PYTHON
GRAFOS
CON NETWORKX

NetworkX es un paquete de Python para la creación, manipulación y estudio de la estructura, dinámica y funciones de redes complejas. Requiere Python 3.5, 3.6, 3.7 o 3.8. Lo primero que tenemos que hacer es instalar la librería. Entonces en la consola del VSC, escribimos:

pip install networkx

Si esta nstalada pueden actualizarla así:

pip install --upgrade networkx

```
PS C:\Users\DELL INSPIRON 13\Documents\PythonScripts> pip install networkx
Requirement already satisfied: networkx in c:\users\dell inspiron 13\appdata\local\programs\python\python38\lib\site-packages (2.4)
Requirement already satisfied: decorator>=4.3.0 in c:\users\dell inspiron 13\appdata\local\programs\python\python38\lib\site-packages (from network
x) (4.4.2)
```

```
import networkx as nx

G = nx.Graph()
```

Graph crea un grafo vacío sin nodos ni aristas (un "grafo nulo") . G es el identificador para este ejemplo.

Esta clase (Graph) implementa un grafo no dirigido.

Si ejecutamos este ejemplo no obtendremos resultados porque el grafo está vacío, no dibujamos nodos ni enlaces y además no importamos el módulo para visualizar.

```
# ejemplo 1

import networkx as nx

G = nx.Graph()
G.add_node('Hola Mundo!!')
```

Con add.node agregamos un nodo y le pasamos como parámetro el valor. Si ejecutamos este ejemplo tampoco obtendremos resultados porque el grafo no está dibujado y además no importamos el módulo para visualizar.

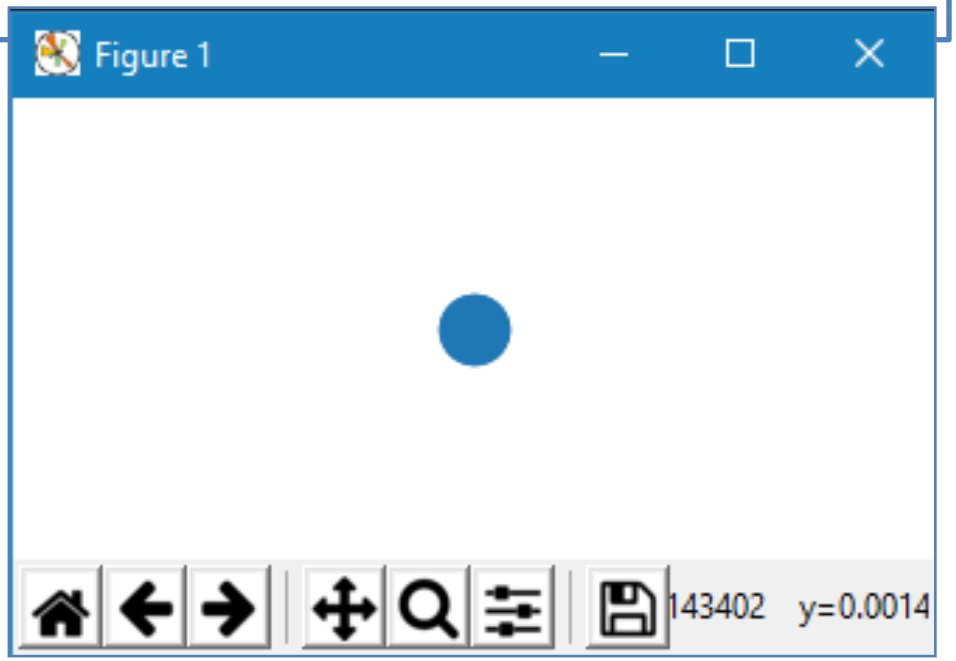
```
# ejemplo 2

import networkx as nx

G = nx.Graph()
G.add_node('Hola Mundo!!')
nx.draw(G)
```

Con nx.draw dibujamos nuestro nodo.

Si ejecutamos este ejemplo tampoco obtendremos resultados , aún no importamos el módulo para visualizar. Se pueden generar diferentes tipos de grafos, draw() es el normal, también está el draw_circular, draw_espectral(), draw_shell(), etc. Pueden hallar más en la documentación: https://networkx.github.io/documentation/stable/reference/drawing.html?highlight=draw

```
 # ejemplo 3
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Hola Mundo!!')
nx.draw(G)
plt.show()
```

Importamos matplotlib.pyplot y con plt.show() visualizamos nuestro primer grafo con un nodo:



Atributos de los nodos

```
# ejemplo 4
import networkx as nx

import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Hola Mundo!!')
nx.draw(G, with_labels=True)
plt.show()
```

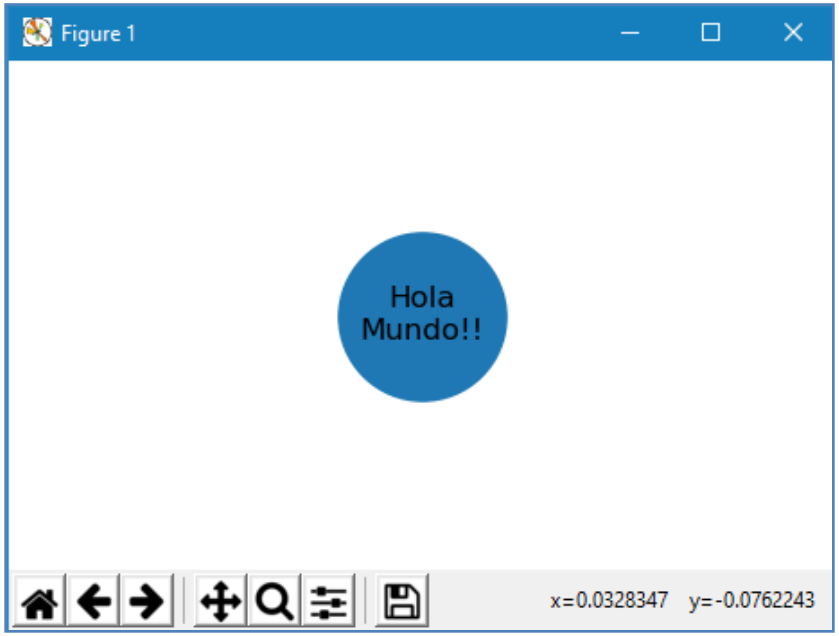
Con with_labels=True visualizamos las etiquetas del nodo.

```
# ejemplo 5
import networkx as nx

import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Hola Mundo!!')
nx.draw(G, with_labels=True, node_size=4500)
plt.show()
```

Con node_size modificamos el tamaño del nodo.

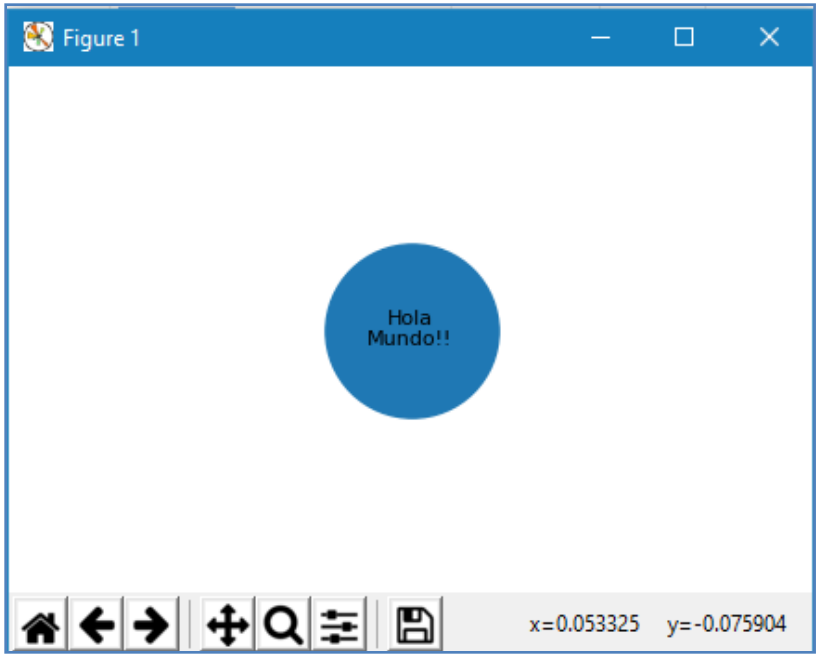
```
# ejemplo 6
import networkx as nx

import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Hola\nMundo!!')
nx.draw(G, with_labels=True, node_size=4500)
plt.show()
```

Podemos utilizar carácter de escape en las cadenas del nodo.



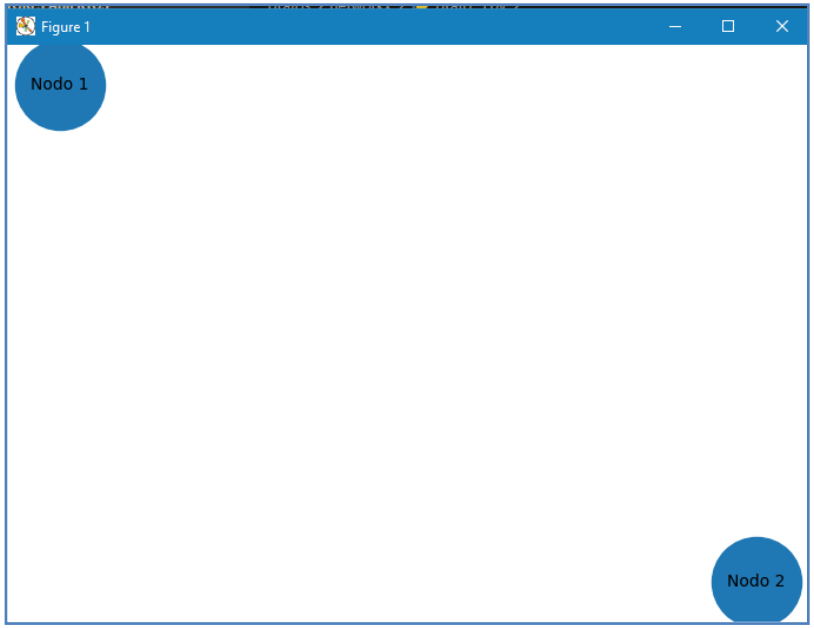
```
# ejemplo 7
import networkx as nx

import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Hola\nMundo!!')
nx.draw(G, with_labels=True, node_size=4500, font_size=8)
plt.show()
```

Podemos cambiar el tamaño de fuente del nodo.



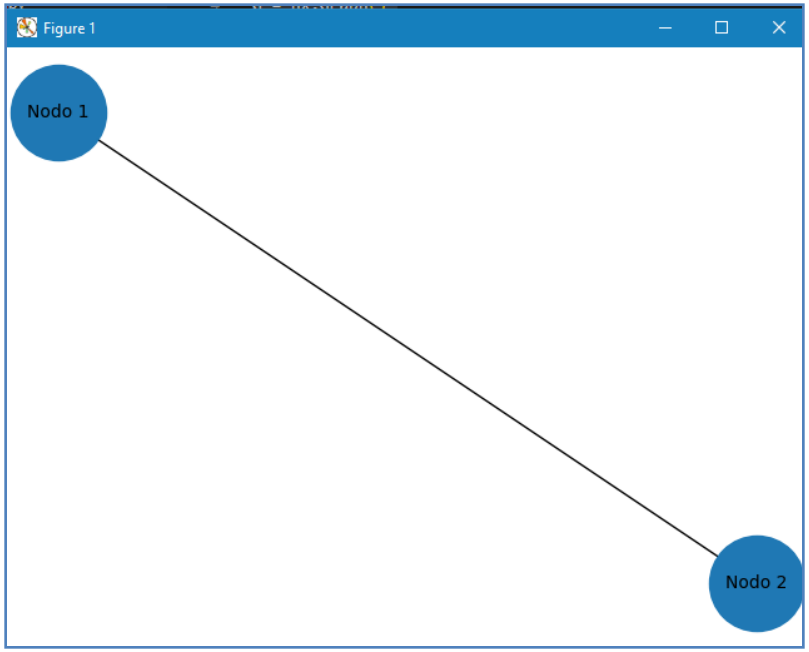
```
# ejemplo 8
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Nodo 1')
G.add_node('Nodo 2')
nx.draw(G, with_labels=True, node_size=3000, font_size=10)
plt.show()
```

Agregamos dos nodos y visualizamos:



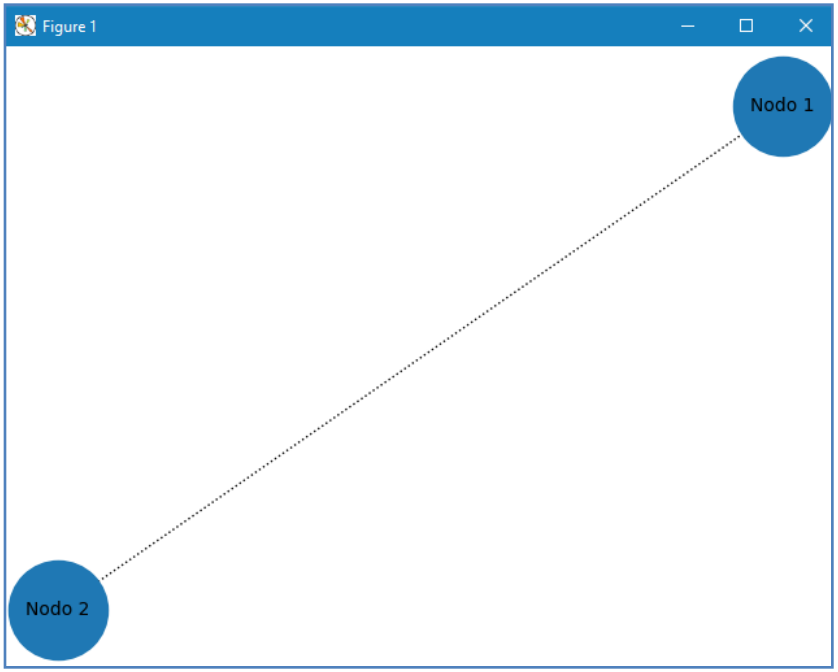
```
# ejemplo 9
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Nodo 1')
G.add_node('Nodo 2')
G.add_edge('Nodo 1','Nodo 2')
nx.draw(G, with_labels=True, node_size=3000, font_size=10)
plt.show()
```

Enlazamos los dos nodos mediante add.edge y visualizamos:



```
# ejemplo 10
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Nodo 1')
G.add_node('Nodo 2')
G.add_edge('Nodo 1','Nodo 2')
nx.draw(G, with_labels=True, node_size=3000, font_size=10, style='dotted')
plt.show()
```

Cambiamos el estilo de la línea de unión y visualizamos:



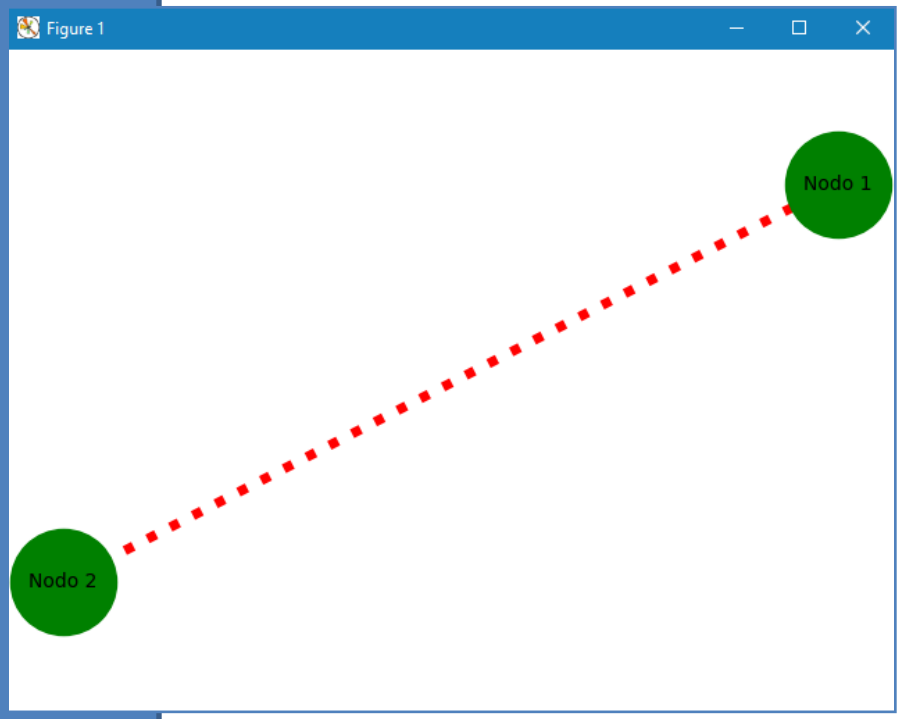
```
# ejemplo 11
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Nodo 1', weight=5.0, labels = 'Hola')
G.add_node('Nodo 2', weight=4.0)
G.add_node('Nodo 3', weight=3.0, labels = 'cómo estás?')
G.nodes['Nodo 3']
list(G.nodes(data=True))
list(G.nodes.data())
```

```
>>> G.nodes['Nodo 3']
{'weight': 3.0, 'labels': 'cómo estás?'}
>>> list(G.nodes(data=True))
[('Nodo 1', {'weight': 5.0, 'labels': 'Hola'}), ('Nodo 2', {'weight': 4.0}), ('Nodo 3', {'weight': 3.0, 'labels': 'cómo estás?'})]
>>> list(G.nodes.data())
[('Nodo 1', {'weight': 5.0, 'labels': 'Hola'}), ('Nodo 2', {'weight': 4.0}), ('Nodo 3', {'weight': 3.0, 'labels': 'cómo estás?'})]
>>>
```

Los atributos de los nodos se devuelven como un diccionarios. Y con **list(G.nodes(data=True)) o list(G.nodes.data())** devuelve los atributos de los nodos como listas de tuplas compuestas del valor del nodo y un diccionario de sus atributos.

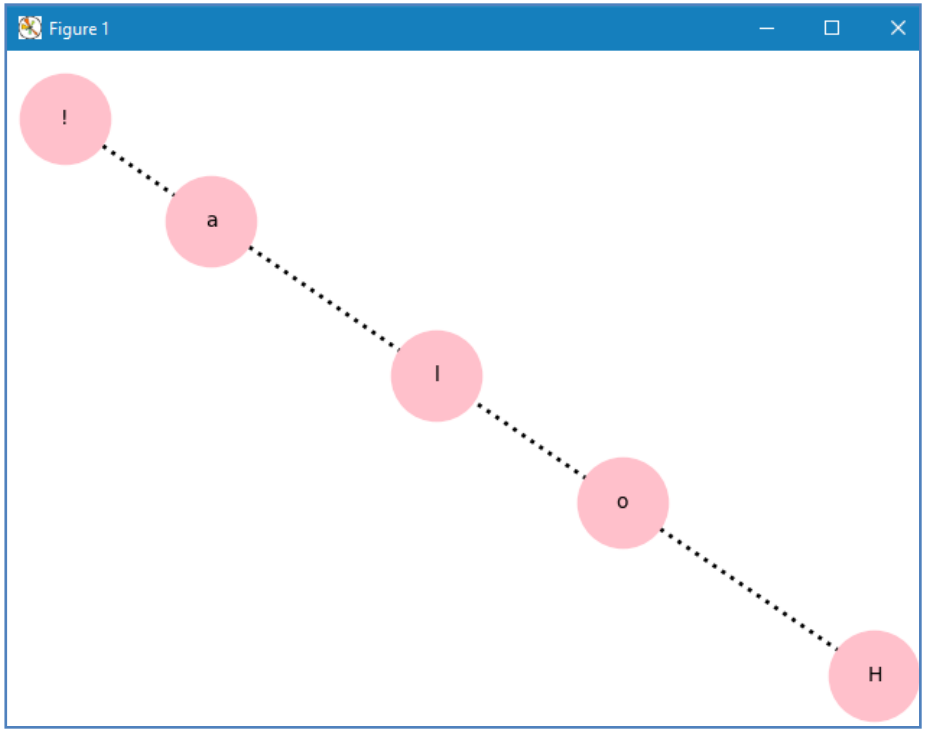
Atributos de los enlaces

```
# ejemplo 12
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Nodo 1')
G.add_node('Nodo 2')
G.add_edge('Nodo 1','Nodo 2')
nx.draw(G,
        node_color="green",
        edge_color="red",
        width=5,
        font_size=10,
        with_labels=True,
        node_size=3000,
        style='dotted'
        )

plt.show()
```



Cambiamos el color del nodo, color de la arista, el ancho y visualizamos.

```
# ejemplo 13
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_nodes_from("Hola!")
G.add_edge('H','o‘,  weight=1.0)
G.add_edge('o','l‘,   weight=3.0)
G.add_edge('l','a‘,   weight=2.0)
G.add_edge('a','!‘,   weight=4.0)
print("Número de nodos: ", G.number_of_nodes(
))
print("Nodos: ", G.nodes())
print("Número de enlaces: ", G.number_of_edges
())
print("Enlaces: ", G.edges())

nx.draw(G,
        node_color="pink",
        edge_color="black",
        font_size=10,
        width=2,
        with_labels=True,
        node_size=2000,
        style= 'dotted'
        )
plt.show()
```



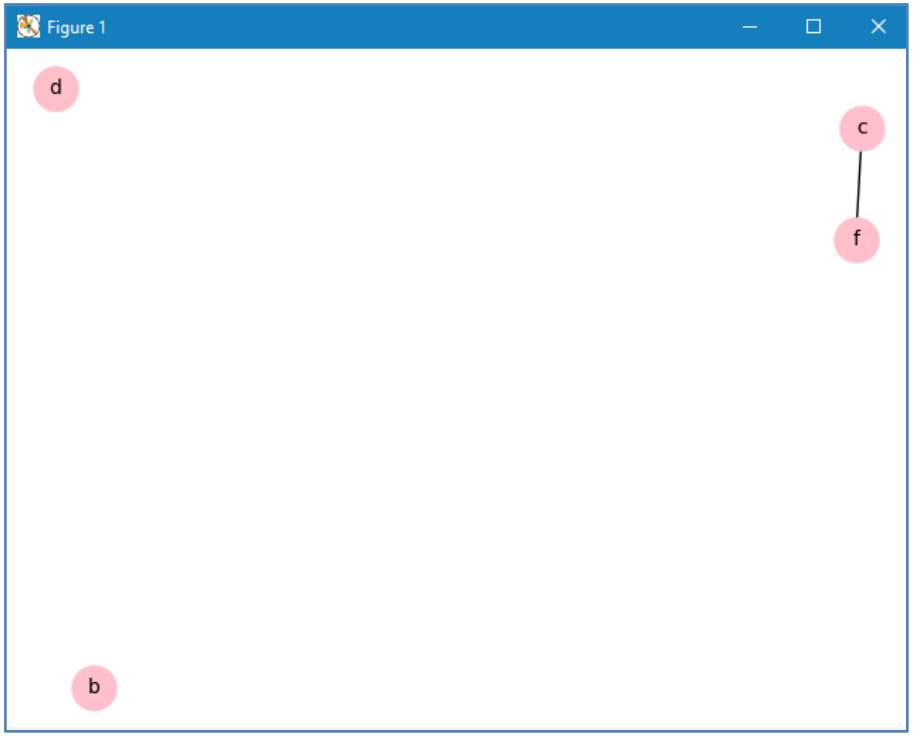
Con **weight** agregamos peso a los enlaces.

```
# ejemplo 14
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Nodo 1')
G.add_node('Nodo 2')
G.add_node('Nodo 3')
G.add_edge('Nodo 1', 'Nodo 2', weight=5.0)
G.add_edge('Nodo 2', 'Nodo 3', weight=3.5, capacity=15, length=342.7)
G.add_edge('Nodo 3', 'Nodo 1', weight=1.5)

list(G.edges(data=True))

list(G.edges.data())
```

```
>>> list(G.edges(data=True))
[('Nodo 1', 'Nodo 2', {'weight': 5.0}), ('Nodo 1', 'Nodo 3', {'weight': 1.5}), ('Nodo 2', 'Nodo 3', {'weight': 3.5, 'capacity': 15, 'l
ength': 342.7})]
>>> list(G.edges.data())
[('Nodo 1', 'Nodo 2', {'weight': 5.0}), ('Nodo 1', 'Nodo 3', {'weight': 1.5}), ('Nodo 2', 'Nodo 3', {'weight': 3.5, 'capacity': 15, 'l
ength': 342.7})]
>>>
```

Los atributos de un enlace también se devuelven como **listas de tuplas** compuestas de pares de nodos y un diccionario de sus atributos.

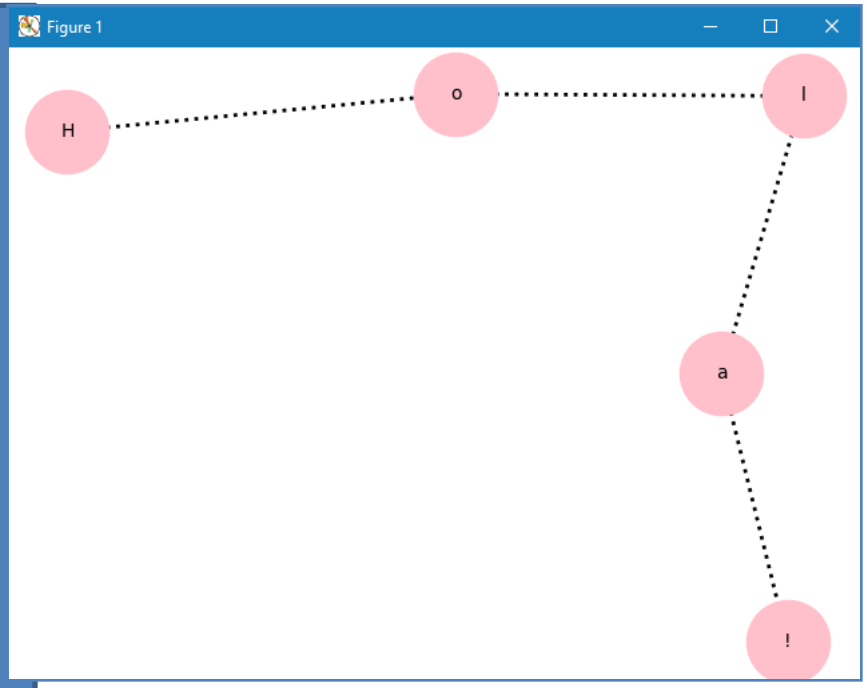
```
# ejemplo 15
import networkx as nx
import matplotlib.pyplot as plt

L1 = ['a','b','c','d','e','f'] #vertices
L2 = [('a','b'), ('a','f'), ('b','e'),('c','e'), ('c','d'),('e','f')
,('f','c')]
G = nx.Graph()
G.add_nodes_from(L1)
G.add_edges_from(L2)

G.remove_node('a')
G.remove_nodes_from('e')
list(G.nodes)
G.remove_edge('c','d')
list(G.edges)

nx.draw(G,
        node_color="pink",
        edge_color="black",
        font_size=10,
        width=1,
        with_labels=True,
        node_size=500,
        )
plt.show()
```




```
>>> G.remove_node('a')
>>> G.remove_nodes_from('e')
>>> list(G.nodes)
['b', 'c', 'd', 'f']
>>> G.remove_edge('c','d')
>>> list(G.edges)
[('c', 'f')]
```

Consultar el grafo

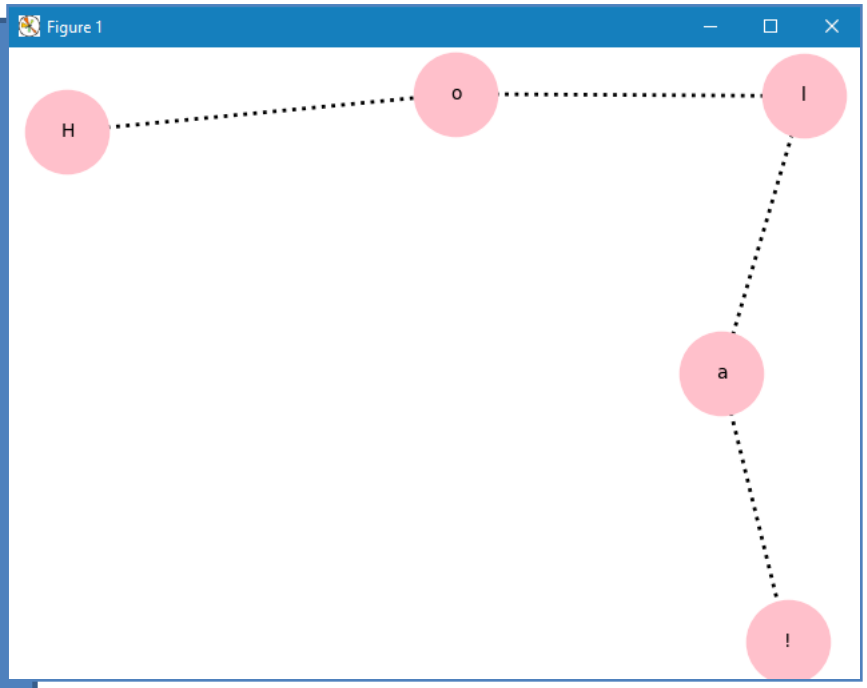
```
# ejemplo 16
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_nodes_from("Hola!")
G.add_edge('H','o')
G.add_edge('o','l')
G.add_edge('l','a')
G.add_edge('a','!')
print("Número de nodos: ", G.number_of_nodes())
print("Nodos: ", G.nodes())
print("Número de enlaces: ", G.number_of_edges())
print("Enlaces: ", G.edges())

nx.draw(G,
        node_color="pink",
        edge_color="black",
        font_size=10,
        width=2,
        with_labels=True,
        node_size=2000,
        style= 'dotted'
        )

plt.show()
```




```
>>> print("Número de nodos: ", G.number_of_nodes())
Número de nodos:  5
>>> print("Nodos: ", G.nodes())
Nodos:  ['H', 'o', 'l', 'a', '!']
>>> print("Número de enlaces: ", G.number_of_edges())
Número de enlaces:  4
>>> print("Enlaces: ", G.edges())
Enlaces:  [('H', 'o'), ('o', 'l'), ('l', 'a'), ('a', '!')]
>>>
```

Podemos convertir una cadena en nodos **add_nodes_from** y también otros contenedores.

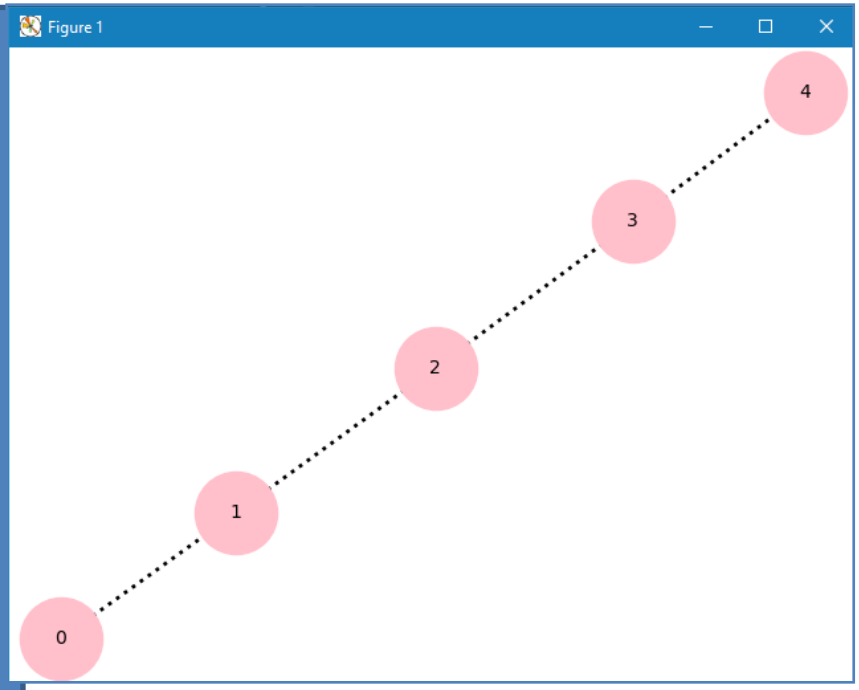
```
# ejemplo 17
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_nodes_from("Hola!")
G.add_edge('H','o')
G.add_edge('o','l')
G.add_edge('l','a')
G.add_edge('a','!')
print("Número de nodos: ", G.number_of_nodes())
print("Nodos: ", G.nodes())
print("Número de enlaces: ", G.number_of_edges())
print("Enlaces: ", G.edges())
print("Vecinos: ", list(G.neighbors('o')))

nx.draw(G,
        node_color="pink",
        edge_color="black",
        font_size=10,
        width=2,
        with_labels=True,
        node_size=2000,
        style= 'dotted'
        )
plt.show()
```




```
>>> print("Vecinos: ", list(G.neighbors('o')))
Vecinos:  ['H', 'l']
>>>
```

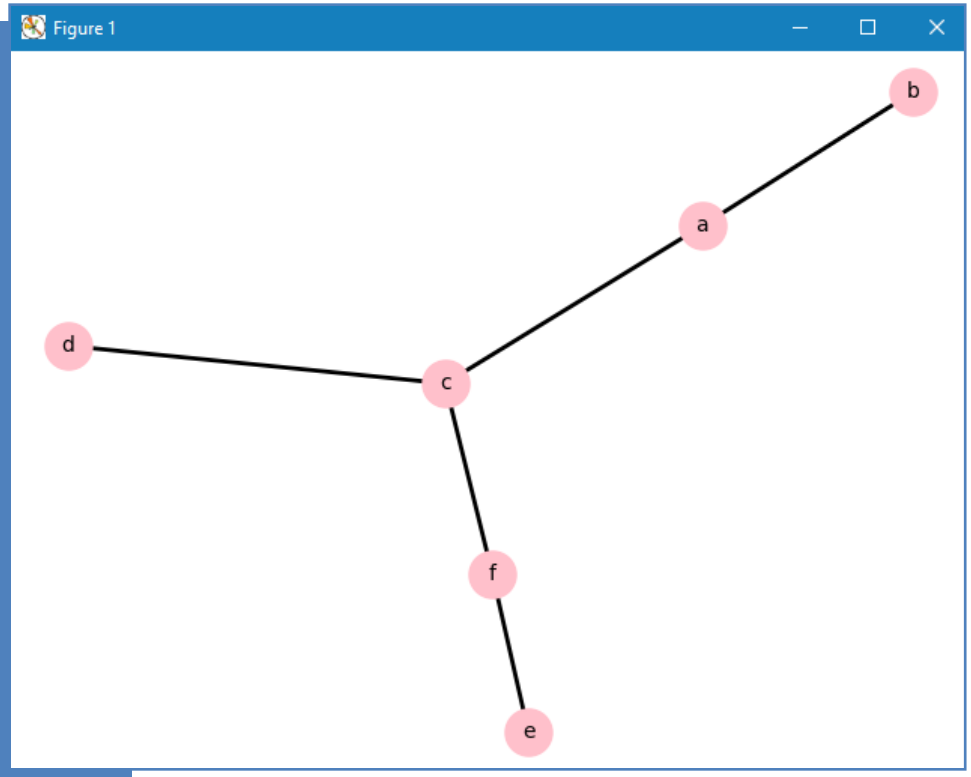
Podemos ver los vecinos con **neighbors().** Desde networkx 2.0 en adelante neighbors(elemento) devuelve un iterador en lugar de una lista. Para obtener la lista, simplemente hay que anteponer **list**

```
# ejemplo 18
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_nodes_from("Hola!")
G.add_edge('H','o')
G.add_edge('o','l')
G.add_edge('l','a')
G.add_edge('a','!')
print("Número de nodos: ", G.number_of_nodes())
print("Nodos: ", G.nodes())
print("Número de enlaces: ", G.number_of_edges())
print("Enlaces: ", G.edges())
G=nx.path_graph(5)
nx.draw(G,
        node_color="pink",
        edge_color="black",
        font_size=10,
        width=2,
        with_labels=True,
        node_size=2000,
        style= 'dotted'
        )

plt.show()
```



path_graph() devuelve el grafo de ruta n nodos conectados linealmente por n-1 aristas.
Las etiquetas de nodo son los enteros 0 a n - 1.

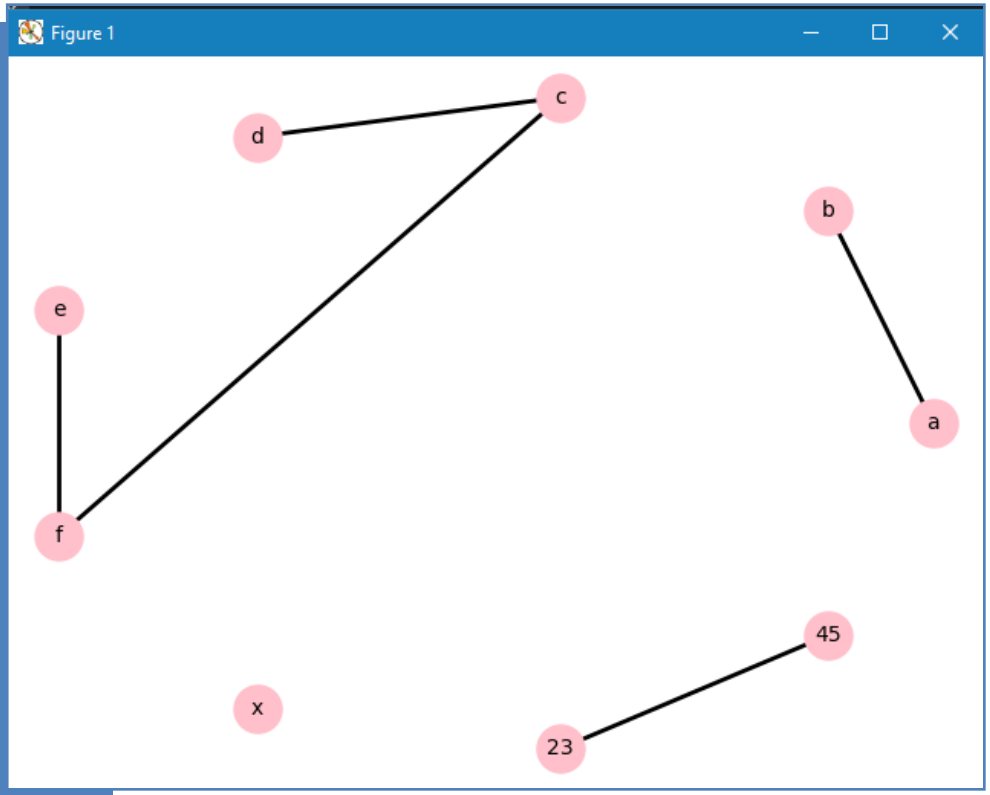
```
# ejemplo 19
import networkx as nx
import matplotlib.pyplot as plt


L1 = ['a', 'b', 'c', 'd', 'e', 'f']
L2 = [('a', 'b'), ('a', 'c'), ('c', 'd'), ('e', 'f'), ('f', 'c')]
G=nx.Graph()
G.add_nodes_from(L1)
G.add_edges_from(L2)
G.nodes()
G.edges()

nx.draw(G,
        node_color="pink",
        edge_color="black",
        font_size=10,
        width=2,
        with_labels=True,
        node_size=500,
        )

plt.show()
```




```
>>> G.nodes()
NodeView(('a', 'b', 'c', 'd', 'e', 'f'))
>>> G.edges()
EdgeView([('a', 'b'), ('a', 'c'), ('c', 'd'), ('c', 'f'), ('e', 'f')])
>>>
```

Los nodos pueden ser asignados desde listas con add_nodes_from -agrega múltiples nodos- y los enlaces pueden ser asignados por listas de tuplas con add_edges_from -agrega múltiples enlaces-

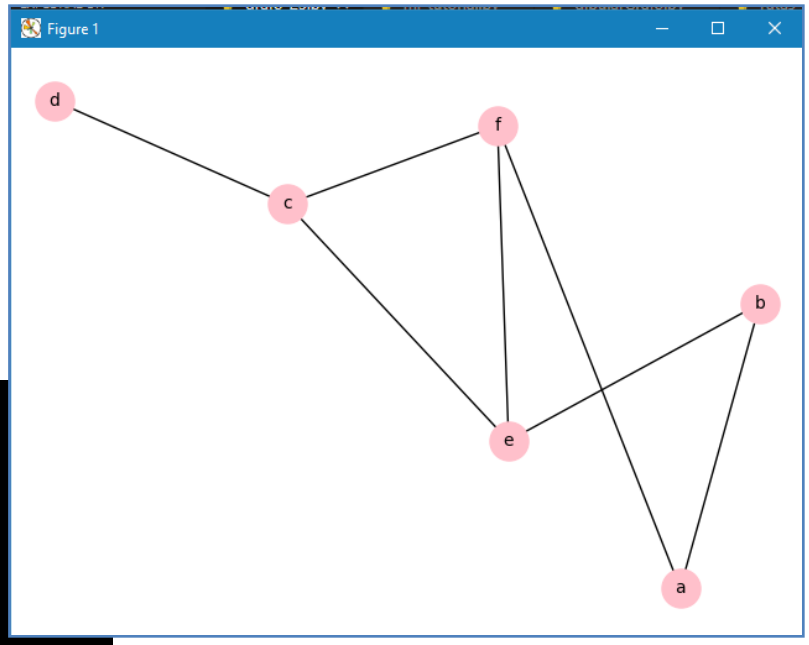
```
# ejemplo 20
import networkx as nx
import matplotlib.pyplot as plt
L1 = ['a','b','c','d','e','f']
L2 = [('a','b'),('c','d'),('e','f'),('f','c')]
G = nx.Graph()
G.add_nodes_from(L1)
G.add_edges_from(L2)
G.add_node('x')
G.add_edge(23,45)
G.nodes()
G.edges()
G.degree('a')
G.degree()
nx.draw_circular(G,
                 node_color="pink",
                 edge_color="black",
                 font_size=10,
                 width=2,
                 with_labels=True,
                 node_size=500,
        )
plt.show()
G.size()
G.order()
len(G)
```



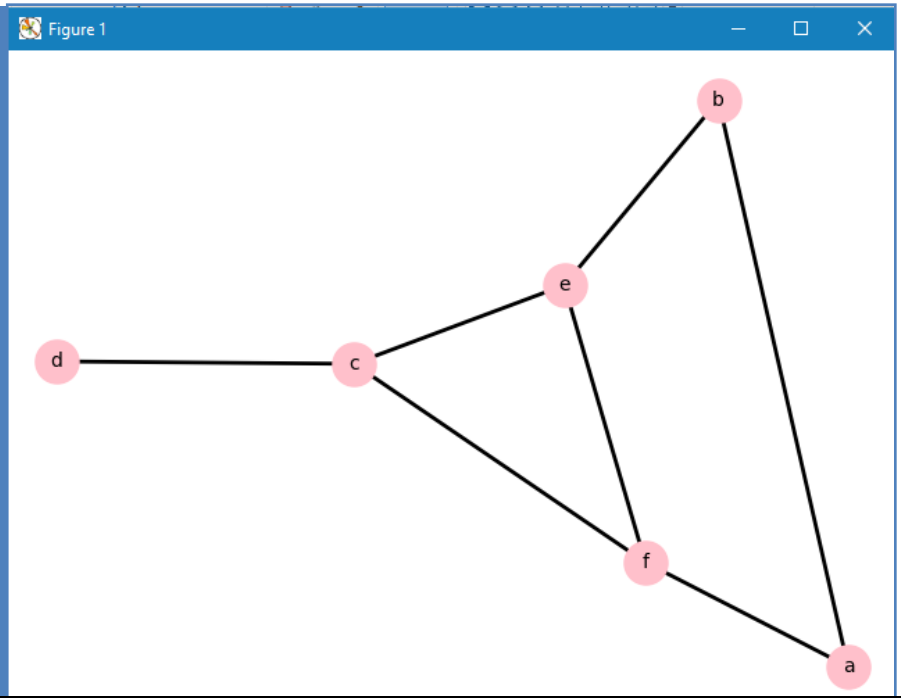

```
NodeView(('a', 'b', 'c', 'd', 'e', 'f', 'x', 23, 45))
>>> G.edges()
EdgeView([('a', 'b'), ('c', 'd'), ('c', 'f'), ('e', 'f'
>>> G.degree()  # número de enlaces de cada nodo
DegreeView({'a': 1, 'b': 1, 'c': 2, 'd': 1, 'e': 1, 'f': 2, 'x': 0, 23: 1, 45: 1})
>>> G.size()  #número de aristas
5
>>> G.order() #número de vértices
9
>>> len(G) #número de vértices ** longitud de diccionario
9
```

```
# ejemplo 21
import networkx as nx
import matplotlib.pyplot as plt


L1 = ['a','b','c','d','e','f'] L2 = [('a','b'), ('a','f'), ('b','e'
),('c','e'), ('c','d'),
                ('e','f'),('f','c')]
G = nx.Graph()
G.add_nodes_from(L1)
G.add_edges_from(L2)
list(G.adj['f'])
list(G.neighbors('f'))
G.degree['f']

M = nx.adjacency_matrix(G)
print(M.todense())
nx.draw(G,
        node_color="pink",
        edge_color="black",
        font_size=10,
        width=1,
        with_labels=True,
        node_size=500,
        )
plt.show()
```

```
>>> list(G.adj['f'])
['a', 'e', 'c']
>>> list(G.neighbors('f'))
['a', 'e', 'c']
>>> G.degree['f']
3
>>> M = nx.adjacency_matrix(G)
>>> print(M.todense()) emite matriz
[[0 1 0 0 0 1]
 [1 0 0 0 1 0]
 [0 0 0 1 1 1]
 [0 0 1 0 0 0]
 [0 1 1 0 0 1]
 [1 0 1 0 1 0]]
>>> print(M) #emite la matriz como una lista de
vectores no nulos
  (0, 1)      1
  (0, 5)      1
  (1, 0)      1
  (1, 4)      1
  (2, 3)      1
  (2, 4)      1
  (2, 5)      1
  (3, 2)      1
  (4, 1)      1
  (4, 2)      1
  (4, 5)      1
  (5, 0)      1
  (5, 2)      1
  (5, 4)      1
```



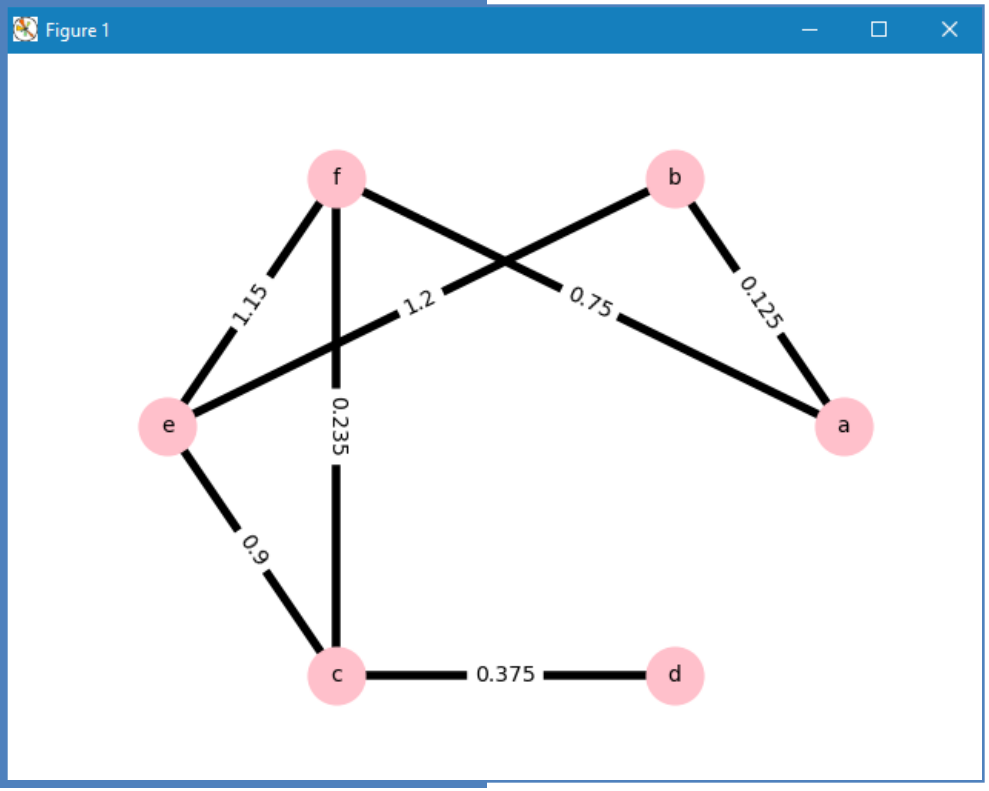
```python
# ejemplo 22
import networkx as nx
Import matplotlib.pyplot as plt

G = nx.Graph()
G.add_weighted_edges_from([('a', 'b', 0.125),
                           ('a', 'f', 0.75),
                           ('b', 'e', 1.2),
                           ('c', 'e', 0.90),
                           ('c', 'd', 0.375),
                           ('e', 'f', 1.15),
                           ('f', 'c', 0.235)])

for n, nbrs in G.adj.items():
   print(f"\n(Nodo: {n}, Enlaces: {nbrs})\n")
   for nbr, eattr in nbrs.items():
      print(f"(Enlace: {nbr}, Atributo: {eattr})")

for (u, v, wt) in G.edges.data('weight'):
   print(f"({u}, {v}, {wt})")

nx.draw(G,
      node_color="pink",
      edge_color="black",
      font_size=10,
      width=2,
      with_labels=True,
      node_size=500,
      )
plt.show()
```



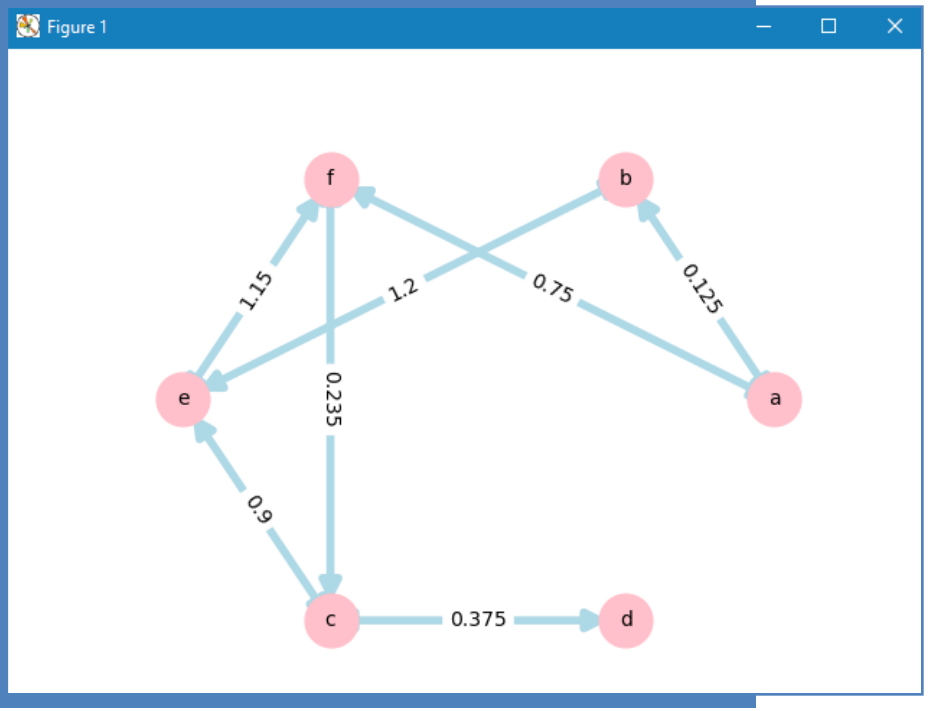

```
#  G.adj.items():
(Nodo: a, Diccionario de vecinos: {'b': {'weight': 0.125}, 'f': {'weight': 0.75}})
(Vecino: b, Atributo: {'weight': 0.125})
(Vecino: f, Atributo: {'weight': 0.75})

(Nodo: b, Diccionario de vecinos: {'a': {'weight': 0.125}, 'e': {'weight': 1.2}})
(Vecino: a, Atributo: {'weight': 0.125})
(Vecino: e, Atributo: {'weight': 1.2})
... # sigue  hasta finalizar la iteración
# G.edges.data('weight')
(a, b, 0.125)
(a, f, 0.75)
(b, e, 1.2)
(f, e, 1.15)
(f, c, 0.235)
(e, c, 0.9)
(c, d, 0.375)
```

El examen de todos los pares (no do, adyacencia) se logra usando G .adjacency() o G.adj.items().
Para los grafos no dirigidos, la ite ración de adyacencia ve cada bor de dos veces.
Los enlaces se deben dar como tu plas de 3 (u, v, w) donde w es un número.

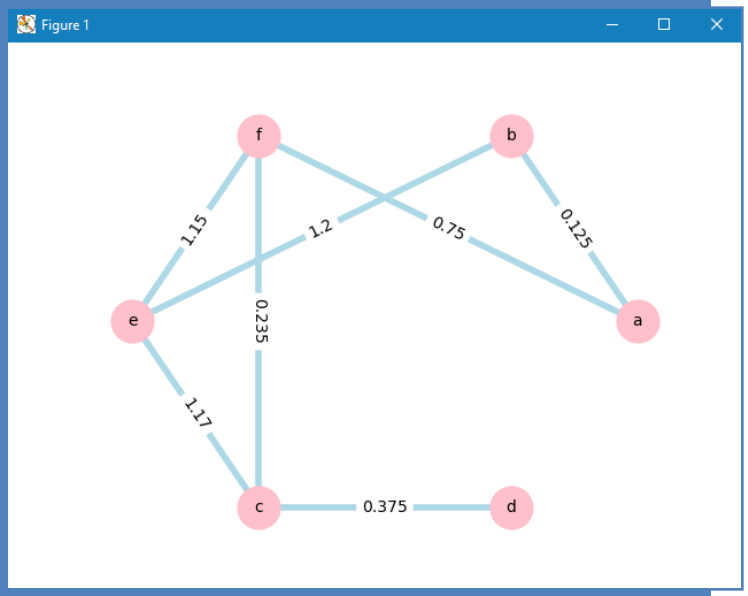
```
# ejemplo 23
import networkx as nx
import matplotlib.pyplot as plt
def emitoGraph(G, pos):
    nx.draw_networkx_nodes(G, pos, node_color="pink", node_size=700)
    # nodos
    nx.draw_networkx_labels(G, pos, font_size=10, font_family='sans-serif')
    # etiquetas de los nodos
    nx.draw_networkx_edges(G, pos, width=4)
    # enlaces
    labels = nx.get_edge_attributes(G, 'weight')
    # pesos de los nodos
    nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
    # etiquetas de los enlaces
    plt.axis('off')
    plt.show()

def cargoGraph(G):
    G.add_weighted_edges_from([('a', 'b', 0.125),
                               ('a', 'f', 0.75),
                               ('b', 'e', 1.2),
                               ('c', 'e', 0.90),
                               ('c', 'd', 0.375),
                               ('e', 'f', 1.15),
                               ('f', 'c', 0.235)])
G = nx.Graph()
cargoGraph(G)
pos = nx.shell_layout(G)
# posiciona los nodos en círculos concéntricos
emitoGraph(G, pos)
```



```
# ejemplo 24
import networkx as nx
import matplotlib.pyplot as plt

def emitoGraph(G, pos):
    nx.draw_networkx_nodes(G, pos, node_color="pink", node_size=700)
    nx.draw_networkx_labels(G, pos, font_size=10, font_family='sans-serif')

    nx.draw_networkx_edges(G, pos, edge_color='lightblue', width=4, arrowstyle= '<|-|>', arrowsize = 20)

    labels = nx.get_edge_attributes(G, 'weight')
    nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
    plt.axis('off')
    plt.show()

def cargoGraph(G):
    G.add_weighted_edges_from([('a', 'b', 0.125),
                               ('a', 'f', 0.75),
                               ('b', 'e', 1.2),
                               ('c', 'e', 0.90),
                               ('c', 'd', 0.375),
                               ('e', 'f', 1.15),
                               ('f', 'c', 0.235)])
G = nx.Graph()
# convertimos el grafo no dirigido a uno dirigido
H = G.to_directed()
#  o directamente creo un digrafo: H = nx.DiGraph()
cargoGraph(H)
pos = nx.shell_layout(H)
emitoGraph(H, pos)
```



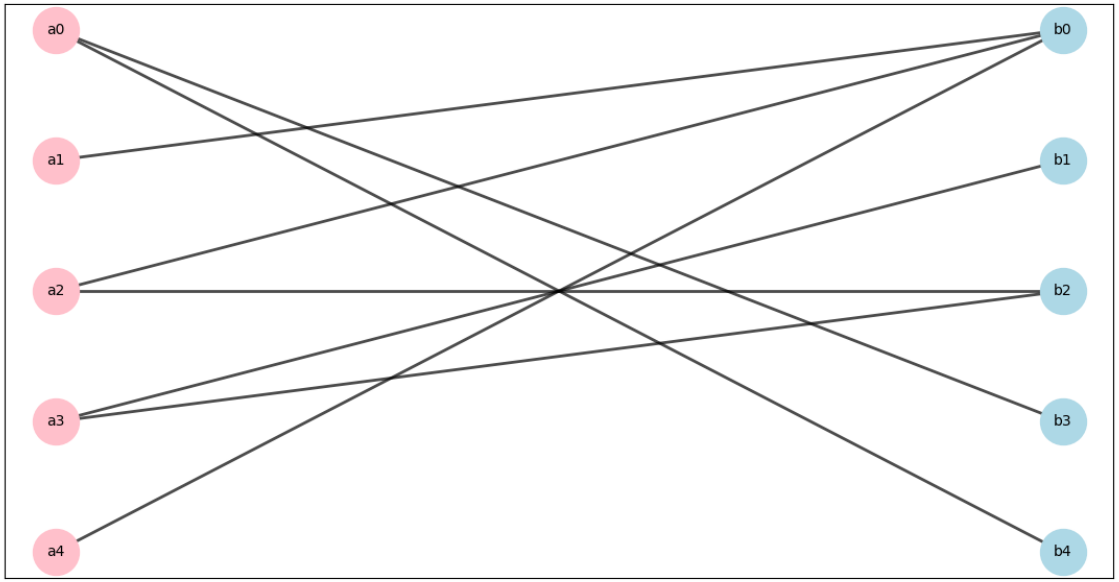
```
# ejemplo 25
import networkx as nx
import matplotlib.pyplot as plt
def emitoGraph(G, pos):
    nx.draw_networkx_nodes(G, pos, node_color="pink", node_size=700)
    nx.draw_networkx_labels(G, pos, font_size=10, font_family='sans-serif')
    nx.draw_networkx_edges(G, pos, edge_color='lightblue', width=4)
    labels = nx.get_edge_attributes(G, 'weight')
    nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
    plt.axis('off')
    plt.show()
def cargoGraph(G):
    G.add_weighted_edges_from([('a', 'b', 0.125),
                               ('a', 'f', 0.75),
                               ('b', 'e', 1.2),
                               ('c', 'e', 0.90),
                               ('c', 'd', 0.375),
                               ('e', 'f', 1.15),
                               ('f', 'c', 0.235)])
G = nx.Graph()
cargoGraph(G)
G['c']['e']['weight'] = 1.17 # Cambiamos el peso a la relación entre el nodo c y e
print("Valor del nodo c: ", G['c'])
print("Peso de la relación entre a y b: ", G['a']['b'])
print("Ruta mas corta entre a y e: ", nx.algorithms.shortest_path(G, 'a', 'e'))
print("Promedio de la ruta mas corta ", nx.algorithms.average_shortest_path_length(G))
print("Relación de ruta mas corta entre pares de nodos relacionado con e: ", dict(nx.all_pairs_shortest_path(G))['e'])
print("Ruta mas corta usando el algoritmo de Dijkstra entre a y e: ", nx.algorithms.dijkstra_path(G, 'a', 'e'))
pos = nx.shell_layout(G)
emitoGraph(G, pos)
```



```
>>>
>>> G['c']['e']['weight'] = 1.17 # Cambiamos el peso a la relacion entre el nodo c y e

>>> print("Valor del nodo c: ", G['c'])
Valor del nodo c:  {'e': {'weight': 1.17}, 'd': {'weight': 0.375}, 'f': {'weight': 0.235}}

>>> print("Peso de la relación entre a y b: ", G['a']['b'])
Peso de la relación entre a y b:  {'weight': 0.125}

>>> print("Ruta mas corta entre a y e: ", nx.algorithms.shortest_path(G, 'a', 'e'))
Ruta mas corta entre a y e:  ['a', 'b', 'e']

>>> print("Promedio de la ruta mas corta ", nx.algorithms.average_shortest_path_length(G))
Promedio de la ruta mas corta  1.6666666666666667

>>> print("Relación de ruta mas corta entre pares de nodos relacionado con e: ",
dict(nx.all_pairs_shortest_path(G))['e'])

Relación de ruta mas corta entre pares de nodos relacionado con e:  {'e': ['e'], 'b': ['e', 'b'], 'c': ['e', 'c'], 'f': ['e', 'f'], 'a':
['e', 'b', 'a'], 'd': ['e', 'c', 'd']}

>>> print("Ruta mas corta usando el algoritmo de Dijkstra entre a y e: ", nx.algorithms.dijkstra_path(G, 'a', 'e'))
Ruta mas corta usando el algoritmo de Dijkstra entre a y e:  ['a', 'b', 'e']
```

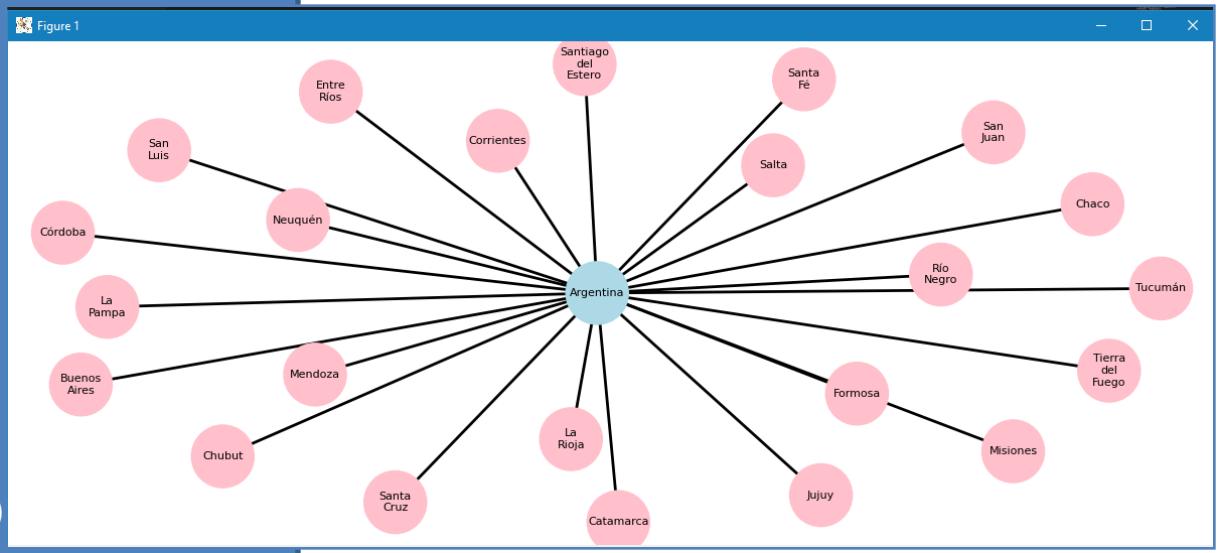
```
# ejemplo 26
import networkx as nx
import matplotlib.pyplot as plt
mat = [
   [0, 0, 0, 1, 1],
   [1, 0, 0, 0, 0],
   [1, 0, 1, 0, 0],
   [0, 1, 1, 0, 0],
   [1, 0, 0, 0, 0]
]
G = nx.Graph()
a = ["a"+str(i) for i in range(len(mat))]
b = ["b"+str(j) for j in range(len(mat[0]))]
G.add_nodes_from(a, bipartite=0)
G.add_nodes_from(b, bipartite=1)
for i in range(len(mat)):
   for j in range(len(mat[i])):
      if mat[i][j] != 0:
         G.add_edge(a[i], b[j])
pos_a = {}
const = 0.100
x = 0.100
y = 1.0
for i in range(len(a)):
   pos_a[a[i]] = [x, y-i*const]
print(pos_a)
pos_b = {}
x = 0.500
for i in range(len(b)):
   pos_b[b[i]] = [x, y-i*const]
```

```
print(pos_b)
nx.draw_networkx_nodes(G, pos_a, nodelist=a,
                       node_color="pink", node_size=1000)
nx.draw_networkx_nodes(G, pos_b, nodelist=b,
                       node_color="lightblue", node_size=1000)
pos = {}
pos.update(pos_a)
pos.update(pos_b)
print(pos)
nx.draw_networkx_labels(G, pos, font_size=10, font_family="sans-
serif")
nx.draw_networkx_edges(G, pos, width=2, alpha=0.7)
plt.show()
```



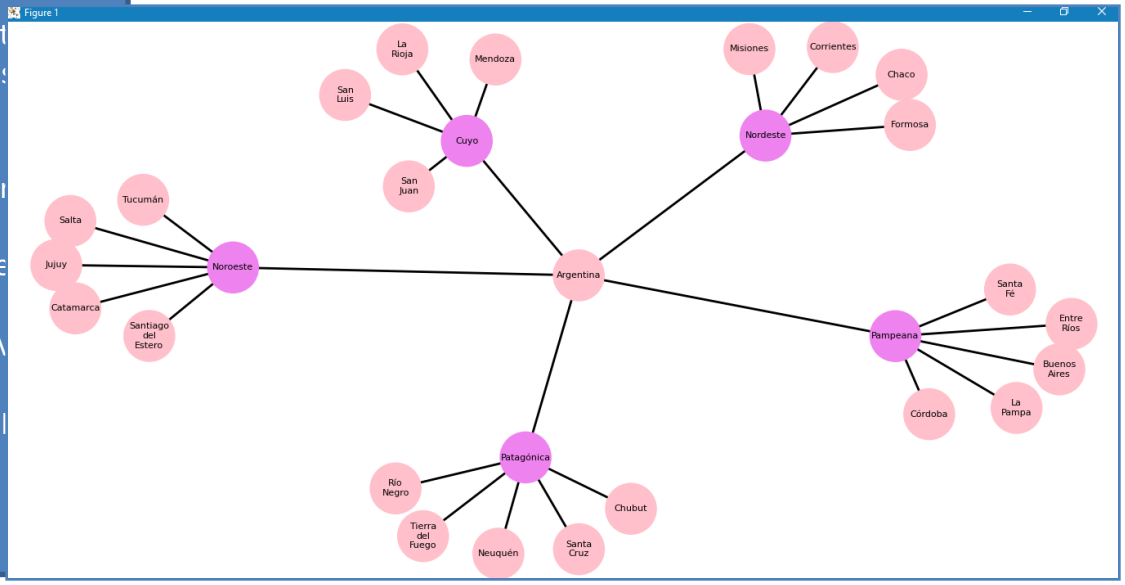
```
# ejemplo 27
import networkx as nx
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12.0, 5.0)
G = nx.Graph()
G.add_nodes_from((["Argentina", "Jujuy", "Salta", "Tucumán",
"Catamarca", "Santiago\ndel\nEstero",
                "Formosa", "Chaco", "Misiones", "Corrientes",
                "Entre\nRíos", "Santa\nFé", "Córdoba", "Buenos\nAires",
"La\nPampa",
                "San\nJuan", "San\nLuis", "Mendoza", "La\nRioja",
                "Neuquén", "Río\nNegro", "Chubut", "Santa\nCruz",
"Tierra\ndel\nFuego"]))
G.add_edges_from([("Jujuy", "Argentina"),
                ("Salta", "Argentina"),
                ("Tucumán",  "Argentina"),
                ("Catamarca", "Argentina"),
                ("Santiago\ndel\nEstero", "Argentina"),
                ("Formosa", "Argentina"),
                ("Chaco", "Argentina"),
                ("Misiones", "Argentina"),
                ("Corrientes", "Argentina"),
                ("Entre\nRíos", "Argentina"),
                ("Santa\nFé", "Argentina"),
                ("Córdoba", "Argentina"),
                ("Buenos\nAires", "Argentina"),
                ("La\nPampa", "Argentina"),
                ("San\nJuan", "Argentina"),
                ("San\nLuis", "Argentina"),
                ("Mendoza", "Argentina"),
                ("La\nRioja", "Argentina"),
                ("Neuquén", "Argentina"),
                ("Río\nNegro", "Argentina"),
                ("Chubut", "Argentina"),
                ("Santa\nCruz", "Argentina"),
                ("Tierra\ndel\nFuego", "Argentina")])
```

```
pais = ["Argentina"]
nx.draw(G,
      node_color=['lightblue' if node in pais else 'pink' for node in
G.nodes()],
      edge_color="black",
      font_size=8,
      width=2,
      with_labels=True,
      node_size=2000
      )
plt.show()

print("Nodos: ", G.number_of_nodes(), G.nodes())
print("Enlaces: ", G.number_of_edges(), G.edges())

print("Radio: %d" % nx.radius(G))
print("Diámetro: %d" % nx.diameter(G))
print("Excentricidad: %s" % nx.eccentricity(G))
print("Centro: %s" % nx.center(G))
print("Periferia: %s" % nx.periphery(G))
print("Densidad: %s" % nx.density(G))
```



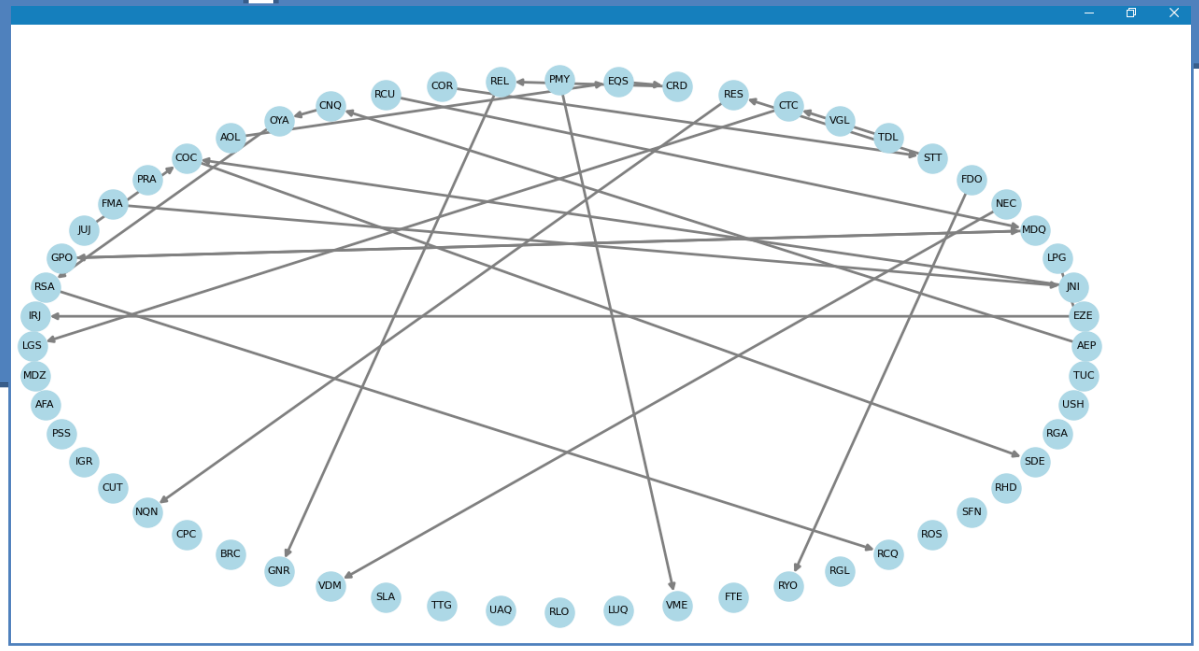
```
# ejemplo 28
import networkx as nx
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12.0, 5.0)
G = nx.Graph()
G.add_node("Argentina")
G.add_nodes_from(["Noroeste", "Nordeste", "Pampeana",
                  "Cuyo", "Patagónica"])
G.add_nodes_from((["Jujuy", "Salta", "Tucumán", "Catamarca", "
Santiago\ndel\nEstero",
                   "Formosa", "Chaco", "Misiones", "Corrientes",
                   "Entre\nRíos", "Santa\nFé", "Córdoba", "Buenos\nAire
s", "La\nPampa",
                   "San\nJuan", "San\nLuis", "Mendoza", "La\nRioja",
                   "Neuquén", "Río\nNegro", "Chubut", "Santa\nCruz", "
Tierra\ndel\nFuego"]))
G.add_edge("Noroeste", "Argentina")
G.add_edge("Nordeste", "Argentina")
G.add_edge("Pampeana", "Argentina")
G.add_edge("Cuyo", "Argentina")
G.add_edge("Patagónica", "Argentina")
lista_noroeste = [("Jujuy",  "Noroeste"), ("Salta", "Noroest
                  ("Catamarca", "Noroeste"), ("Santiago\ndel\nE
Noroeste"), ("Tucumán", "Noroeste")]
G.add_edges_from(lista_noroeste)
lista_nordeste = [("Formosa", "Nordeste"), ("Chaco", "Nor
,
                  ("Misiones", "Nordeste"), ("Corrientes", "Norde
G.add_edges_from(lista_nordeste)
lista_pampeana = [("Entre\nRíos", "Pampeana"), ("Santa\
Pampeana"),
                  ("Córdoba", "Pampeana"),("La\nPampa", "Pam
, ("Buenos\nAires", "Pampeana")]
G.add_edges_from(lista_pampeana)
```

```
lista_cuyo = [("San\nJuan", "Cuyo"), ("San\nLuis", "Cuyo"),
              ("Mendoza", "Cuyo"), ("La\nRioja", "Cuyo")]
G.add_edges_from(lista_cuyo)
lista_patagonica = [("Neuquén", "Patagónica"), ("Río\nNegro", "Patagó
nica"), ("Chubut", "Patagónica"),
                    ("Santa\nCruz", "Patagónica"), ("Tierra\ndel\nFuego", "Pata
gónica")]
G.add_edges_from(lista_patagonica)
regiones = ["Noroeste", "Nordeste", "Pampeana","Cuyo", "Patagónica"
]
nx.draw(G,
        node_color= ['pink' if not node in regiones else 'violet' for node in
G.nodes()],
        edge_color="black",
        font_size=8,
        width=2,
        with_labels=True,
        node_size=2000,
        )
plt.show()
```



```
# ejemplo 29
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = (15.0, 7.5)

aeropuertos = pd.read_csv('<ruta>/aeropuertos.csv', encoding
='latin-1')
df_a = pd.DataFrame(aeropuertos)
df_a.head(5)

vuelos = pd.read_csv("<ruta>/combi_precios.csv", encoding='l
atin-1')
df_p = pd.DataFrame(vuelos)
df_p.head(5)

df_p = df_p.iloc[0:25,0:2]
print(df_p)

DG = nx.DiGraph()

for i in range(0, len(df_a)):
   DG.add_node(df_a.iloc[i]['COD'])
   i = i + 1
print(DG.number_of_nodes())
print(DG.nodes())
```

```
for i in range(0, len(df_p)):
   DG.add_edge(df_p.iloc[i]['Origen'], df_p.iloc[i]['Destino'])
   i = i + 1
print(DG.number_of_edges())
print(DG.edges())

DG.nodes(data=True)

nx.draw_circular(DG,
                 node_color="lightblue",
                 edge_color="gray",
                 font_size=8,
                 width=2, with_labels=True, node_size=500,
                 )
plt.show()
```



¿Qué clase de grafo necesito representar?

Clase Networkx	Tipo	Auto-loops permitidos	Bordes paralelos permitidos	Documentación
Graph	no dirigido	si	No	https://networkx.github.io/documentation/stable/reference/classes/graph.html
DiGraph	dirigido	si	No	https://networkx.github.io/documentation/stable/reference/classes/digraph.html
MultiGraph	no dirigido	si	si	https://networkx.github.io/documentation/stable/reference/classes/multigraph.html
MultiDiGraph	dirigido	si	si	https://networkx.github.io/documentation/stable/reference/classes/multidigraph.html

- https://networkx.github.io/documentation/stable/reference/algorithms/index.html
- https://networkx.github.io/documentation/stable/auto_examples/index.html
- https://networkx.github.io/
- https://networkx.github.io/documentation/stable/_downloads/networkx_reference.pdf
- https://networkx.org/
- https://networkx.org/documentation/stable/tutorial.html

FIN