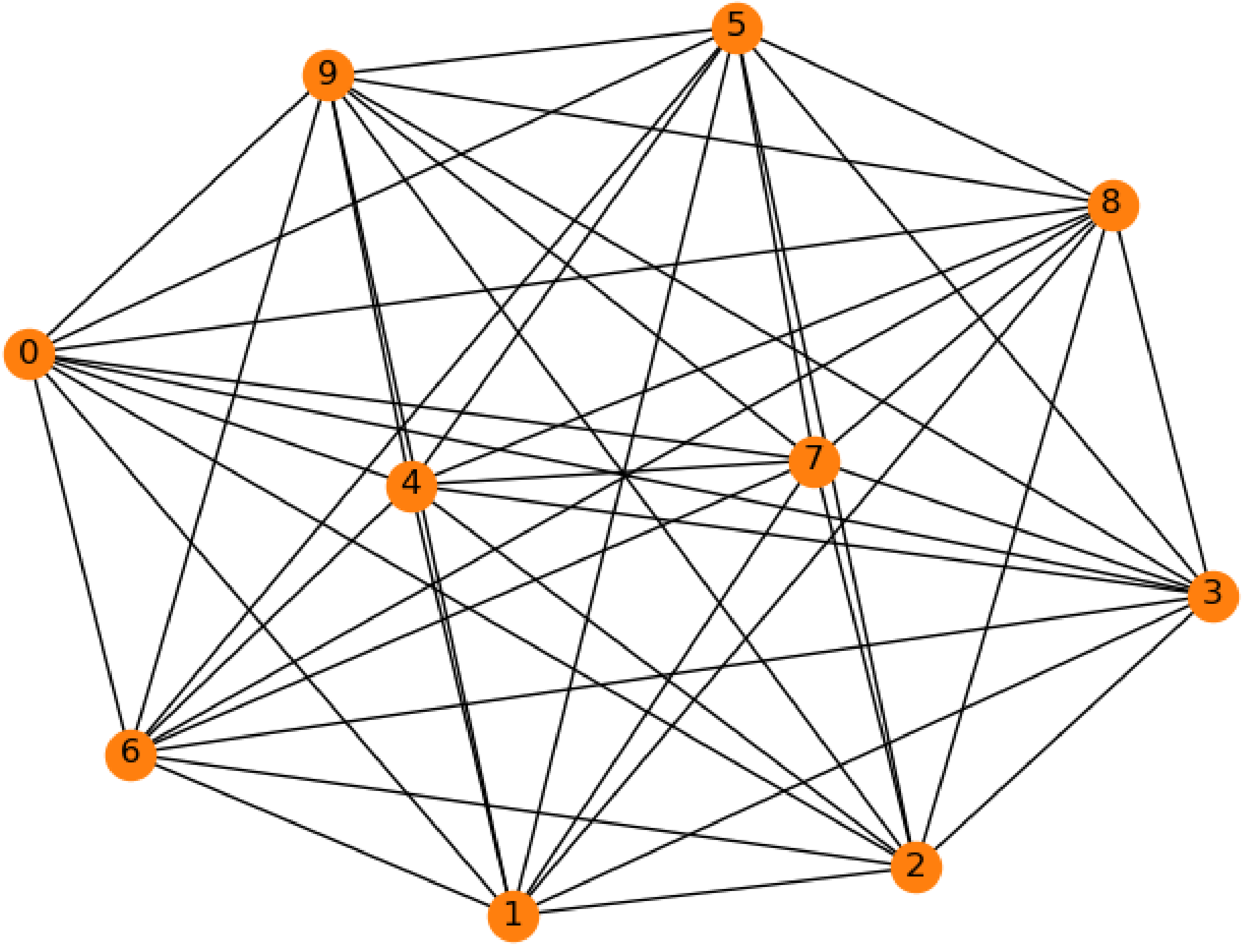
Ponderada:
Una red se pondera si sus bordes están etiquetados con pesos de borde, por ejemplo, valores de clasificación.

Generar gráficos aleatorios

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

graph = nx.complete_graph(10)

#Observo los elementos de mi red
print(nx.info(graph))

nx.draw(graph,
node_color='C1',with_labels=True)
plt.show()
```



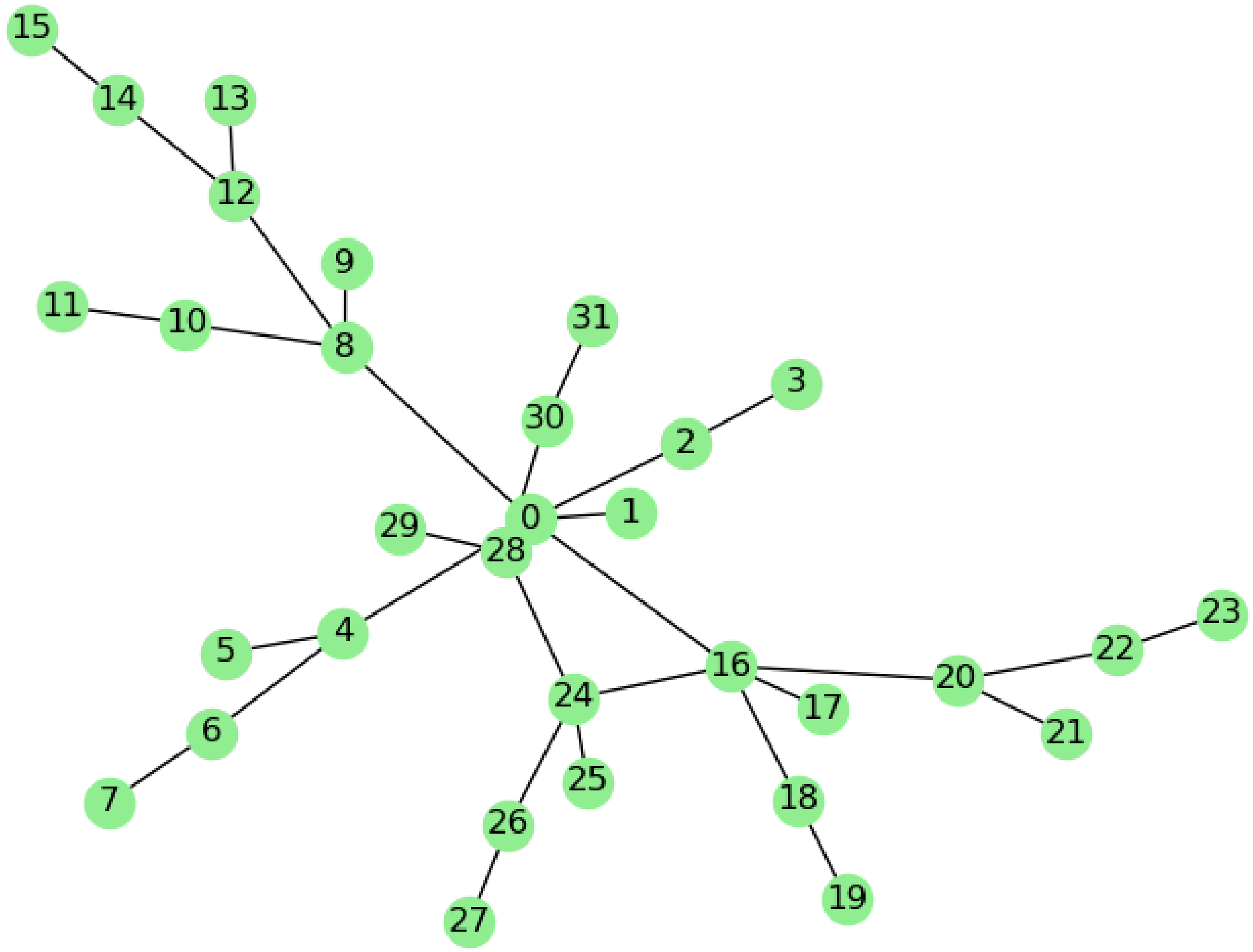
```
>>> print(nx.info(graph))
Name:
Type: Graph
Number of nodes: 10
Number of edges: 45
Average degree:   9.0000
>>>
```



Generar gráficos aleatorios

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

graph = nx.complete_graph(10)

graph = nx.binomial_tree(5)
nx.draw(graph,
node_color='lightgreen',
with_labels=True)
plt.show()
```

Devuelve un árbol binomial de orden n.

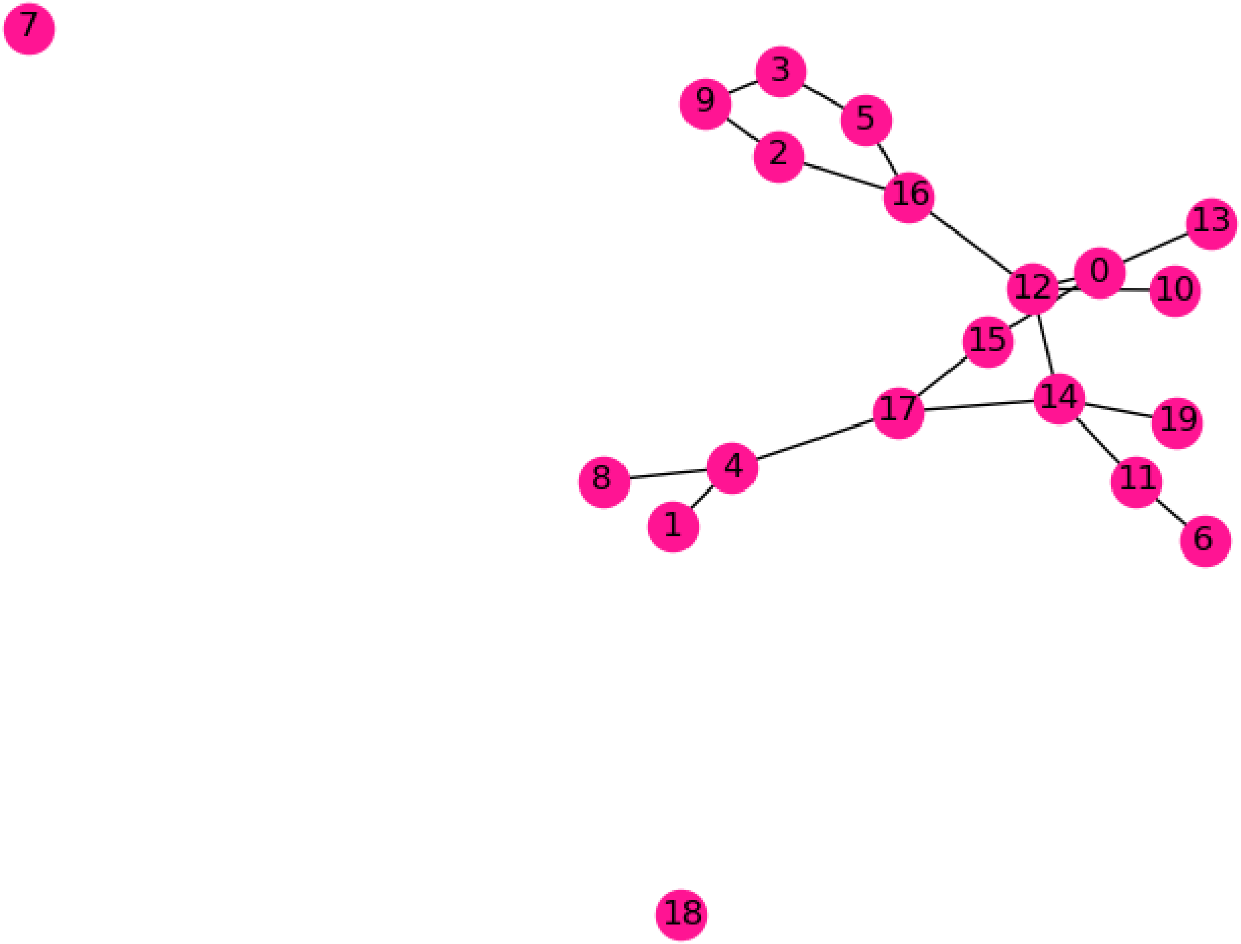


Generar gráficos aleatorios

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

graph = nx.complete_graph(10)

graph = nx.binomial_graph(20,0.15)
```

Devuelve un gráfico aleatorio, también conocido como gráfico Erdős-Rényi o gráfico binomial.

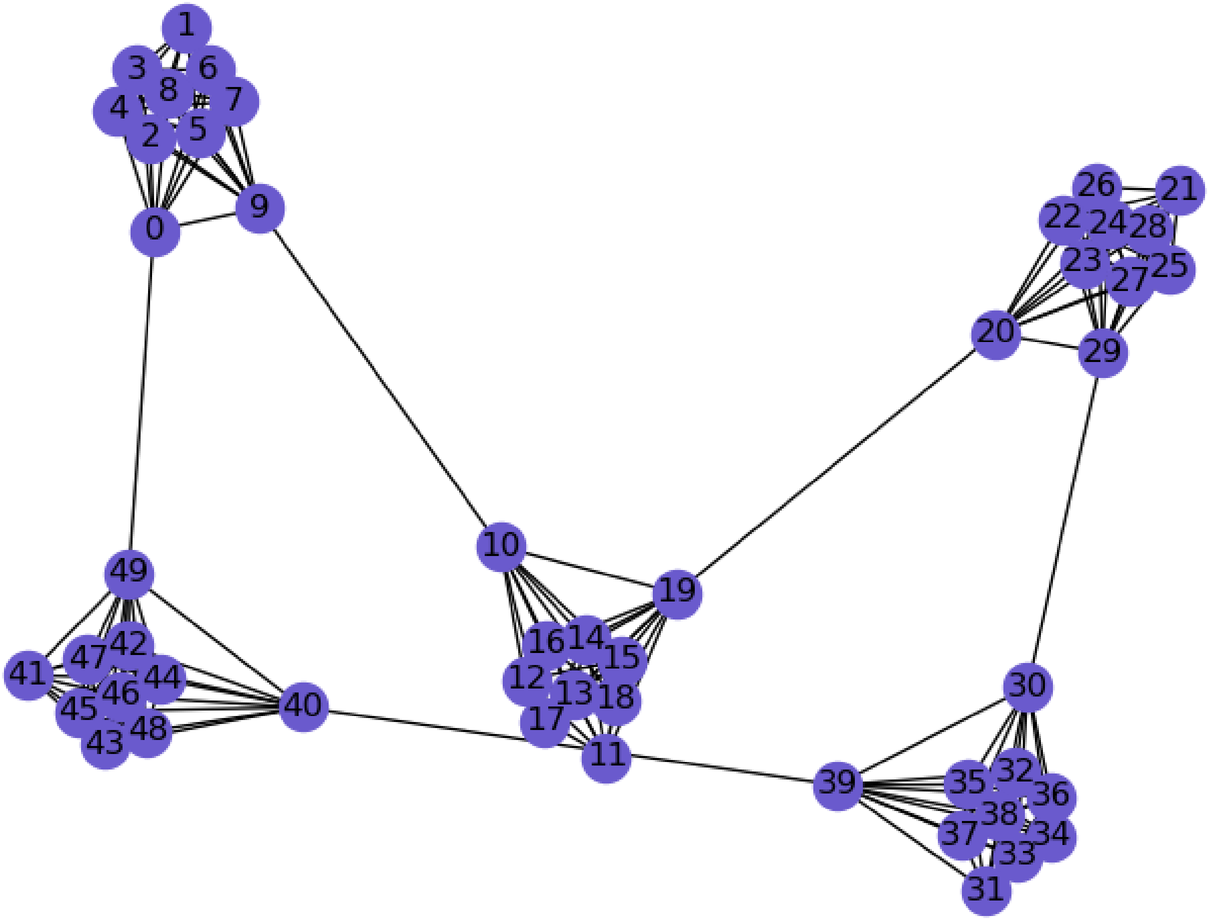
```
nx.draw(graph, node_color='#FF1493',
with_labels=True)
plt.show()
```



Generar gráficos aleatorios

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

graph = nx.complete_graph(10)
graph = nx.connected_caveman_graph(5,10)

nx.draw(graph, node_color='#6A5ACD',
with_labels=True)
plt.show()
```

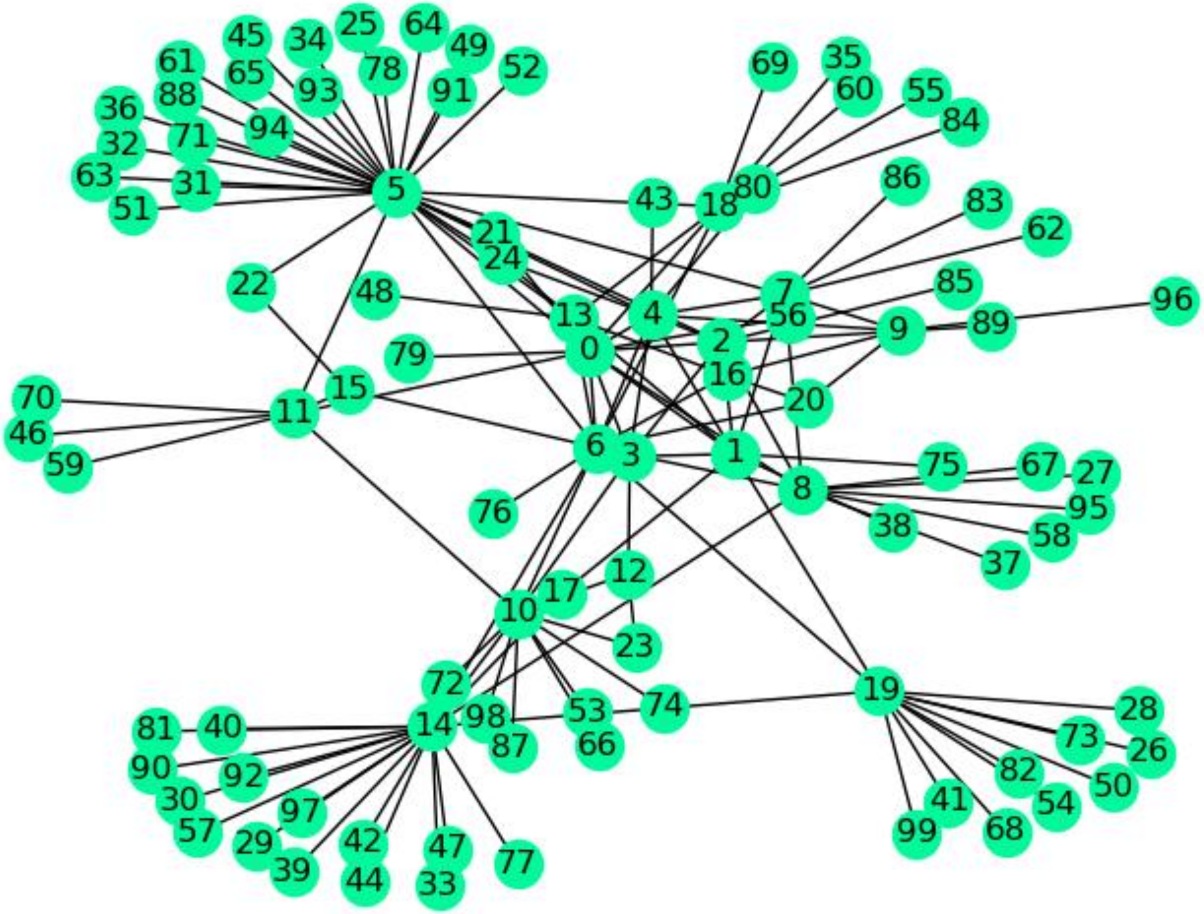
Devuelve un gráfico conectado de l grupos de tamaño k.



Generar gráficos aleatorios

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

graph = nx.complete_graph(10)

graph = nx.random_internet_as_graph(100)

nx.draw(graph, node_color='#00FA9A',
with_labels=True)
plt.show()
```

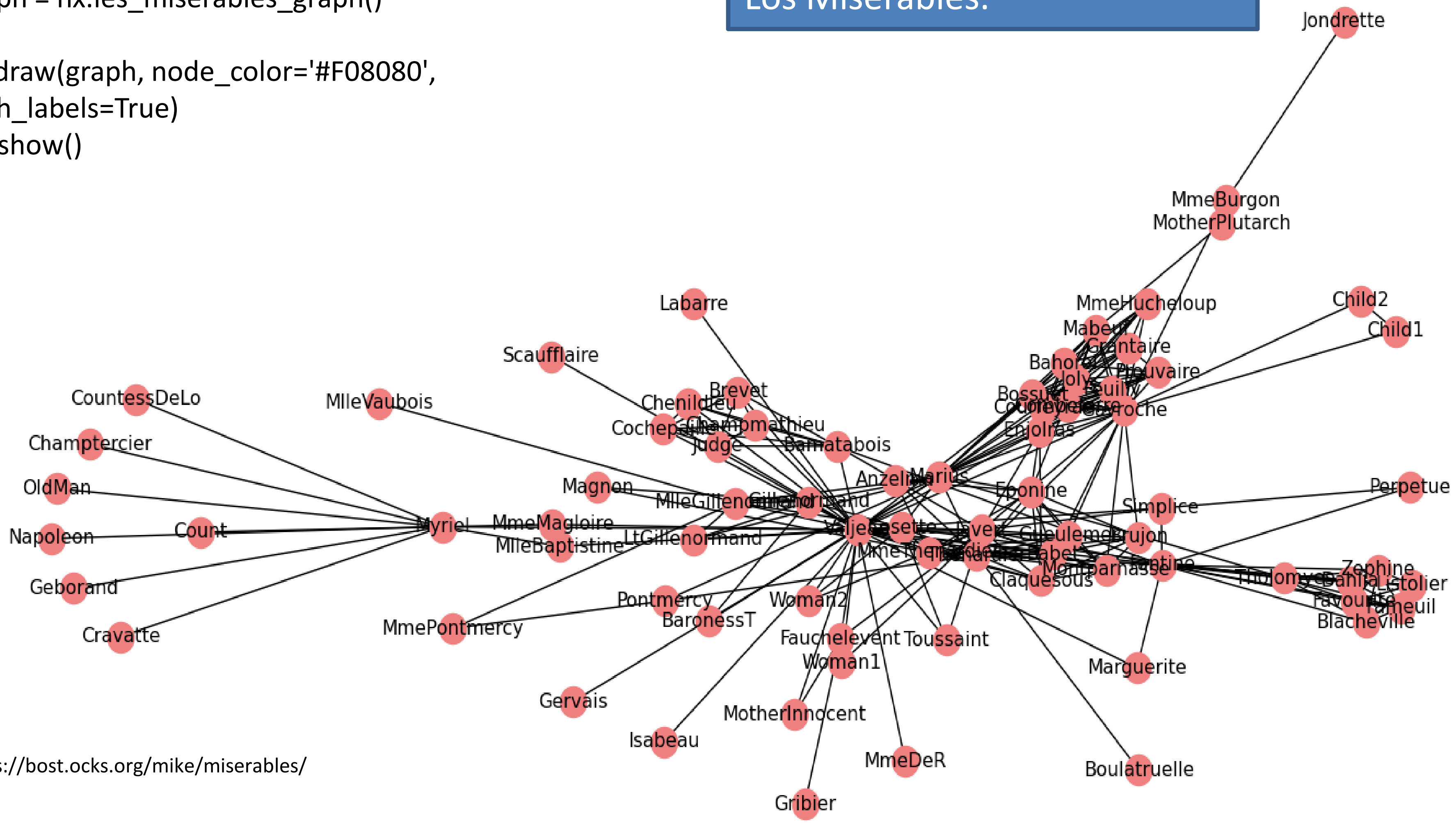
Genera un gráfico aleatorio no dirigido que se asemeja a la red de Internet



Generar gráficos aleatorios

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

plt.figure(figsize=(10,15))

graph = nx.les_miserables_graph()

nx.draw(graph, node_color='#F08080',
with_labels=True)
plt.show()
```

Devuelve la co-aparición de la red de personajes en la novela Los Miserables.

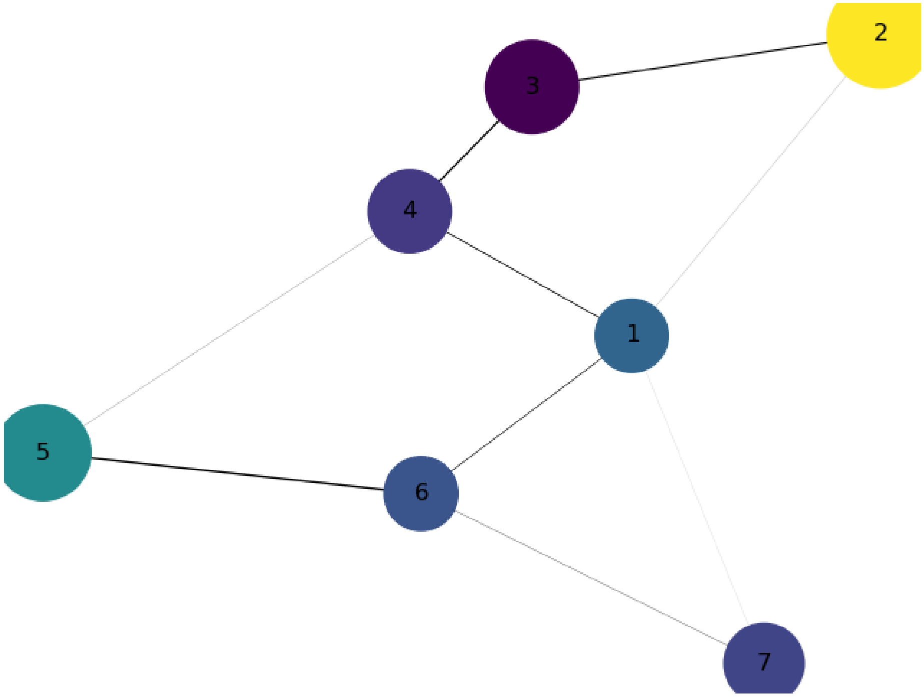


https://bost.ocks.org/mike/miserables/

Manipular atributos de nodos y bordes usando numpy

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

graph = nx.Graph()
edges = [(1, 2), (2, 3), (3, 4),
        (4, 5), (5, 6), (6, 1),
        (1, 4), (1, 7), (6, 7)]
graph.add_edges_from(edges)
graph.nodes[1]
graph.nodes[1]['category'] = 'A'
print(graph.nodes[1])

graph.edges[1, 2]
graph.edges[1, 2]['weight'] = 2
print(graph.edges[1, 2])

edge_weights = {edge: np.random.rand()
              for edge in graph.edges}

nx.set_edge_attributes(graph, edge_weights, 'weight')
graph.edges[3, 4]

node_sizes = {node: np.random.rand() * 300
            for node in graph.nodes}

nx.set_node_attributes(graph, node_sizes, 'size')
graph.nodes[5]
```

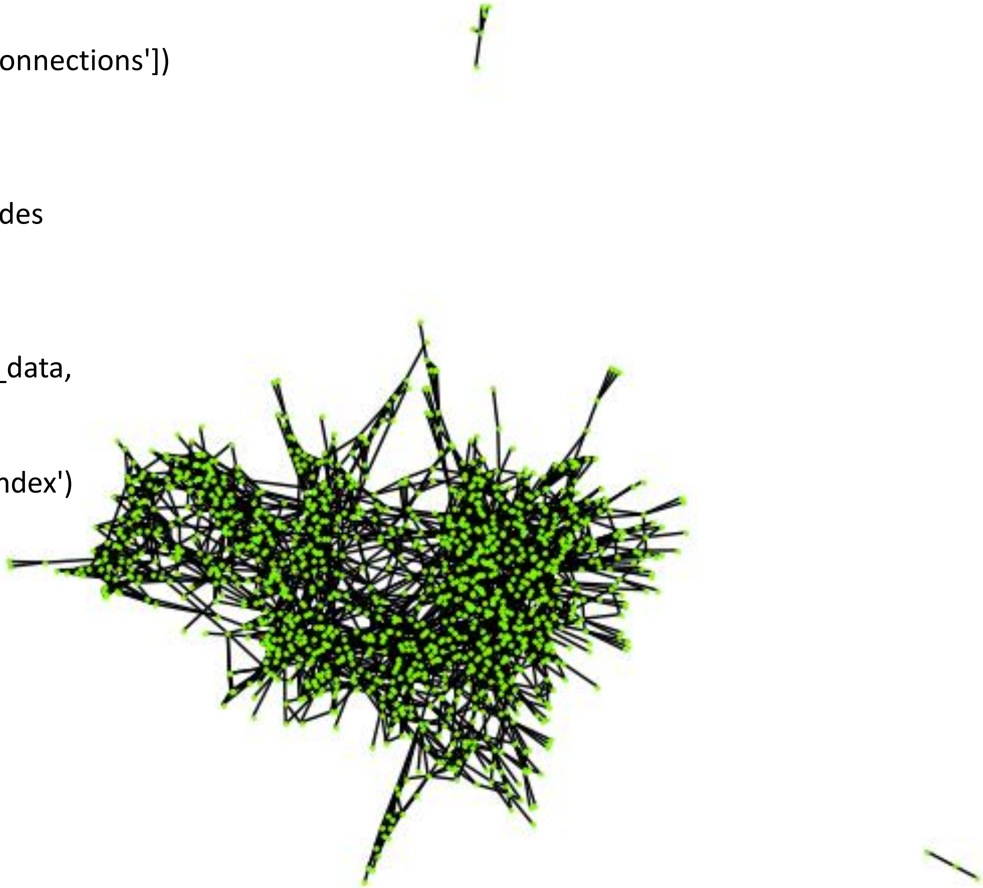
```
node_colors = {node: np.random.rand()
             for node in graph.nodes}

nx.set_node_attributes(graph, node_colors, 'color')
node_colors

width = list(nx.get_edge_attributes(graph, 'weight').values())
node_size = list(nx.get_node_attributes(graph, 'size').values())
node_color = list(nx.get_node_attributes(graph, 'color').values())

nx.draw(graph,
      width=width,
      node_size=node_size,
      node_color=node_color,
      with_labels=True)
plt.show()
```



TRABAJANDO CON CONJUNTO DE DATOS

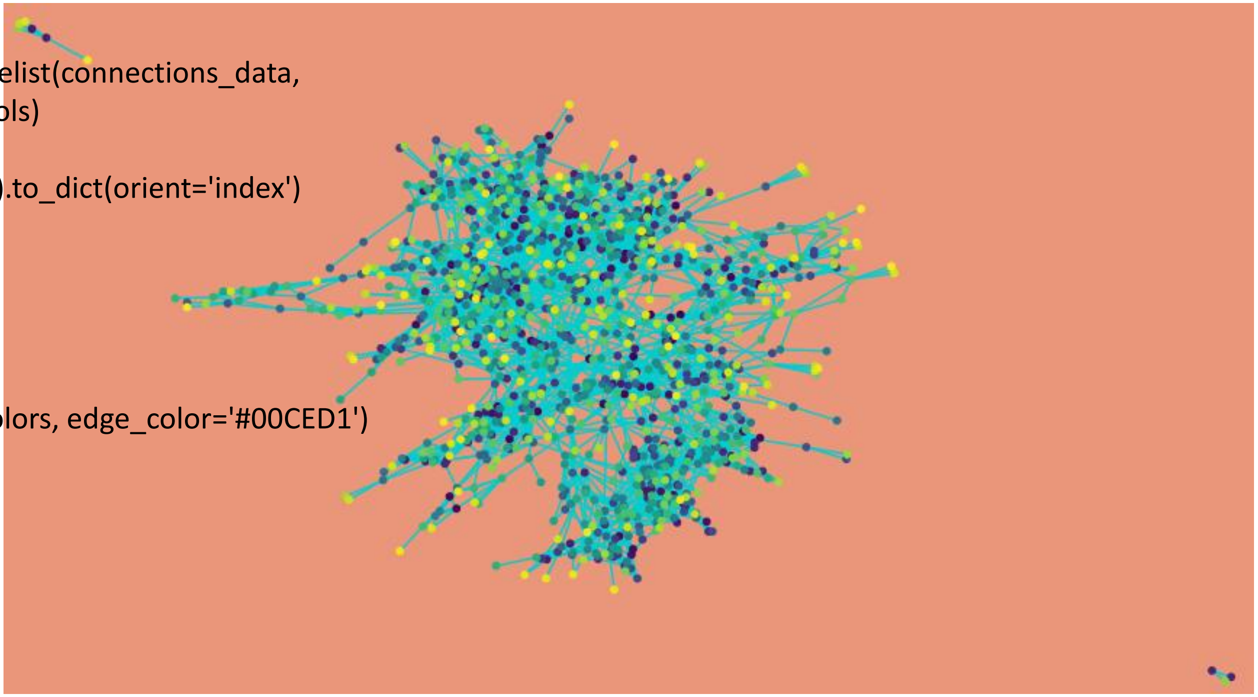
Graficando una red desde datos de un archivo utilizando Pandas

```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx

xls = pd.ExcelFile('15.Social Network Dataset.xlsx')

print(xls.sheet_names) # emito el nombre de las hojas del archivo

network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])

elements_data = network_data['Elements'] # lista de nodos

connections_data = network_data['Connections'] # lista de bordes

edge_cols = ['Type', 'Weight', 'When']

graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
source='From', target='To',edge_attr=edge_cols)

node_dict = elements_data.set_index('Label').to_dict(orient='index')

nx.set_node_attributes(graph, node_dict)

nx.draw(graph, node_size=5, node_color='#7FFF00')
plt.show()
```



Graficando una red desde datos de un archivo utilizando numpy y pandas

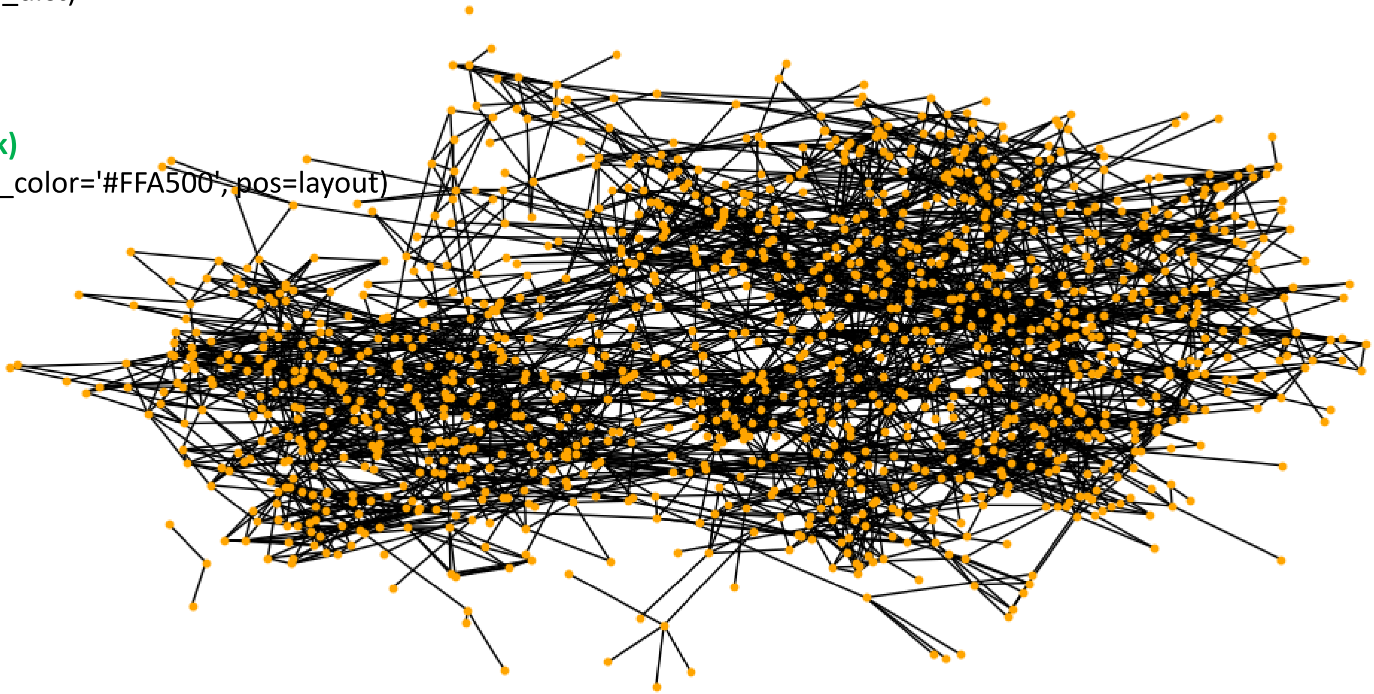
```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np

xls = pd.ExcelFile('15.Social Network Dataset.xlsx')

network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])

elements_data = network_data['Elements']

connections_data = network_data['Connections']

edge_cols = ['Type', 'Weight', 'When']

graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
source='From', target='To',edge_attr=edge_cols)

node_dict = elements_data.set_index('Label').to_dict(orient='index')

nx.set_node_attributes(graph, node_dict)

fig = plt.figure(figsize=(10,5))
colors = np.linspace(0,1,len(graph.nodes))
nx.draw(graph, node_size=20, node_color=colors, edge_color='#00CED1')
fig.set_facecolor('salmon')
plt.show()
```



Graficando una red desde datos de un archivo utilizando numpy y pandas

spring_layout: ubica los nodos utilizando el algoritmo dirigido por la fuerza de Fruchterman-Reingold

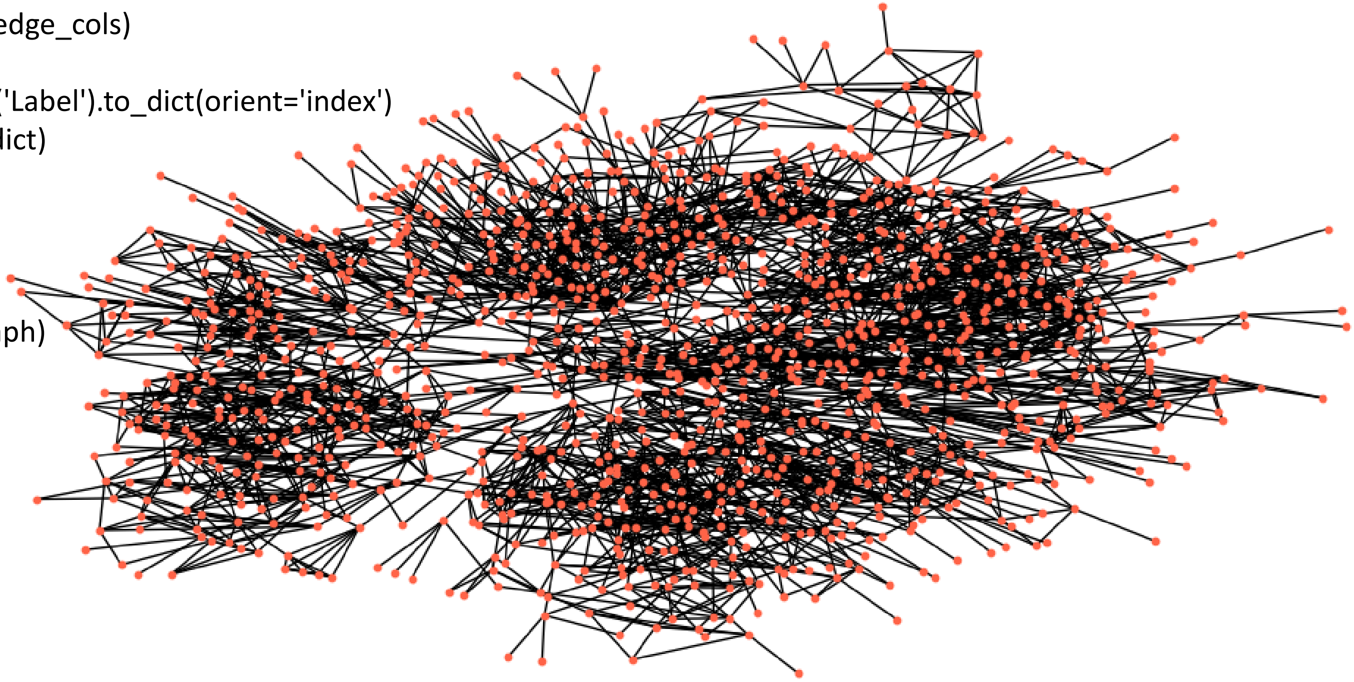
```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np

xls = pd.ExcelFile('15.Social Network Dataset.xlsx')
network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])
elements_data = network_data['Elements']
connections_data = network_data['Connections']
edge_cols = ['Type', 'Weight', 'When']
graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
source='From', target='To',edge_attr=edge_cols)
node_dict = elements_data.set_index('Label').to_dict(orient='index')
nx.set_node_attributes(graph, node_dict)

fig = plt.figure(figsize=(10,5))
k = 0.1
layout = nx.spring_layout(graph,k=k)
nx.draw(graph, node_size=20, node_color='#FFA500', pos=layout)
plt.show()
```



Graficando una red desde datos de un archivo utilizando numpy y pandas

kamada_kawai_layout: ubica los nodos utilizando la función de costo de longitud de ruta Kamada-Kawai

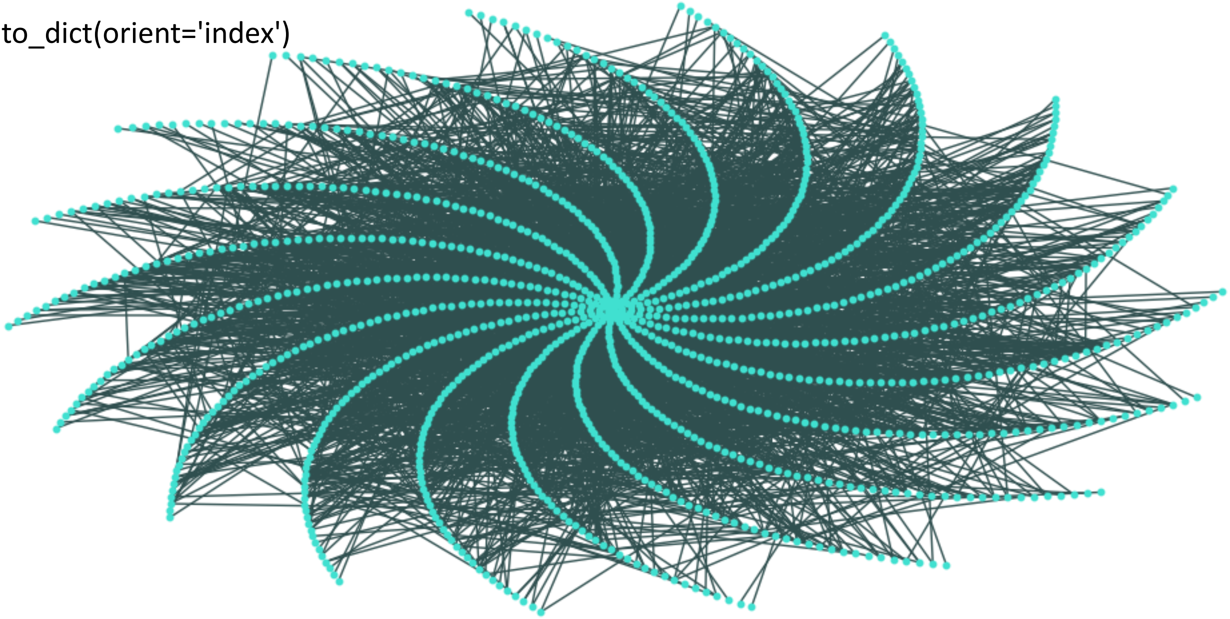
```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np

xls = pd.ExcelFile(‘15.Social Network Dataset.xlsx')

network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])
elements_data = network_data['Elements']
connections_data = network_data['Connections']
edge_cols = ['Type', 'Weight', 'When']
graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
                                    source='From',
                                    target='To',
                                    edge_attr=edge_cols)

node_dict = elements_data.set_index('Label').to_dict(orient='index')
nx.set_node_attributes(graph, node_dict)



plt.figure(figsize=(10, 5))
layout = nx.kamada_kawai_layout(graph)

nx.draw(graph,
      node_size=10,
      node_color='#FF6347',
      pos=layout)
plt.show()
```



Graficando una red desde datos de un archivo utilizando numpy y pandas

spiral_layout: ubica los nodos en un diseño en espiral.

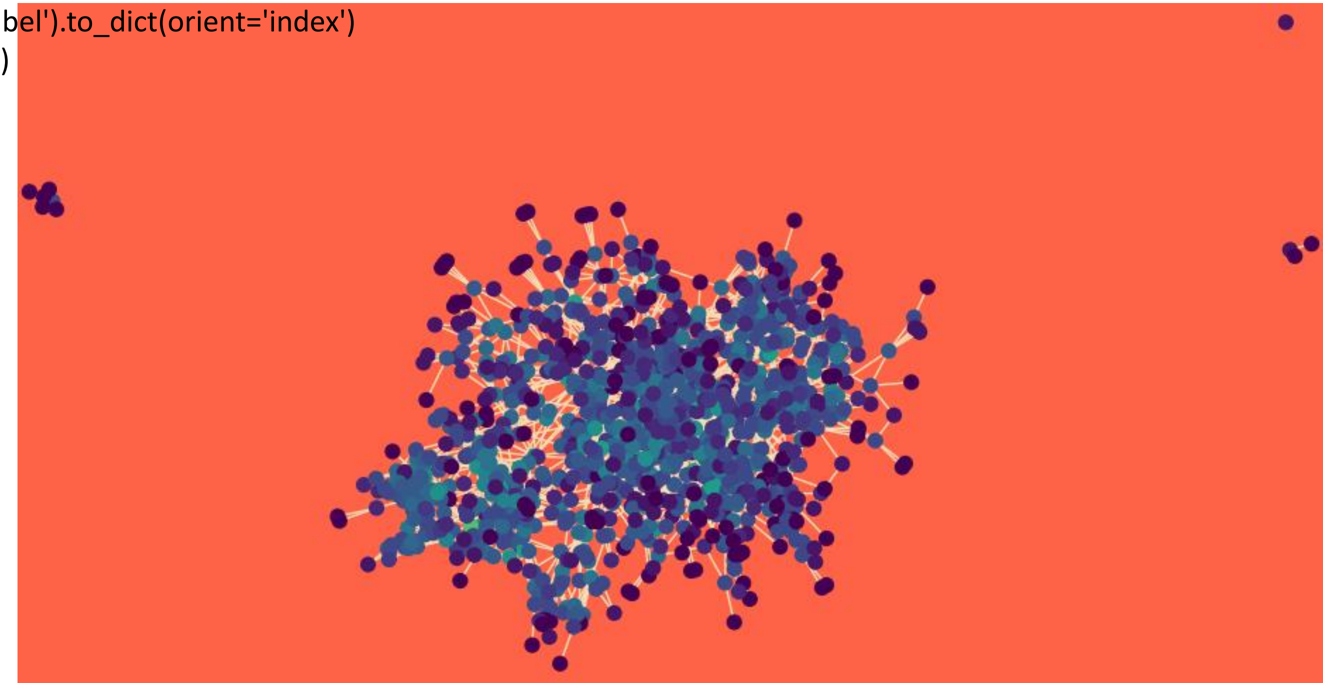
```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np

xls = pd.ExcelFile(‘15.Social Network Dataset.xlsx')

network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])
elements_data = network_data['Elements']
connections_data = network_data['Connections']
edge_cols = ['Type', 'Weight', 'When']
graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
                                    source='From',
                                    target='To',
                                    edge_attr=edge_cols)

node_dict = elements_data.set_index('Label').to_dict(orient='index')
nx.set_node_attributes(graph, node_dict)

plt.figure(figsize=(10, 5))
layout = nx.spiral_layout(graph)

nx.draw(graph,
      node_size=10,
      node_color='#40E0D0',
      edge_color='#2F4F4F',
      pos=layout)
plt.show()
```



Resumir gráficos mediante métricas de centralidad

degree_centrality calcula la centralidad de grados para los nodos en una red bipartita.

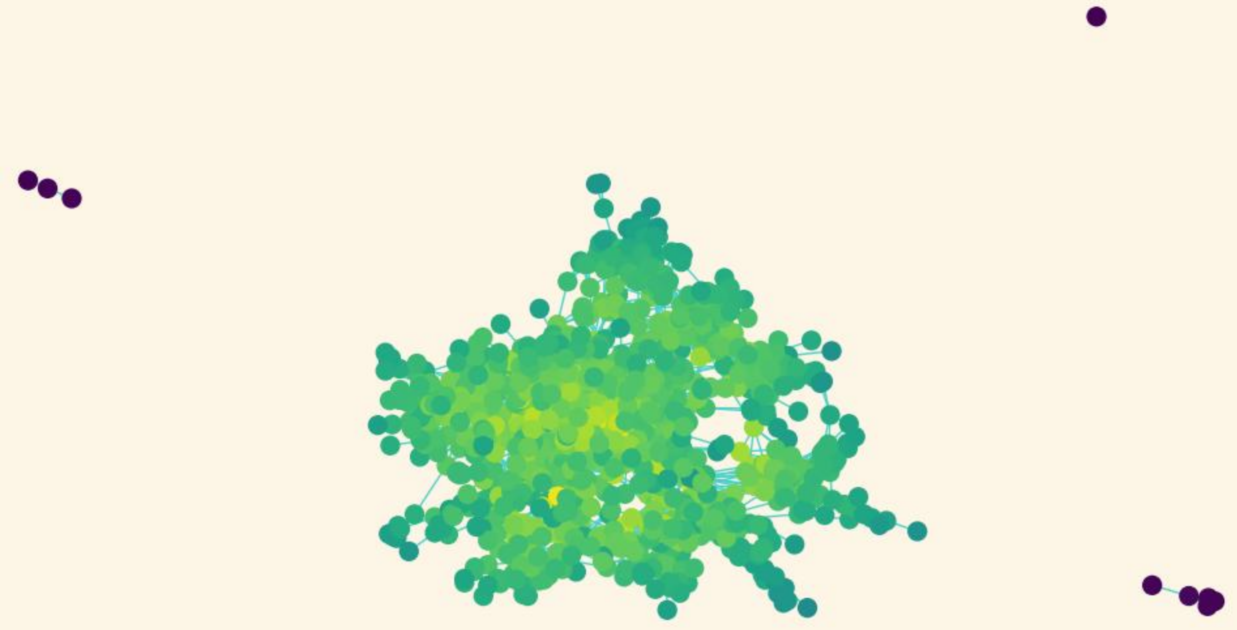
```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
from random import sample

xls = pd.ExcelFile(‘15.Social Network Dataset.xlsx')

network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])
elements_data = network_data['Elements'] # node list
connections_data = network_data['Connections'] # edge list
edge_cols = ['Type', 'Weight', 'When']
graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
                                source='From',
                                target='To',
                                edge_attr=edge_cols)

node_dict = elements_data.set_index('Label').to_dict(orient='index')
nx.set_node_attributes(graph, node_dict)

fig = plt.figure(figsize=(10, 5))

centrality = nx.degree_centrality(graph)
colors = list(centrality.values())

nx.draw(graph,
      node_size=50,
      node_color=colors,
      edge_color='#F5DEB3')
fig.set_facecolor('#FF6347')
plt.show()
```



Resumir gráficos mediante métricas de centralidad

closeness_centrality calcula la centralidad de proximidad para los nodos en una red bipartita.

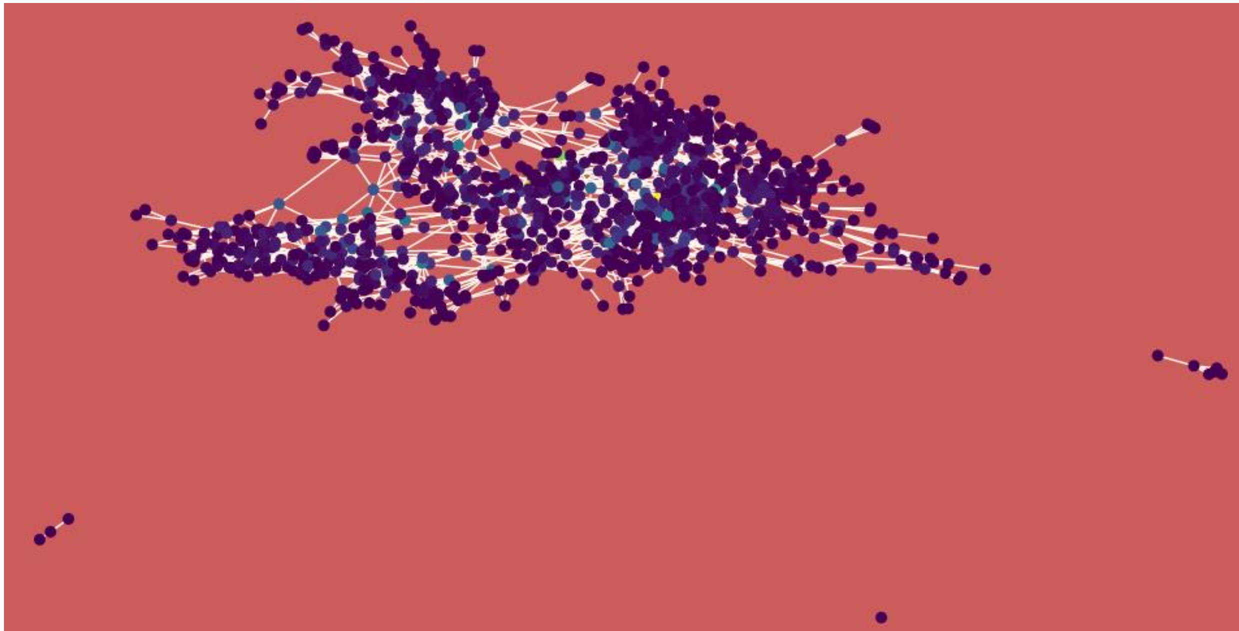
```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
from random import sample

xls = pd.ExcelFile(‘15.Social Network Dataset.xlsx')

network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])
elements_data = network_data['Elements'] # node list
connections_data = network_data['Connections'] # edge list
edge_cols = ['Type', 'Weight', 'When']
graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
                                   source='From',
                                   target='To',
                                   edge_attr=edge_cols)

node_dict = elements_data.set_index('Label').to_dict(orient='index')
nx.set_node_attributes(graph, node_dict)

fig = plt.figure(figsize=(10, 5))

centrality = nx.closeness_centrality(graph)
colors = list(centrality.values())

nx.draw(graph,
      node_size=100,
      node_color=colors,
      edge_color='#48D1CC')
fig.set_facecolor('#FDF5E6')
plt.show()
```



Resumir gráficos mediante métricas de centralidad

betweenness_centrality calcula la centralidad de intermediación para los nodos en una red bipartita.

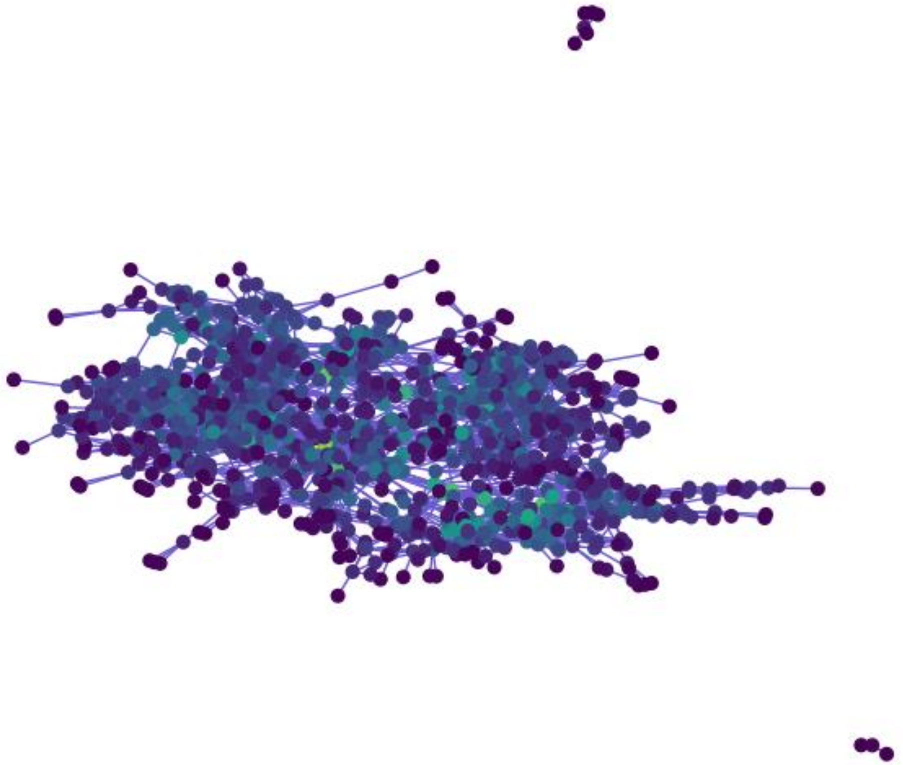
```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
from random import sample

xls = pd.ExcelFile(‘15.Social Network Dataset.xlsx')

network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])
elements_data = network_data['Elements'] # node list
connections_data = network_data['Connections'] # edge list
edge_cols = ['Type', 'Weight', 'When']
graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
                                    source='From',
                                    target='To',
                                    edge_attr=edge_cols)

node_dict = elements_data.set_index('Label').to_dict(orient='index')
nx.set_node_attributes(graph, node_dict)

sample(centrality.items(), 10)

fig = plt.figure(figsize=(10, 5))

centrality = nx.betweenness_centrality(graph)
colors = list(centrality.values())

nx.draw(graph,
      node_size=30,
      node_color=colors,
      edge_color='#FFFAF0')
fig.set_facecolor('#CD5C5C')
plt.show()
```



Resumir gráficos mediante métricas de centralidad

katz_centrality calcula la centralidad de Katz para los nodos del gráfico.

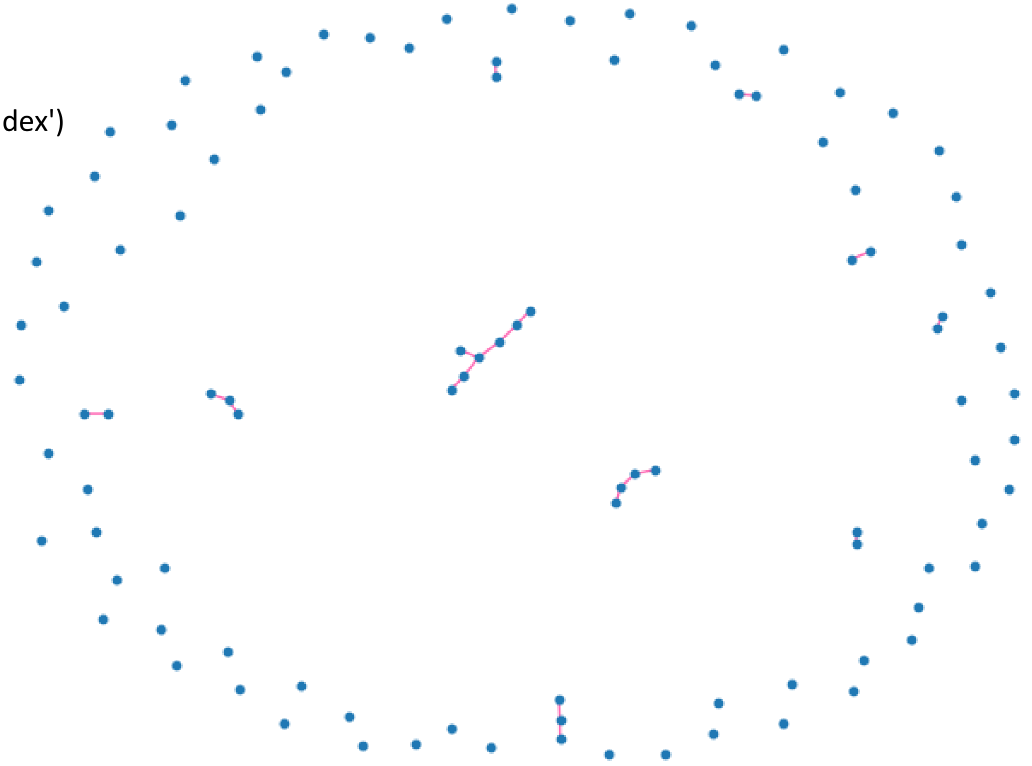
```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
from random import sample

xls = pd.ExcelFile(‘15.Social Network Dataset.xlsx')

network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])
elements_data = network_data['Elements'] # node list
connections_data = network_data['Connections'] # edge list
edge_cols = ['Type', 'Weight', 'When']
graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
                                   source='From',
                                   target='To',
                                   edge_attr=edge_cols)

node_dict = elements_data.set_index('Label').to_dict(orient='index')
nx.set_node_attributes(graph, node_dict)

sample(centrality.items(), 10)

fig = plt.figure(figsize=(10, 5))

centrality = nx.katz_centrality(graph)
colors = list(centrality.values())

nx.draw(graph,
      node_size=30,
      node_color=colors,
      edge_color='#6A5ACD')
fig.set_facecolor('#FFFFFF')
plt.show()
```



Crear y visualizar sub-gráficos

```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
from random import sample

xls = pd.ExcelFile('Social_Network_Dataset.xlsx')

network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])
elements_data = network_data['Elements'] # node list
connections_data = network_data['Connections'] # edge list
edge_cols = ['Type', 'Weight', 'When']
graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
                                        source='From',
                                        target='To',
                                        edge_attr=edge_cols)

node_dict = elements_data.set_index('Label').to_dict(orient='index')
nx.set_node_attributes(graph, node_dict)

fig = plt.figure(figsize=(10,5))
nx.draw(graph,
      node_size=10,
      edge_color='#FF69B4')
fig.set_facecolor('#FFFFFF')
plt.show()

# luego ejecutar
```

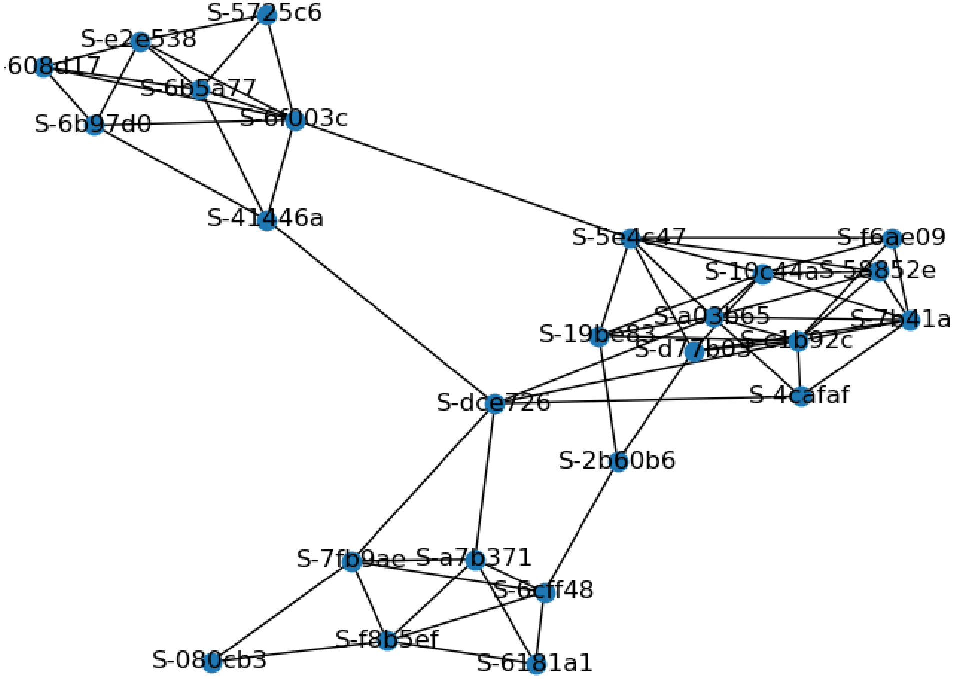
```
len(graph.nodes)
len(graph.edges)

node = sample(graph.nodes, 1)[0]
graph.nodes[node]

sampled_nodes = sample(graph.nodes, 100)
subgraph = graph.subgraph(sampled_nodes)
nx.draw(subgraph,
      node_size=10,
      with_labels=False,
      edge_color='#FF69B4')
plt.show()
```



Crear y visualizar sub-gráficos

```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
from random import sample
xls = pd.ExcelFile(‘15.Social Network Dataset.xlsx')
network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])
elements_data = network_data['Elements'] # node list
connections_data = network_data['Connections'] # edge list
edge_cols = ['Type', 'Weight', 'When']
graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
                                    source='From',
                                    target='To',
                                    edge_attr=edge_cols)
node_dict = elements_data.set_index('Label').to_dict(orient='index')
nx.set_node_attributes(graph, node_dict)
fig = plt.figure(figsize=(10,5))
nx.draw(graph,
      node_size=10,
      edge_color='#FF69B4')
fig.set_facecolor('#FFFFFF')
plt.show()
len(graph.nodes)
len(graph.edges)
node = sample(graph.nodes, 1)[0]
graph.nodes[node]
sampled_nodes = sample(graph.nodes, 100)
subgraph = graph.subgraph(sampled_nodes)
nx.draw(subgraph,
      node_size=10,
      with_labels=False,
      edge_color='#FF69B4')
plt.show()
```

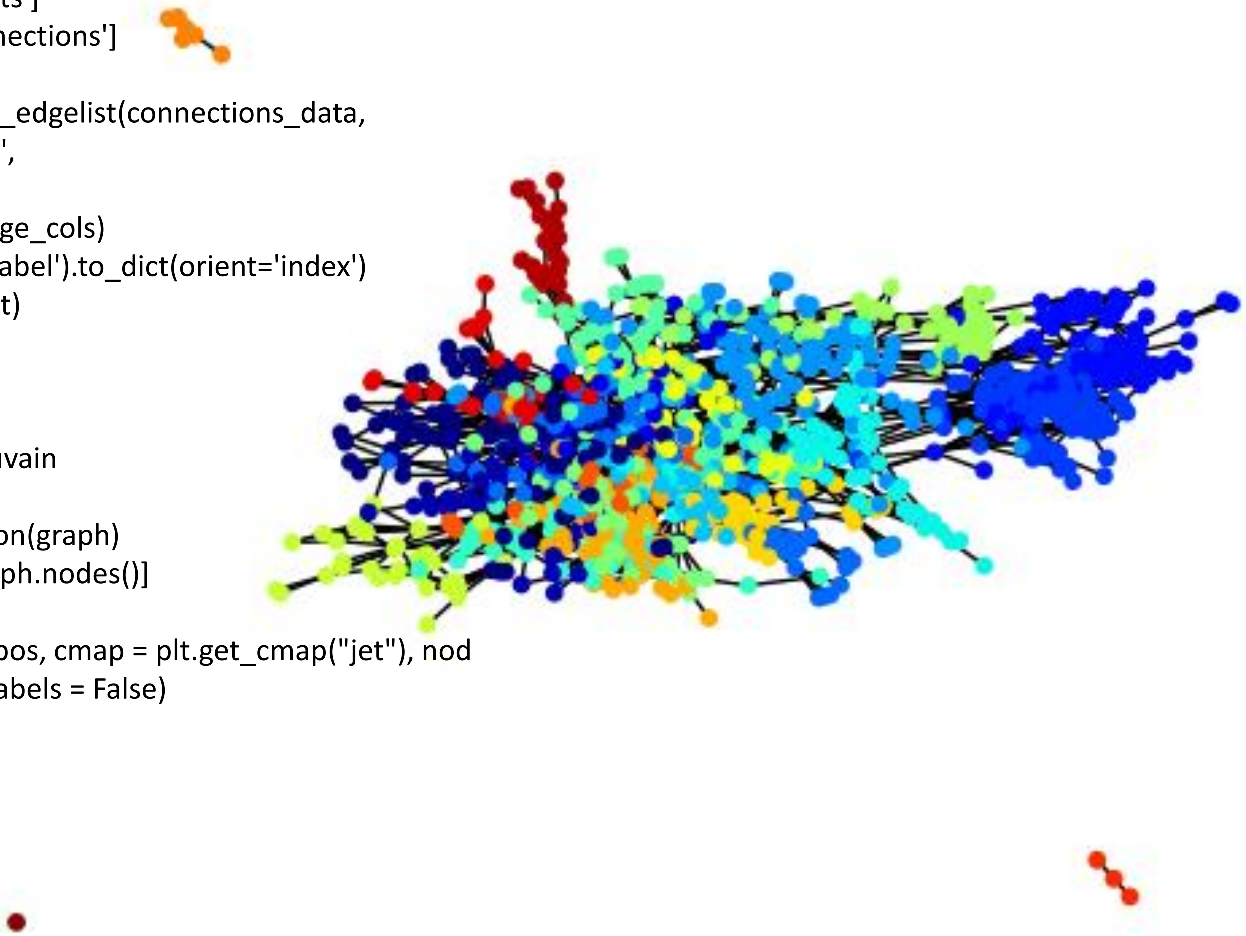
```
from collections import defaultdict
nodes_school_id = nx.get_node_attributes(graph,
                                'School (ID)')
school_nodes = defaultdict(list)
for node, school_id in nodes_school_id.items():
   school_nodes[school_id].append(node)
school_nodes[5]
graph.nodes['S-087f53']
subgraphs = {}

for school_id, nodes in school_nodes.items():
   subgraph = graph.subgraph(nodes)
   subgraphs[school_id] = subgraph
subgraphs[5].nodes
nx.draw(subgraphs[3],
      node_size=80,
      with_labels=True)
plt.show()
```



Detección de comunidades en networkx

Es importante que tenga networkx actualizado:
pip install --upgrade networkx
y además debe instalar:
pip install **community**
pip install **python-louvain**

```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
from random import sample

xls = pd.ExcelFile('Social_Network_Dataset.xlsx')
network_data = pd.read_excel(xls, sheet_name=['Elements', 'Connections'])
elements_data = network_data['Elements']
connections_data = network_data['Connections']
edge_cols = ['Type', 'Weight', 'When']
graph = nx.convert_matrix.from_pandas_edgelist(connections_data,
                                 source='From',
                                 target='To',
                                 edge_attr=edge_cols)
node_dict = elements_data.set_index('Label').to_dict(orient='index')
nx.set_node_attributes(graph, node_dict)

fig = plt.figure(figsize=(15, 10))

from community import community_louvain
spring_pos = nx.spring_layout(graph)
parts = community_louvain.best_partition(graph)
values = [parts.get(node) for node in graph.nodes()]
plt.axis("off")
nx.draw_networkx(graph, pos = spring_pos, cmap = plt.get_cmap("jet"), nod
e_color = values, node_size = 15, with_labels = False)
plt.show()
```



https://networkx.github.io/documentation/latest/tutorial.html

https://networkx.github.io/documentation/stable/reference/generators.html

https://networkx.github.io/documentation/stable/auto_examples/drawing/plot_labels_and_colors.html

https://networkx.github.io/documentation/stable/reference/generated/networkx.drawing.nx_pylab.draw_networkx.html

FIN