19360859053

Hümeyra ÇİMEN

BURSA TEKNİK UNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
BİOINFORMATİK DERSİ PROJE6 RAPOR

19360859053 HÜMEYRA ÇİMEN

> BURSA TEKNİK UNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ BİOINFORMATİK DERSİ PROJE-6 RAPOR

Grafik oluşturma Düğümsüz ve kenarsız boş bir grafik oluşturun.

```
[1]  1 import networkx as nx
     2 G = nx.Graph()
```

```
1 print(G)
2 # Çıktı: Graph()
3
4 print(type(G))
5 # Çıktı: <class 'networkx.classes.graph.Graph'>
6
```

```
Graph with 0 nodes and 0 edges
<class 'networkx.classes.graph.Graph'>
```

Düğümler

```
[3]  1 G.add_node(1) # 1 NODE DÜĞÜM EKLENDI
     2 print(G)
```

```
Graph with 1 nodes and 0 edges
```

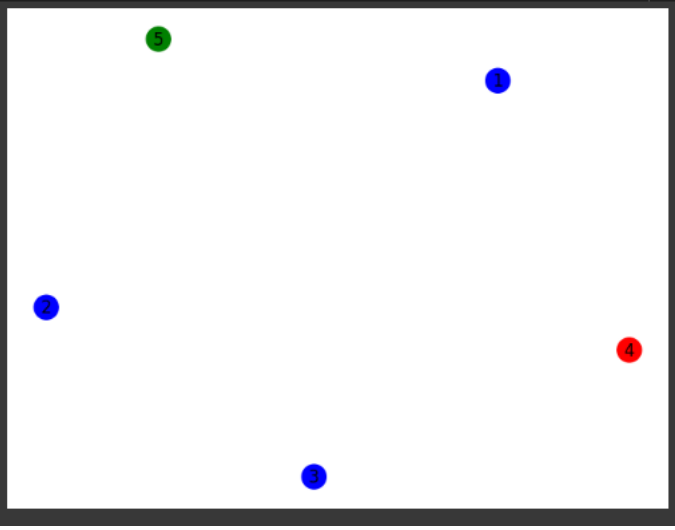
```
1 G.add_nodes_from([2, 3])
2 print(G)
```

```
Graph with 3 nodes and 0 edges
```

```
[5]  1 G.add_nodes_from([
     2     (4, {"color": "red"}),
     3     (5, {"color": "green"}),
     4 ])
     5 print(G) # 4 VE 5 EKLENDI
```

```
Graph with 5 nodes and 0 edges
```

```
# Renkleri ayarla
node_colors = [data.get('color', 'blue') for _, data in G.nodes(data=True)]

# Grafiği çiz
nx.draw(G, with_labels=True, node_color=node_colors)

# Grafiği göster
plt.show()
```



Bir grafikteki düğümler başka bir grafiğe dahil edilebilir:

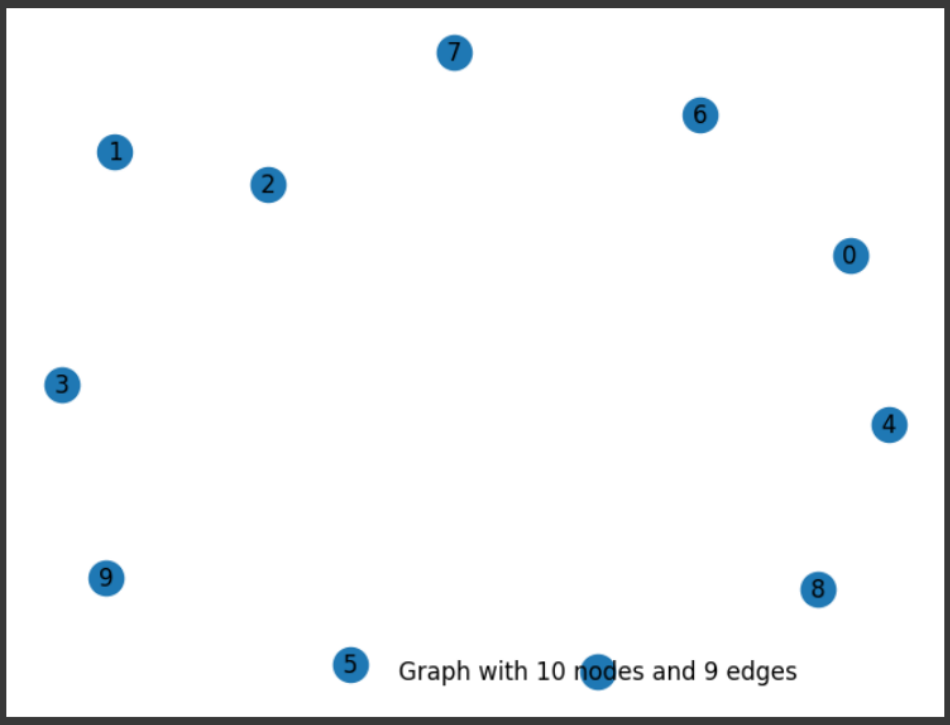
```
[6]  print("G: ",G)
     H = nx.path_graph(10)
     #H grafiğinin düğümleri G grafiğine ekleni
     G.add_nodes_from(H)
     print("G: ",G)
     print("H: ",H)
```

```
G:  Graph with 5 nodes and 0 edges
G:  Graph with 10 nodes and 0 edges
H:  Graph with 10 nodes and 9 edges
```

```
[7]  G.add_node(H)
     print("G: ",G)
     #nx.draw(G, with_labels=True)
     #plt.show()
```

```
G:  Graph with 11 nodes and 0 edges
```

```
[8]  1 import matplotlib.pyplot as plt
     2
     3 nx.draw(G, with_labels=True)
     4 plt.show()
```



Edges

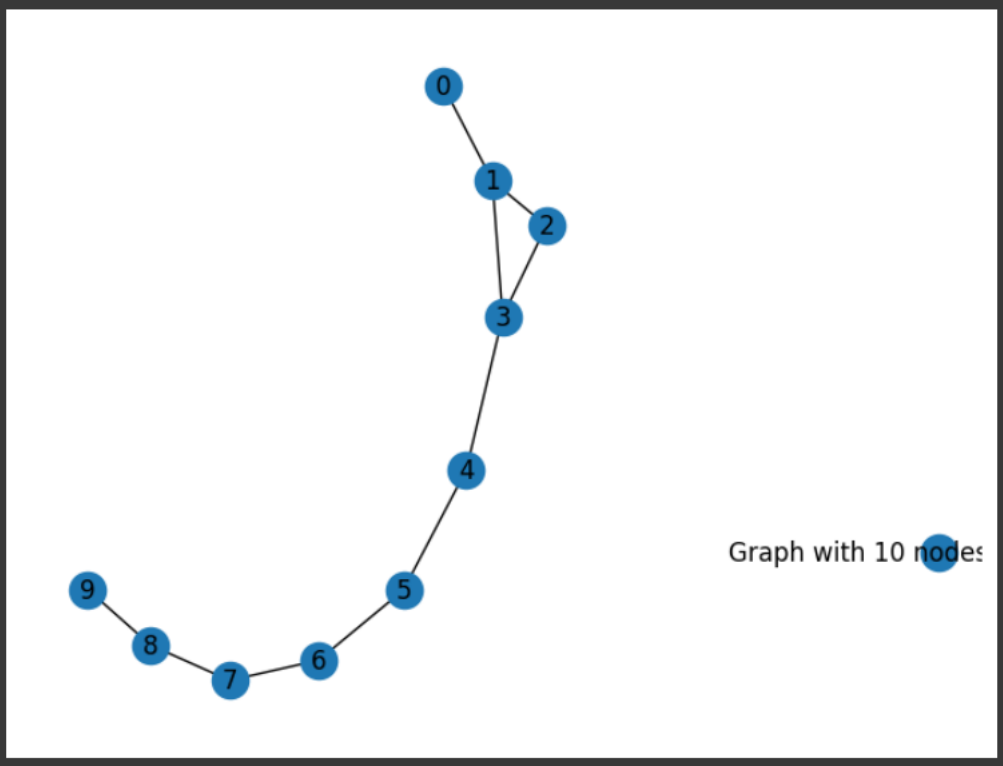
```
[9]  1 G.add_edge(1, 2)
     2 e = (2, 3)
     3 G.add_edge(*e)  # unpack edge tuple*
     4 print(G)
```

```
Graph with 11 nodes and 2 edges
```

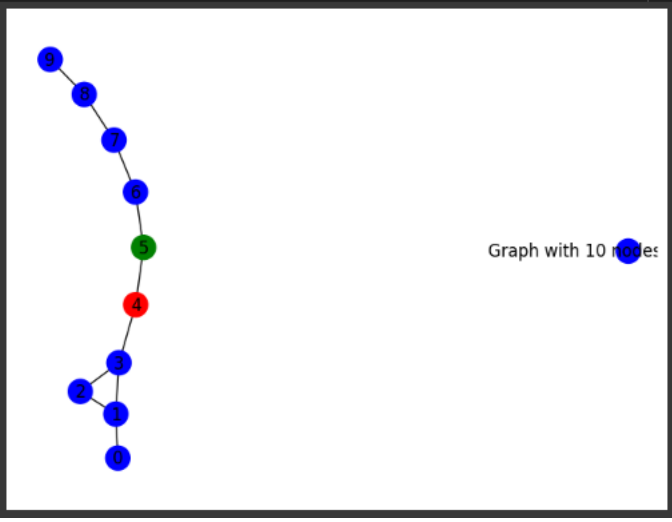
```
[10]  1 G.add_edges_from([(1, 2), (1, 3)])
```

```
[11]  1 G.add_edges_from(H.edges)
```

```
1 nx.draw(G, with_labels=True)
2 plt.show()
```




```
1
2 # Renkleri ayarla
3 node_colors = [data.get('color', 'blue') for _, data in G.nodes(data=True)]
4
5 # Grafiği çiz
6 nx.draw(G, with_labels=True, node_color=node_colors)
7
8 # Grafiği göster
9 plt.show()
```

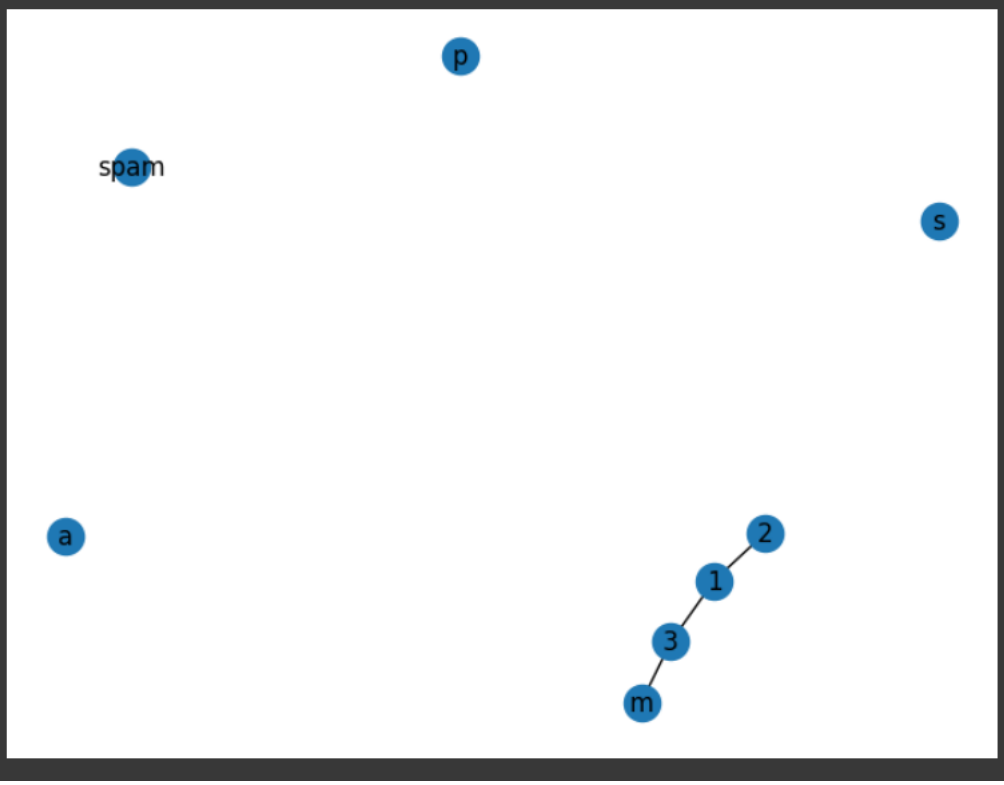



```
[13]  1 G.clear()
```

```
nx.draw(G, with_labels=True)
plt.show()
```

```
G.add_edges_from([(1, 2), (1, 3)])# 1 İLE 2 VE 3 BAGLI
G.add_node(1)
G.add_edge(1, 2)
G.add_node("spam")        # adds node "spam"
G.add_nodes_from("spam")  # adds 4 nodes: 's', 'p', 'a', 'm'
G.add_edge(3, 'm')

```

```
[16]  1 nx.draw(G, with_labels=True)
      2 plt.show()
```




```
[17]  1 G.number_of_nodes() # 1/2/3/m/s/p/a/spam
```

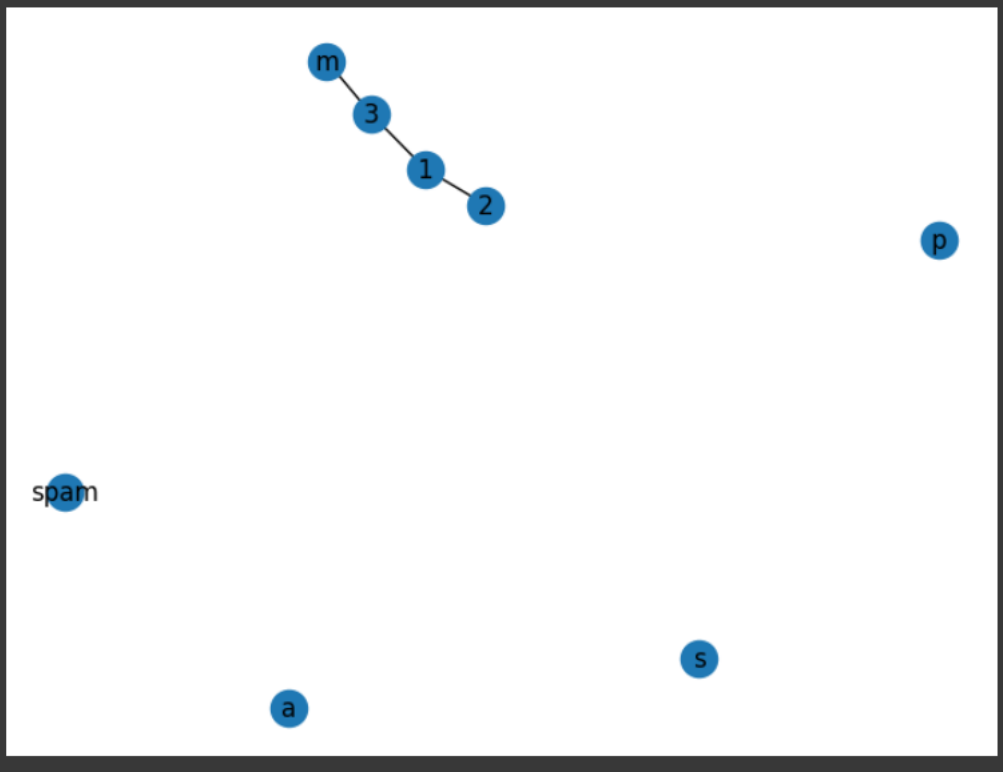
```
8
```

```
[18]  1 G.number_of_edges() # 3.m/ 1.3/ 1.2
```

```
3
```

```
[19]  1 DG = nx.DiGraph() # bir yönlü graf oluşturulur.
      2 DG.add_edge(2, 1)   # adds the nodes in order 2, 1** 2'den 1'e yönlü bir bağlantı ekleni
      3 DG.add_edge(1, 3) # 1'den 3'e yönlü bir bağlantı eklen
      4 DG.add_edge(2, 4)
      5 DG.add_edge(1, 2)
      6 assert list(DG.successors(2)) == [1, 4] #2 düğümünden hangi düğümlere bağlantılar olduğunu kontrol edilir
      7 assert list(DG.edges) == [(2, 1), (2, 4), (1, 3), (1, 2)] #2 düğümünden hangi düğümlere bağlantılar olduğunu kontrol edilir
```

```
[20]  1 nx.draw(G, with_labels=True)
      2 plt.show()
```



Examining elements of a graph

```
[21]  1 list(G.nodes)
      2
```

```
[1, 2, 3, 'spam', 's', 'p', 'a', 'm']
```

```
[22]  1 list(G.edges)
```

```
[(1, 2), (1, 3), (3, 'm')]
```

```
[23]  1 list(G.adj[1])  # or list(G.neighbors(1))
```

```
[2, 3]
```

```
[24]  1 G.degree[1]  # the number of edges incident to 1
```

```
2
```

```
[25]  1 G.edges([2, 'm'])
```

```
EdgeDataView([(2, 1), ('m', 3)])
```

```
[26]  1 G.degree([2, 3])
```

```
DegreeView({2: 1, 3: 2})
```

Removing elements from a graph

```
[27]  1 G.remove_node(2)
      2
```

```
[28]  1 G.remove_nodes_from("spam")
```

```
[29]  1 list(G.nodes)
```

```
[1, 3, 'spam']
```

```
[30]  1 G.remove_edge(1, 3)
```

Using the graph constructors

```
[31]  1 G.add_edge(1, 2)
      2 H = nx.DiGraph(G)  # create a DiGraph using the connections from G
      3 list(H.edges())
```

```
[(1, 2), (2, 1)]
```

```
[32]  1 edgelist = [(0, 1), (1, 2), (2, 3)]
      2 H = nx.Graph(edgelist)  # create a graph from an edge list
      3 list(H.edges())
```

```
[(0, 1), (1, 2), (2, 3)]
```

```
[33]  1 adjacency_dict = {0: (1, 2), 1: (0, 2), 2: (0, 1)}
      2 H = nx.Graph(adjacency_dict)  # create a Graph dict mapping nodes to nbrs
      3 list(H.edges())
```

```
[(0, 1), (0, 2), (1, 2)]
```

Accessing edges and neighbors

```
[34]  1 G = nx.Graph([(1, 2, {"color": "yellow"})])
      2 G[1]  # same as G.adj[1]
```

```
AtlasView({2: {'color': 'yellow'}})
```

```
[35]  1 G[1][2]
```

```
{'color': 'yellow'}
```

```
[36]  1 G.edges[1, 2]
```

```
{'color': 'yellow'}
```

```
[37]  1 G.add_edge(1, 3)
      2 G[1][3]['color'] = "blue"
      3 G.edges[1, 2]['color'] = "red"
      4 G.edges[1, 2]
```

```
{'color': 'red'}
```

```
[38]  1 FG = nx.Graph()
      2 FG.add_weighted_edges_from([(1, 2, 0.125), (1, 3, 0.75), (2, 4, 1.2), (3, 4, 0.375)])
      3 for n, nbrs in FG.adj.items():
      4    for nbr, eattr in nbrs.items():
      5        wt = eattr['weight']
      6        if wt < 0.5: print(f"({n}, {nbr}, {wt:.3})")
```

```
(1, 2, 0.125)
(2, 1, 0.125)
(3, 4, 0.375)
(4, 3, 0.375)
```

```
[39]  1 for (u, v, wt) in FG.edges.data('weight'):
      2     if wt < 0.5:
      3         print(f"({u}, {v}, {wt:.3})")
```

```
(1, 2, 0.125)
(3, 4, 0.375)
```

Graph attributes

```
[40]  1 G = nx.Graph(day="Friday")
      2 G.graph
```

```
{'day': 'Friday'}
```

```
[41]  1 G.graph['day'] = "Monday"
      2 G.graph
```

```
{'day': 'Monday'}
```

Node attributes

```
[42]  1 G.add_node(1, time='5pm')
      2 G.add_nodes_from([3], time='2pm')
      3 G.nodes[1]
```

```
{'time': '5pm'}
```

```
[43]  1 G.nodes[1]['room'] = 714
      2 G.nodes.data()
```

```
NodeDataView({1: {'time': '5pm', 'room': 714}, 3: {'time': '2pm'}})
```

Edge Attributes

```
[44]  1 G.add_edge(1, 2, weight=4.7 ) #ağırlıklı bir kenar ekler ve bu kenar 1 ve 2 düğümleri arasındadır. Kenarın ağırlığı 4.7 olarak belirtilir.
      2 G.add_edges_from([(3, 4), (4, 5)], color='red') # 3 ve 4 düğümleri arasında iki kırmızı renkli kenar ekler.
      3 G.add_edges_from([(1, 2, {'color': 'blue'}), (2, 3, {'weight': 8})]) #1 ve 2 düğümleri arasındaki bir mavi renkli kenar ve 2 ve 3 düğümleri arasındaki bir 8 ağırlığındaki kenar ekler.
      4 G[1][2]['weight'] = 4.7  # 1 ve 2 düğümleri arasındaki kenarın ağırlığı 4.7 olarak güncellenir.
      5 G.edges[3, 4]['weight'] = 4.2 # 3 ve 4 düğümleri arasındaki kenarın ağırlığı 4.2 olarak güncelleni
      6
```

Directed graphs

```
[45]  1 DG = nx.DiGraph()
      2 DG.add_weighted_edges_from([(1, 2, 0.5), (3, 1, 0.75)])
      3 DG.out_degree(1, weight='weight')
      4
```

```
0.5
```

```
[46]  1 DG.degree(1, weight='weight')
```

```
1.25
```

```
[47]  1 list(DG.successors(1))
```

```
[2]
```

```
[48]  1 list(DG.neighbors(1))
```

```
[2]
```

```
[49]  1 H = nx.Graph(G)  # create an undirected graph H from a directed graph G
      2
```

Multigraphs

```
[50]  1 MG = nx.MultiGraph()
      2 MG.add_weighted_edges_from([(1, 2, 0.5), (1, 2, 0.75), (2, 3, 0.5)])
      3 dict(MG.degree(weight='weight'))
```

```
{1: 1.25, 2: 1.75, 3: 0.5}
```

```
[51]  1 GG = nx.Graph()
      2 for n, nbrs in MG.adjacency():
      3    for nbr, edict in nbrs.items():
      4        minvalue = min([d['weight'] for d in edict.values()])
      5        GG.add_edge(n, nbr, weight = minvalue)
      6
```

```
[52]  1 nx.shortest_path(GG, 1, 3)
```

```
[1, 2, 3]
```

Graph generators and graph operations

3. Using a (constructive) generator for a classic graph, e.g

```
[53]  1 K_5 = nx.complete_graph(5)
      2 K_3_5 = nx.complete_bipartite_graph(3, 5)
      3 barbell = nx.barbell_graph(10, 10)
      4 lollipop = nx.lollipop_graph(10, 20)
```

4. Using a stochastic graph generator, e.g,

```
[54]  1 er = nx.erdos_renyi_graph(100, 0.15)
      2 ws = nx.watts_strogatz_graph(30, 3, 0.1)
      3 ba = nx.barabasi_albert_graph(100, 5)
      4 red = nx.random_lobster(100, 0.9, 0.9)
```

5. Reading a graph stored in a file using common graph formats

```
[55]  1 nx.write_gml(red, "path.to.file")
      2 mygraph = nx.read_gml("path.to.file")
```

Analyzing graphs

```
[56]  1 G = nx.Graph()
      2 G.add_edges_from([(1, 2), (1, 3)])
      3 G.add_node("spam")       # adds node "spam"
      4 list(nx.connected_components(G))
```

```
[{1, 2, 3}, {'spam'}]
```

```
[57]  1 sorted(d for n, d in G.degree())
```

```
[0, 1, 1, 2]
```

```
[58]  1 nx.clustering(G)
```

```
{1: 0, 2: 0, 3: 0, 'spam': 0}
```

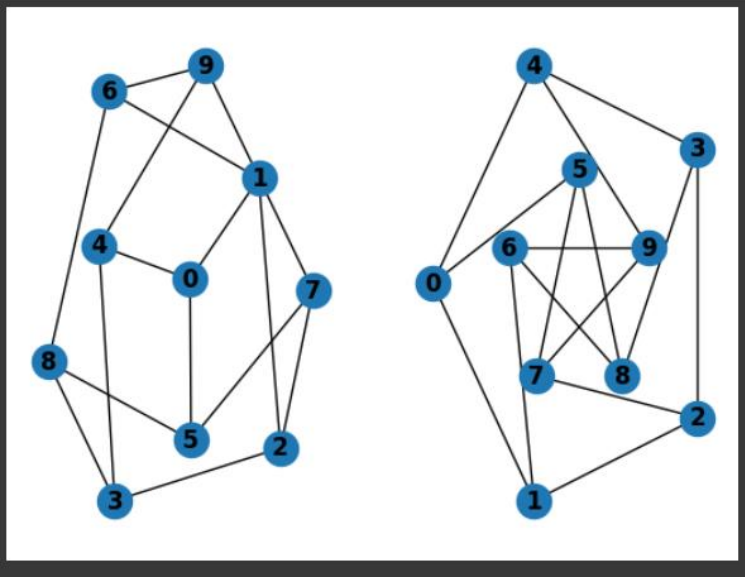
```
[59]  1 sp = dict(nx.all_pairs_shortest_path(G))
      2 sp[3]
```

```
{3: [3], 1: [3, 1], 2: [3, 1, 2]}
```

Drawing graphs

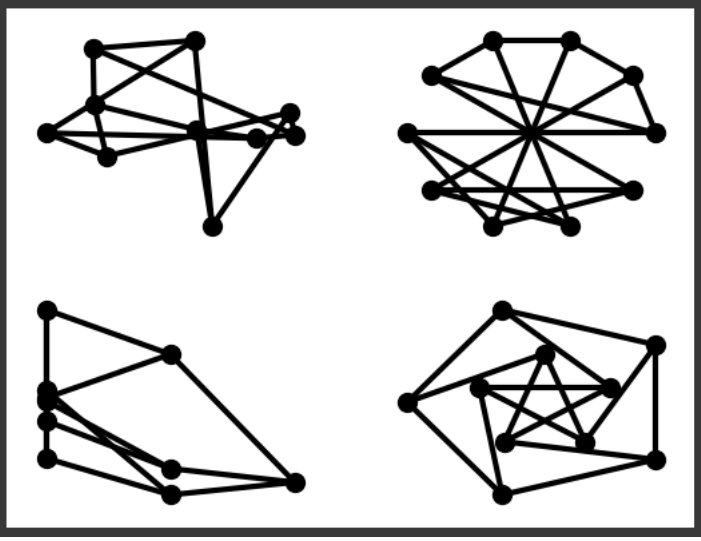
```
[60]  1 import matplotlib.pyplot as plt
```

```
[61]  1 G = nx.petersen_graph()
      2 subax1 = plt.subplot(121)
      3 nx.draw(G, with_labels=True, font_weight='bold')
      4 subax2 = plt.subplot(122)
      5 nx.draw_shell(G, nlist=[range(5, 10), range(5)], with_labels=True, font_weight='bold')
```

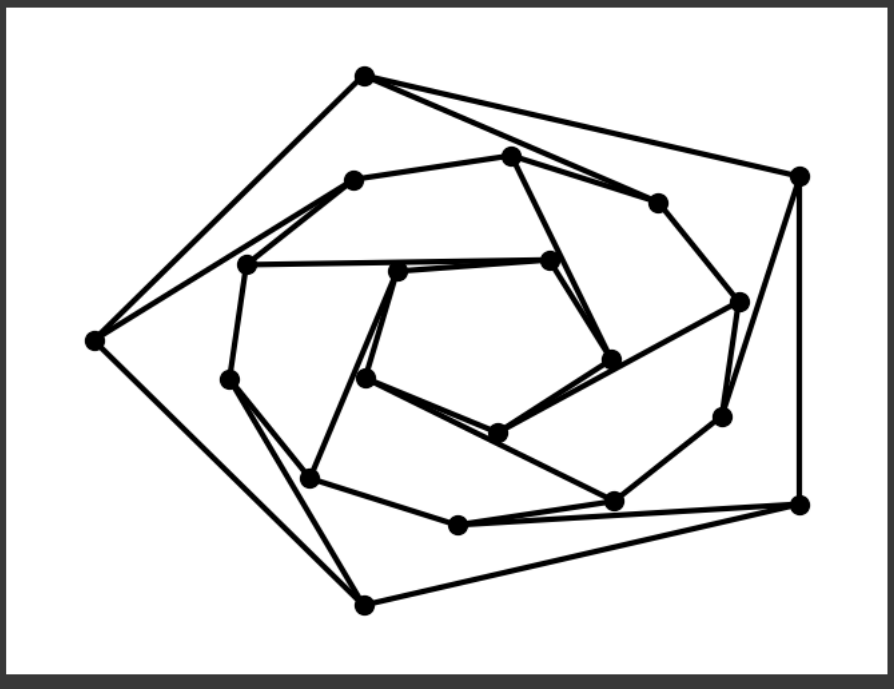



```
[62]  1 plt.show()
```

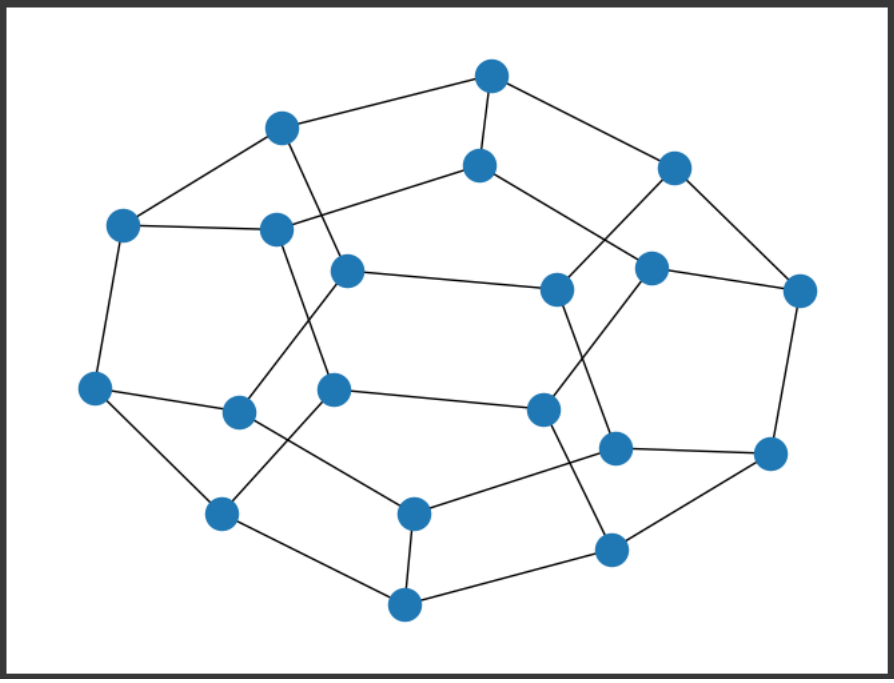
```
[63]  1 options = {
      2     'node_color': 'black',
      3     'node_size': 100,
      4     'width': 3,
      5 }
      6 subax1 = plt.subplot(221)
      7 nx.draw_random(G, **options)
      8 subax2 = plt.subplot(222)
      9 nx.draw_circular(G, **options)
     10 subax3 = plt.subplot(223)
     11 nx.draw_spectral(G, **options)
     12 subax4 = plt.subplot(224)
     13 nx.draw_shell(G, nlist=[range(5,10), range(5)], **options)
```



```
[64]  1 G = nx.dodecahedral_graph()
      2 shells = [[2, 3, 4, 5, 6], [8, 1, 0, 19, 18, 17, 16, 15, 14, 7], [9, 10, 11, 12, 13]]
      3 nx.draw_shell(G, nlist=shells, **options)
```



```
[65]  1 nx.draw(G)
      2 plt.savefig("path.png")
```



```
[66]  1 !sudo apt-get install graphviz graphviz-dev
      2
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libgraphviz-dev' instead of 'graphviz-dev'
graphviz is already the newest version (2.42.2-3build2).
The following additional packages will be installed:
  libgail-common libgail18 libgtk2.0-0 libgtk2.0-bin libgtk2.0-common
  libgvc6-plugins-gtk libxdot4
Suggested packages:
  gvfs
The following NEW packages will be installed:
  libgail-common libgail18 libgraphviz-dev libgtk2.0-0 libgtk2.0-bin
  libgtk2.0-common libgvc6-plugins-gtk libxdot4
0 upgraded, 8 newly installed, 0 to remove and 24 not upgraded.
Need to get 2,148 kB of archives.
After this operation, 7,427 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 libgtk2.0-common all 2.24.32-4ubuntu4 [126 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 libgtk2.0-0 amd64 2.24.32-4ubuntu4 [1,791 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 libgail18 amd64 2.24.32-4ubuntu4 [14.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal/main amd64 libgail-common amd64 2.24.32-4ubuntu4 [116 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal/universe amd64 libxdot4 amd64 2.42.2-3build2 [15.4 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/universe amd64 libgvc6-plugins-gtk amd64 2.42.2-3build2 [20.6 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/universe amd64 libgraphviz-dev amd64 2.42.2-3build2 [57.2 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/main amd64 libgtk2.0-bin amd64 2.24.32-4ubuntu4 [7,728 B]
Fetched 2,148 kB in 1s (1,441 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76, <> line 8.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package libgtk2.0-common.
(Reading database ... 122518 files and directories currently installed.)
Preparing to unpack .../0-libgtk2.0-common_2.24.32-4ubuntu4_all.deb ...
Unpacking libgtk2.0-common (2.24.32-4ubuntu4) ...
Selecting previously unselected package libgtk2.0-0:amd64.
Preparing to unpack .../1-libgtk2.0-0_2.24.32-4ubuntu4_amd64.deb ...
Unpacking libgtk2.0-0:amd64 (2.24.32-4ubuntu4) ...
Selecting previously unselected package libgail18:amd64.
Preparing to unpack .../2-libgail18_2.24.32-4ubuntu4_amd64.deb ...
Unpacking libgail18:amd64 (2.24.32-4ubuntu4) ...
Selecting previously unselected package libgail-common:amd64.
Preparing to unpack .../3-libgail-common_2.24.32-4ubuntu4_amd64.deb ...
Unpacking libgail-common:amd64 (2.24.32-4ubuntu4) ...
Selecting previously unselected package libxdot4:amd64.
Preparing to unpack .../4-libxdot4_2.42.2-3build2_amd64.deb ...
Unpacking libxdot4:amd64 (2.42.2-3build2) ...
Selecting previously unselected package libgvc6-plugins-gtk.
Preparing to unpack .../5-libgvc6-plugins-gtk_2.42.2-3build2_amd64.deb ...
Unpacking libgvc6-plugins-gtk (2.42.2-3build2) ...
Selecting previously unselected package libgraphviz-dev:amd64.
Preparing to unpack .../6-libgraphviz-dev_2.42.2-3build2_amd64.deb ...
```

```
Unpacking libgvc6-plugins-gtk (2.42.2-3build2) ...
Selecting previously unselected package libgraphviz-dev:amd64.
Preparing to unpack .../6-libgraphviz-dev_2.42.2-3build2_amd64.deb ...
Unpacking libgraphviz-dev:amd64 (2.42.2-3build2) ...
Selecting previously unselected package libgtk2.0-bin.
Preparing to unpack .../7-libgtk2.0-bin_2.24.32-4ubuntu4_amd64.deb ...
Unpacking libgtk2.0-bin (2.24.32-4ubuntu4) ...
Setting up libxdot4:amd64 (2.42.2-3build2) ...
Setting up libgtk2.0-common (2.24.32-4ubuntu4) ...
Setting up libgtk2.0-0:amd64 (2.24.32-4ubuntu4) ...
Setting up libgvc6-plugins-gtk (2.42.2-3build2) ...
Setting up libgail18:amd64 (2.24.32-4ubuntu4) ...
Setting up libgtk2.0-bin (2.24.32-4ubuntu4) ...
Setting up libgail-common:amd64 (2.24.32-4ubuntu4) ...
Setting up libgraphviz-dev:amd64 (2.42.2-3build2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.9) ...
```

[66]

[67]
```
!pip install pygraphviz
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pygraphviz
  Downloading pygraphviz-1.10.zip (120 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 120.6/120.6 kB 5.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: pygraphviz
  Building wheel for pygraphviz (setup.py) ... done
  Created wheel for pygraphviz: filename=pygraphviz-1.10-cp310-cp310-linux_x86_64.whl size=184045 sha256=6207bf547316113c8b48ea31ea75d2861d9cea9bd6b90f8c98ba41f7f1a6e3ff
  Stored in directory: /root/.cache/pip/wheels/e9/50/02/d9d68f6c947a928e517d5cd9af0ab007c1274fdba95fa9cbe3
Successfully built pygraphviz
Installing collected packages: pygraphviz
Successfully installed pygraphviz-1.10
```

file.dot uzantılı dosyayı açmak için eklentiler eklendi

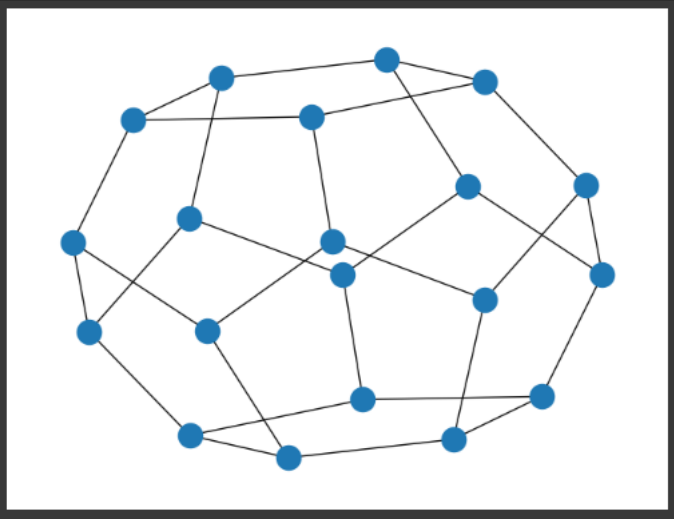
[68]
```
!sudo dnf install graphviz graphviz-devel

```

```
sudo: dnf: command not found
```

[69]
```
!pip install pygraphviz
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pygraphviz in /usr/local/lib/python3.10/dist-packages (1.10)
```

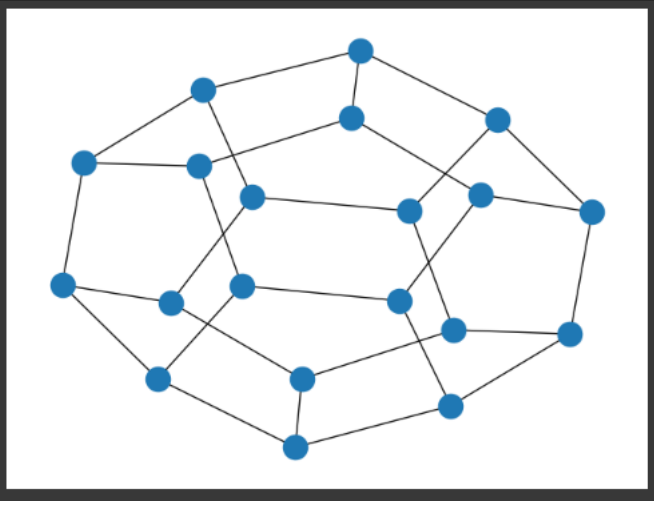
[70]
```
from networkx.drawing.nx_pydot import write_dot
pos = nx.nx_agraph.graphviz_layout(G)
nx.draw(G, pos=pos)
write_dot(G, 'file.dot')

```



ÇALIŞTIRILAN KODLARIN ÇIKTILARIN GÖSTERİLMESİ

1 path.png çıktısının gösterilmesi (ÇIKTILARI KOD İLE GÖSTERDİM)

```
[71]  1 from PIL import Image
      2
      3 # png dosyasının yolu ve adı
      4 image_path = "/content/path.png"
      5
      6 # dosyayı açma
      7 im = Image.open(image_path)
      8
      9 # görüntüyü ekranda gösterme
     10 im.show()
     11
```



```
with open("/content/path.to.file", "r") as f:
    content = f.read()
print(content)
```

```
edge [
  source 6860
  target 6872
]
edge [
  source 6860
  target 6873
]
edge [
  source 6860
  target 6874
]
edge [
  source 6860
  target 6875
]
edge [
  source 6860
  target 6876
]
edge [
  source 6860
  target 6877
]
edge [
  source 6860
  target 6878
]
edge [
  source 6860
  target 6879
]
edge [
  source 6860
  target 6880
]
edge [
  source 6860
  target 6881
]
edge [
  source 6860
  target 6882
]
edge [
  source 6860
  target 6883
]
edge [
  source 6860
```

[72]

```
    edge [
      source 6860
      target 6873
    ]
    edge [
      source 6860
      target 6874
    ]
    edge [
      source 6860
      target 6875
    ]
    edge [
      source 6860
      target 6876
    ]
    edge [
      source 6860
      target 6877
    ]
    edge [
      source 6860
      target 6878
    ]
    edge [
      source 6860
      target 6879
    ]
    edge [
      source 6860
      target 6880
    ]
    edge [
      source 6860
      target 6881
    ]
    edge [
      source 6860
      target 6882
    ]
    edge [
      source 6860
      target 6883
    ]
    edge [
      source 6860
      target 6884
    ]
    edge [
      source 6860
      target 6885
    ]
  ]
```