Elective Course

Course Code: CS4103

Autumn 2025-26

Lecture #9

Artificial Intelligence for Data Science

Week-3:

Python Primer for AI [Part-III]

Course Instructor:

Dr. Monidipa Das

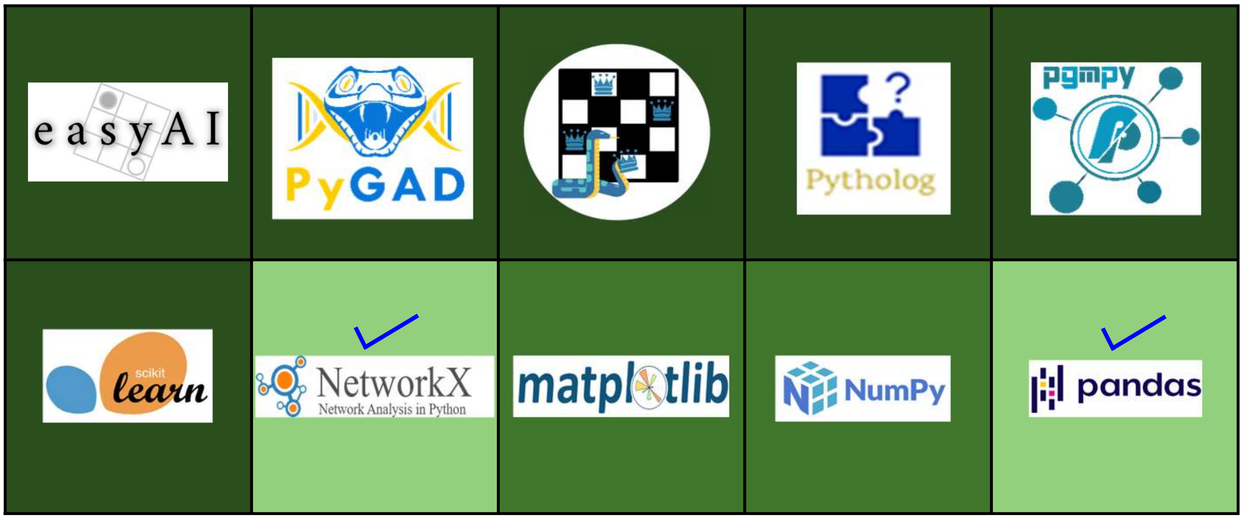
Assistant Professor

Department of Computational and Data Sciences

Indian Institute of Science Education and Research Kolkata, India 741246

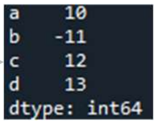
A Few Python Libraries for AI

2



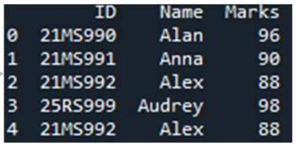
Pandas Data Structures

```
import pandas as pd
#Series:
s = pd.Series([10, -11, 12, 13], index=['a', 'b', 'c', 'd'])
print(s)
print(s['c’]) #output should be 12
```



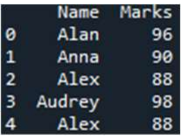
```
a    10
b   -11
c    12
d    13
dtype: int64
```

```
#Dataframe:
data = {'ID': ['21MS990', '21MS991', '21MS992', '25RS999','21MS992'],
        'Name': ['Alan', 'Anna', 'Alex', 'Audrey','Alex'],
        'Marks': [96, 90, 88, 98, 88]}
df = pd.DataFrame(data,columns=list(data.keys()))

print(df)
```



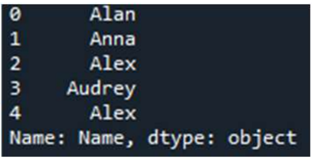
```
        ID    Name  Marks
0  21MS990    Alan     96
1  21MS991    Anna     90
2  21MS992    Alex     88
3  25RS999  Audrey     98
4  21MS992    Alex     88
```

```
#Subset Variables- columns
#Select multiple columns with specific names.
print(df[['Name', 'Marks’]])
```



```
     Name  Marks
0    Alan     96
1    Anna     90
2    Alex     88
3  Audrey     98
4    Alex     88
```

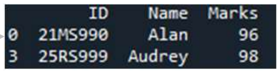
```
#Select single column with specific name.
print(df['Name’])
```



```
0      Alan
1      Anna
2      Alex
3    Audrey
4      Alex
Name: Name, dtype: object
```

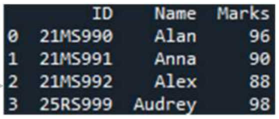
Pandas: Subset Observations - rows

```
#Extract rows that meet logical criteria
print(df[df.Marks > 90])
```



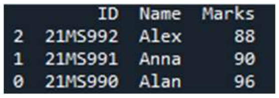
```
        ID    Name  Marks
0  21MS990    Alan     96
3  25RS999  Audrey     98
```

```
#Remove duplicate rows
df=df.drop_duplicates()
print(df)
```



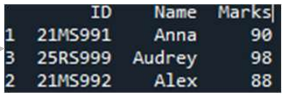
```
        ID    Name  Marks
0  21MS990    Alan     96
1  21MS991    Anna     90
2  21MS992    Alex     88
3  25RS999  Audrey     98
```

```
#Randomly select fraction of rows
print(df.sample(frac=0.8))
```



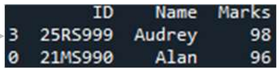
```
        ID  Name  Marks
2  21MS992  Alex     88
1  21MS991  Anna     90
0  21MS990  Alan     96
```

```
#Randomly select 3 rows
print(df.sample(n=3))
```



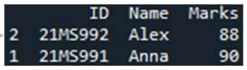
```
        ID    Name  Marks
1  21MS991    Anna     90
3  25RS999  Audrey     98
2  21MS992    Alex     88
```

```
#Select and order top 2 entries
print(df.nlargest(2, 'Marks’))
```



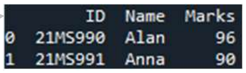
```
        ID    Name  Marks
3  25RS999  Audrey     98
0  21MS990    Alan     96
```

```
#Select and order bottom 2 entries
print(df.nsmallest(2, 'Marks’))
```

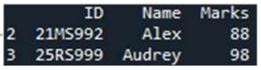


```
        ID  Name  Marks
2  21MS992  Alex     88
1  21MS991  Anna     90
```

```
print(df.head(2)) #Select first 2 rows
print(df.tail(2)) #Select last 2 rows
```



```
        ID  Name  Marks
0  21MS990  Alan     96
1  21MS991  Anna     90
```

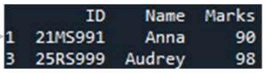


```
        ID    Name  Marks
2  21MS992    Alex     88
3  25RS999  Audrey     98
```

Dr. Monidipa Das, Department of CDS, IISER Kolkata

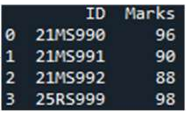
Pandas: Subset - rows and columns

```
#Select 2nd and 4th row
print(df.iloc[[1,3]])
```



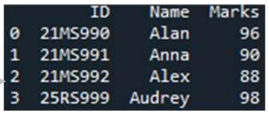
```
        ID    Name  Marks
1  21MS991    Anna     90
3  25RS999  Audrey     98
```

```
#Select 1st and 3rd column
print(df.iloc[:, [0, 2]])
```



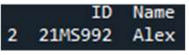
```
        ID  Marks
0  21MS990     96
1  21MS991     90
2  21MS992     88
3  25RS999     98
```

```
#Select all columns between 'ID' and 'Marks’ (inclusive)
print(df.loc[:, 'ID':'Marks’])
```



```
        ID    Name  Marks
0  21MS990    Alan     96
1  21MS991    Anna     90
2  21MS992    Alex     88
3  25RS999  Audrey     98
```

```
#Select rows meeting logical condition, and only the specific columns
print(df.loc[df['Marks']<90, ['ID', 'Name']])
```

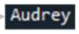


```
        ID  Name
2  21MS992  Alex
```

```
#Access single value by index
print(df.iat[1, 2])
```

```
90
```

```
#Access single value by label
print(df.at[3, 'Name'])
```

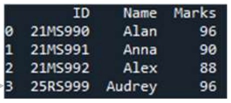


```
Audrey
```

Dr. Monidipa Das, Department of CDS, IISER Kolkata

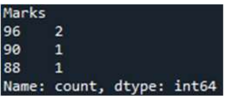
Pandas: Summarizing Data

```
data = {'ID': ['21MS990', '21MS991', '21MS992', '25RS999'],
        'Name': ['Alan', 'Anna', 'Alex', 'Audrey'],
        'Marks': [96, 90, 88, 96]}
df = pd.DataFrame(data,columns=list(data.keys()))
print(df)
```



```
        ID    Name  Marks
0  21MS990    Alan     96
1  21MS991    Anna     90
2  21MS992    Alex     88
3  25RS999  Audrey     96
```

```
#Count number of rows with each unique value of variable
df['Marks'].value_counts()
```

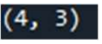


```
Marks
96    2
90    1
88    1
Name: count, dtype: int64
```

```
print(len(df)) # Number of rows in DataFrame
```

```
4
```

```
#Tuple of number of rows, number of columns in DataFrame
print(df.shape)
```

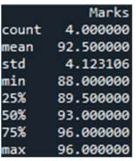


```
(4, 3)
```

```
print(df['Marks'].nunique()) # Number of distinct values in a column
```

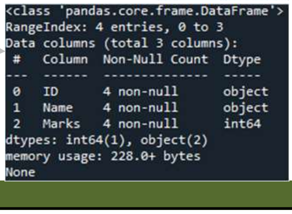
```
3
```

```
#Basic descriptive and statistics for each column
print(df.describe())
```



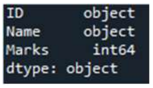
```
           Marks
count   4.000000
mean   92.500000
std     4.123106
min    88.000000
25%    89.500000
50%    93.000000
75%    96.000000
max    96.000000
```

```
print(df.info()) #Prints a concise summary of the DataFrame
```



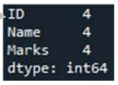
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   ID      4 non-null      object
 1   Name    4 non-null      object
 2   Marks   4 non-null      int64 
dtypes: int64(1), object(2)
memory usage: 228.0+ bytes
None
```

```
print(df.dtypes) #Prints a Series with the dtype of each column
```



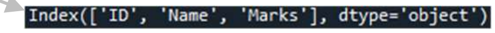
```
ID       object
Name     object
Marks     int64
dtype: object
```

```
print(df.count())#Number of non-NA values
```



```
ID       4
Name     4
Marks    4
dtype: int64
```

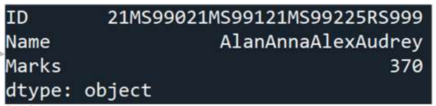
```
print(df.columns)#Describe DataFrame columns
```



```
Index(['ID', 'Name', 'Marks'], dtype='object')
```

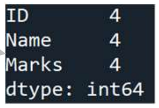
Pandas: Summarizing Data

```
print(df.sum()) #Sum values of each object
```



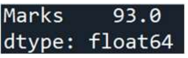
```
ID       21MS99021MS99121MS99225RS999
Name                  AlanAnnaAlexAudrey
Marks                                370
dtype: object
```

```
print(df.count())   #Count non-NA/null values of each object
```



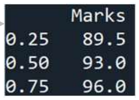
```
ID       4
Name     4
Marks    4
dtype: int64
```

```
print(df.median()) #Median value of each object
```



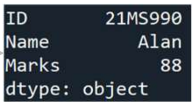
```
Marks     93.0
dtype: float64
```

```
print(df.quantile([0.25,0.5, 0.75])) #Quantiles of each object
```



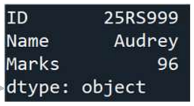
```
      Marks
0.25   89.5
0.50   93.0
0.75   96.0
```

```
print(df.min()) #Minimum value in each object
```



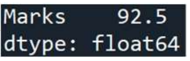
```
ID       21MS990
Name        Alan
Marks         88
dtype: object
```

```
print(df.max())   #Maximum value in each object
```



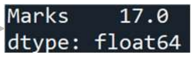
```
ID       25RS999
Name      Audrey
Marks         96
dtype: object
```

```
print(df.mean())  #Mean value of each object
```



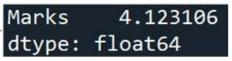
```
Marks     92.5
dtype: float64
```

```
print(df.var())   #Variance of each object
```

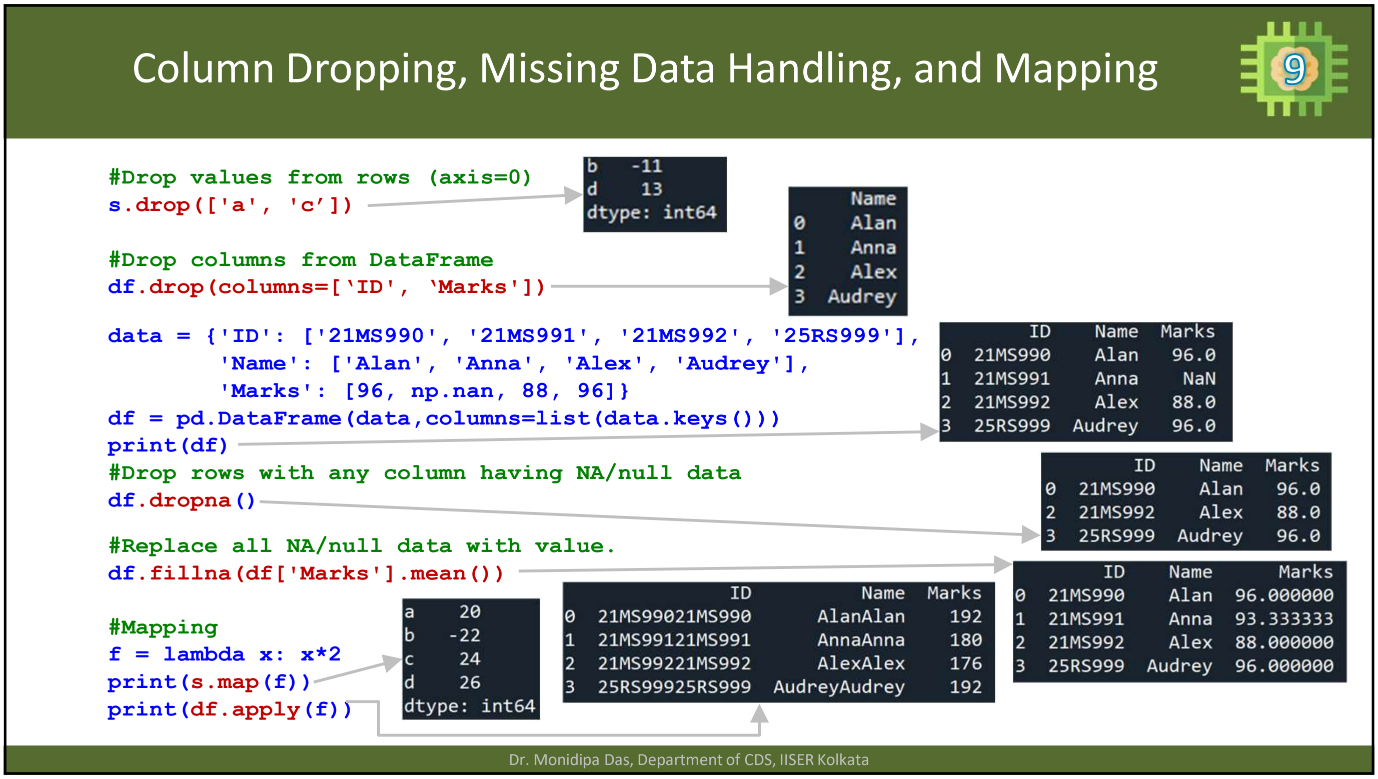


```
Marks     17.0
dtype: float64
```

```
print(df.std())   #Standard deviation of each object
```



```
Marks     4.123106
dtype: float64
```



Column Dropping, Missing Data Handling, and Mapping

9

```
#Drop values from rows (axis=0)
s.drop(['a', 'c’])
```

```
b    -11
d     13
dtype: int64
```

```
#Drop columns from DataFrame
df.drop(columns=[‘ID', ‘Marks'])
```

```
     Name
0    Alan
1    Anna
2    Alex
3  Audrey
```

```
data = {'ID': ['21MS990', '21MS991', '21MS992', '25RS999'],
        'Name': ['Alan', 'Anna', 'Alex', 'Audrey'],
        'Marks': [96, np.nan, 88, 96]}
df = pd.DataFrame(data,columns=list(data.keys()))
print(df)
```

```
        ID    Name  Marks
0  21MS990    Alan   96.0
1  21MS991    Anna    NaN
2  21MS992    Alex   88.0
3  25RS999  Audrey   96.0
```

```
#Drop rows with any column having NA/null data
df.dropna()
```

```
        ID    Name  Marks
0  21MS990    Alan   96.0
2  21MS992    Alex   88.0
3  25RS999  Audrey   96.0
```

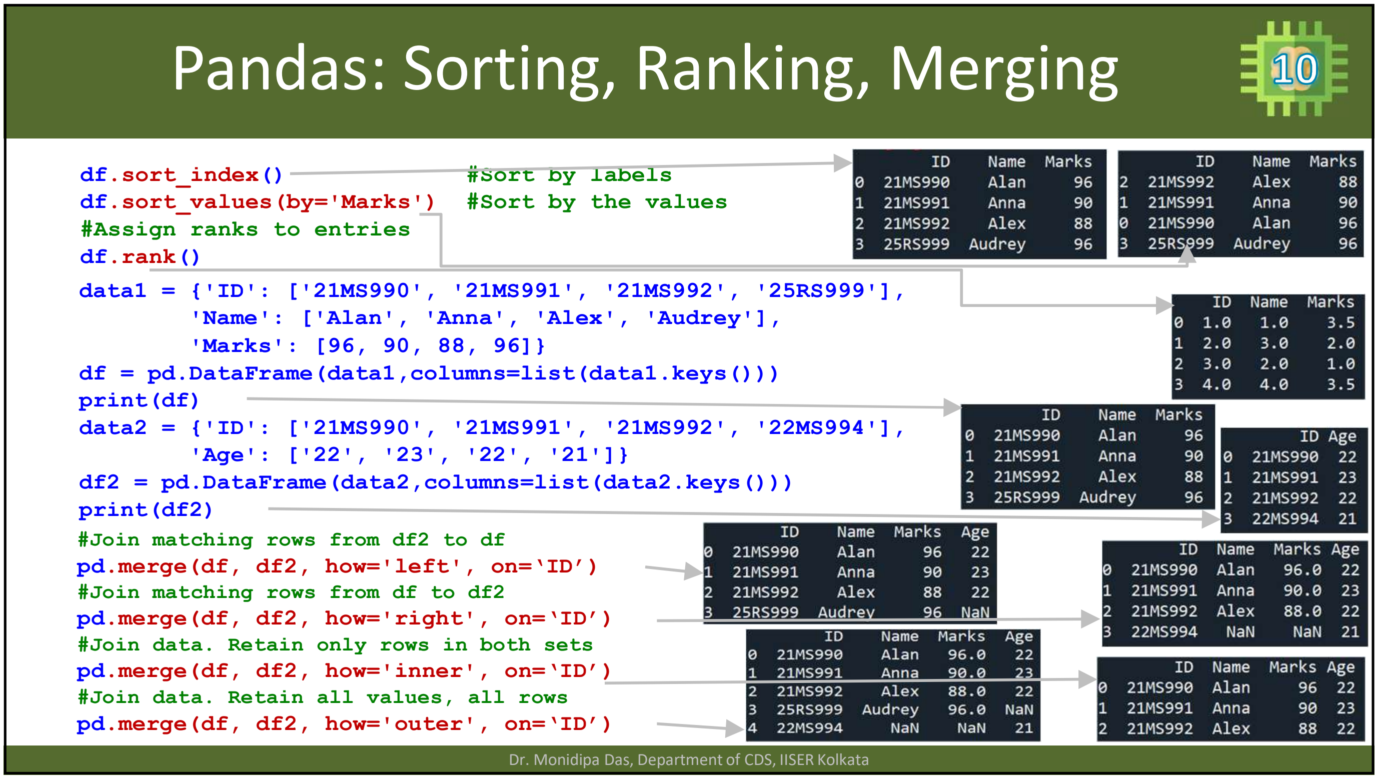
```
#Replace all NA/null data with value.
df.fillna(df['Marks'].mean())
```

```
        ID    Name      Marks
0  21MS990    Alan  96.000000
1  21MS991    Anna  93.333333
2  21MS992    Alex  88.000000
3  25RS999  Audrey  96.000000
```

```
#Mapping
f = lambda x: x*2
print(s.map(f))
print(df.apply(f))
```

```
a    20
b   -22
c    24
d    26
dtype: int64
```

```
                ID          Name  Marks
0  21MS99021MS990      AlanAlan    192
1  21MS99121MS991      AnnaAnna    180
2  21MS99221MS992      AlexAlex    176
3  25RS99925RS999  AudreyAudrey    192
```



Pandas: Sorting, Ranking, Merging

10

```
df.sort_index()               #Sort by labels
df.sort_values(by='Marks')    #Sort by the values
#Assign ranks to entries
df.rank()
```

```
        ID    Name  Marks
0  21MS990    Alan     96
1  21MS991    Anna     90
2  21MS992    Alex     88
3  25RS999  Audrey     96
```

```
        ID    Name  Marks
2  21MS992    Alex     88
1  21MS991    Anna     90
0  21MS990    Alan     96
3  25RS999  Audrey     96
```

```
    ID  Name  Marks
0  1.0   1.0    3.5
1  2.0   3.0    2.0
2  3.0   2.0    1.0
3  4.0   4.0    3.5
```

```
data1 = {'ID': ['21MS990', '21MS991', '21MS992', '25RS999'],
         'Name': ['Alan', 'Anna', 'Alex', 'Audrey'],
         'Marks': [96, 90, 88, 96]}
df = pd.DataFrame(data1,columns=list(data1.keys()))
print(df)
```

```
        ID    Name  Marks
0  21MS990    Alan     96
1  21MS991    Anna     90
2  21MS992    Alex     88
3  25RS999  Audrey     96
```

```
data2 = {'ID': ['21MS990', '21MS991', '21MS992', '22MS994'],
         'Age': ['22', '23', '22', '21']}
df2 = pd.DataFrame(data2,columns=list(data2.keys()))
print(df2)
```

```
        ID Age
0  21MS990  22
1  21MS991  23
2  21MS992  22
3  22MS994  21
```

```
#Join matching rows from df2 to df
pd.merge(df, df2, how='left', on='ID’)
```

```
        ID    Name  Marks  Age
0  21MS990    Alan     96   22
1  21MS991    Anna     90   23
2  21MS992    Alex     88   22
3  25RS999  Audrey     96  NaN
```

```
#Join matching rows from df to df2
pd.merge(df, df2, how='right', on=‘ID’)
```

```
        ID  Name  Marks Age
0  21MS990  Alan   96.0  22
1  21MS991  Anna   90.0  23
2  21MS992  Alex   88.0  22
3  22MS994   NaN    NaN  21
```

```
#Join data. Retain only rows in both sets
pd.merge(df, df2, how='inner', on=‘ID’)
```

```
        ID  Name  Marks Age
0  21MS990  Alan     96  22
1  21MS991  Anna     90  23
2  21MS992  Alex     88  22
```

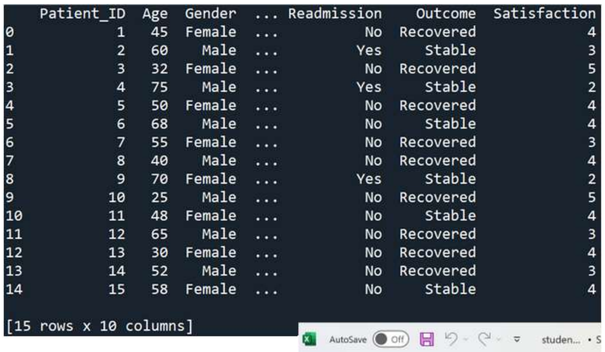
```
#Join data. Retain all values, all rows
pd.merge(df, df2, how='outer', on=‘ID’)
```

```
        ID    Name  Marks  Age
0  21MS990    Alan   96.0   22
1  21MS991    Anna   90.0   23
2  21MS992    Alex   88.0   22
3  25RS999  Audrey   96.0  NaN
4  22MS994     NaN    NaN   21
```

Pandas: Reading from and Writing to Files

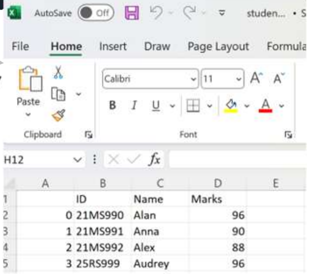
11

```
#Pandas: I/O
df1=pd.read_csv(‘F:/CS4103/code/hospitaldata.csv',
header=True, nrows=15)
print(df1)

data = {'ID': ['21MS990', '21MS991', '21MS992',
'25RS999'],
        'Name': ['Alan', 'Anna', 'Alex', 'Audrey'],
        ‘Age’: [22, 23, 22, 26],
        'Marks': [96, 90, 88, 96]}
df = pd.DataFrame(data1,columns=list(data1.keys()))
df.to_csv(‘F:/CS4103/code/student.csv’)

df2=pd.read_excel(‘F:/CS4103/code/rainfall.xlsx’)
print(df2.columns)
print(len(df2))
print(df2.head())

df.to_excel(‘F:/CS4103/student.xlsx',
sheet_name='Sheet1')
```

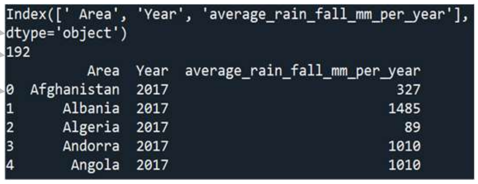


```
    Patient_ID  Age  Gender  ... Readmission   Outcome  Satisfaction
0            1   45  Female  ...          No  Recovered            4
1            2   60    Male  ...         Yes     Stable            3
2            3   32  Female  ...          No  Recovered            5
3            4   75    Male  ...         Yes     Stable            2
4            5   50  Female  ...          No  Recovered            4
5            6   68    Male  ...          No     Stable            4
6            7   55  Female  ...          No  Recovered            3
7            8   40    Male  ...          No  Recovered            4
8            9   70  Female  ...         Yes     Stable            2
9           10   25    Male  ...          No  Recovered            5
10          11   48  Female  ...          No     Stable            4
11          12   65    Male  ...          No  Recovered            3
12          13   30  Female  ...          No  Recovered            4
13          14   52    Male  ...          No  Recovered            3
14          15   58  Female  ...          No     Stable            4

[15 rows x 10 columns]
```

student.csv



	A	B	C	D	E
1		ID	Name	Marks	
2	0	21MS990	Alan	96	
3	1	21MS991	Anna	90	
4	2	21MS992	Alex	88	
5	3	25RS999	Audrey	96	



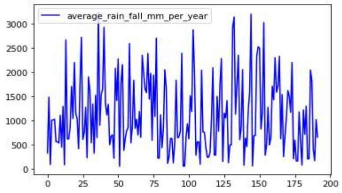
```
Index([' Area', 'Year', 'average_rain_fall_mm_per_year'],
dtype='object')
192
          Area  Year  average_rain_fall_mm_per_year
0  Afghanistan  2017                            327
1      Albania  2017                           1485
2      Algeria  2017                             89
3      Andorra  2017                           1010
4       Angola  2017                           1010
```

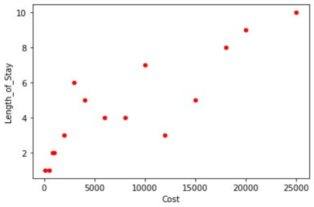
student.xlsx

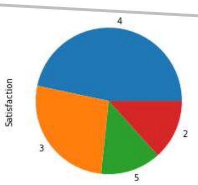
Pandas: Plotting

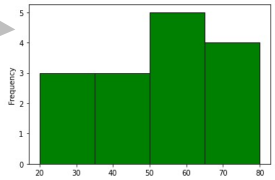
12

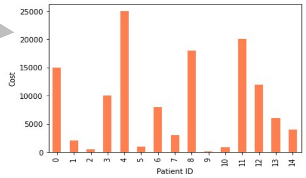
```
#Plot a line graph for the DataFrame
df2.iloc[:,[2]].plot(color=‘b’)

#Plot a scatter graph of the DataFrame
df1.plot.scatter(x='Cost',y='Length_of_Stay’,
color='r’)

#Plot a histogram of the DataFrame
cbins = np.linspace(20, 80, 5)
df1['Age'].plot.hist(bins=cbins,edgecolor='k’,
color='green’)

#Plot a pie chart of the DataFrame
df1['Satisfaction'].value_counts().plot.pie()
#Plot a bar chart of the DataFrame
ax=df1['Cost'].plot.bar(color='coral')
ax.set_xlabel('Patient ID')
ax.set_ylabel('Cost’)

#Area plot
ax=df2['average_rain_fall_mm_per_year'].plot.
area(color='b',alpha=0.4)
ax.set_ylabel('Rainfall')
```

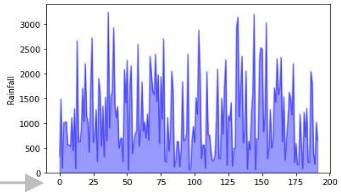








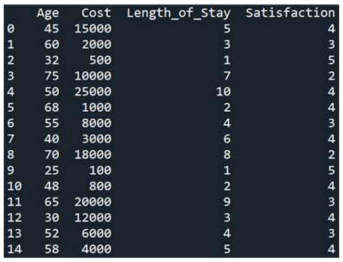




Pandas: Plotting [contd.]

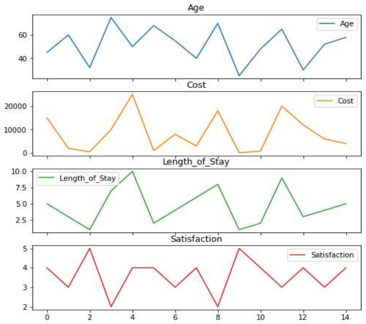
13

```
df1_plot=df1[['Age','Cost',
'Length_of_Stay', 'Satisfaction’]]
```

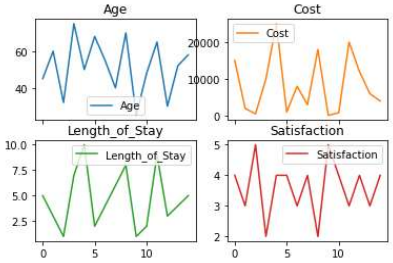


```
    Age   Cost  Length_of_Stay  Satisfaction
0    45  15000               5             4
1    60   2000               3             3
2    32    500               1             5
3    75  10000               7             2
4    50  25000              10             4
5    68   1000               2             4
6    55   8000               4             3
7    40   3000               6             4
8    70  18000               8             2
9    25    100               1             5
10   48    800               2             4
11   65  20000               9             3
12   30  12000               3             4
13   52   6000               4             3
14   58   4000               5             4
```

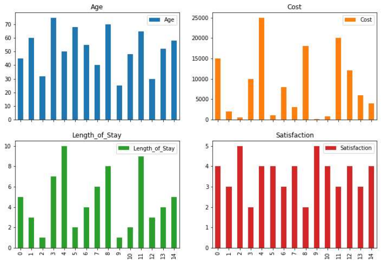
```
df1_plot.plot(subplots=True,
figsize=(8,8), title=['Age','Cost',
'Length_of_Stay', 'Satisfaction’])
```



```
df1_plot.plot(subplots=True, layout=(2,2),
title=['Age','Cost', 'Length_of_Stay',
'Satisfaction’])
```



```
df1_plot.plot.bar(subplots=True,
layout=(2,2),
figsize=(12,8),title=['Age','Cost',
'Length_of_Stay', 'Satisfaction'])
```



14

Basic Graph Manipulation

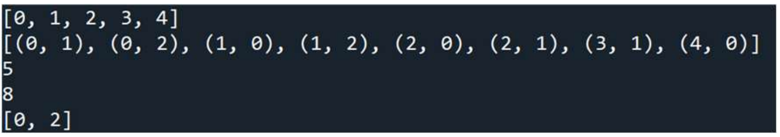
15

```
import networkx as nx

G1=nx.Graph() #Create an empty Undirected graph with no nodes and no edges

G1.add_node(1) #add one node at a time
G1.add_nodes_from([2,3,4]) #add a list of nodes
G1.add_edge(2,3) #add one edge at a time
G1.add_edges_from([(1,2),(1,3),(2,4),(1,4)])

G2=nx.DiGraph() #Create an empty Directed graph with no nodes and no edges

G2.add_node(1) #add one node at a time
G2.add_nodes_from([2,3,4]) #add a list of nodes
G2.add_edge(2,3) #add one edge at a time
G2.add_edges_from([(1,2),(1,3),(2,4),(1,4)])

#Create Undirected Graph by mapping nodes to neighbors
adjacency_dict = {0:(1,2), 1:(0,2,4), 2:(0,1,3,4), 3:(1,), 4:(0,)}
G3 = nx.Graph(adjacency_dict)

#Create Directed Graph dict mapping nodes to neighbors
adjacency_dict = {0: (1, 2), 1: (0, 2), 2: (0, 1), 3: (1,), 4:(0,)}
G4 = nx.DiGraph(adjacency_dict)
```

Basic Graph Manipulation [contd.]

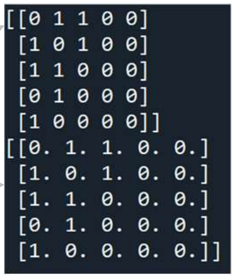
16

```
print(G4.nodes())
print(G4.edges())
print(G4.number_of_nodes())
print(G4.number_of_edges())
```

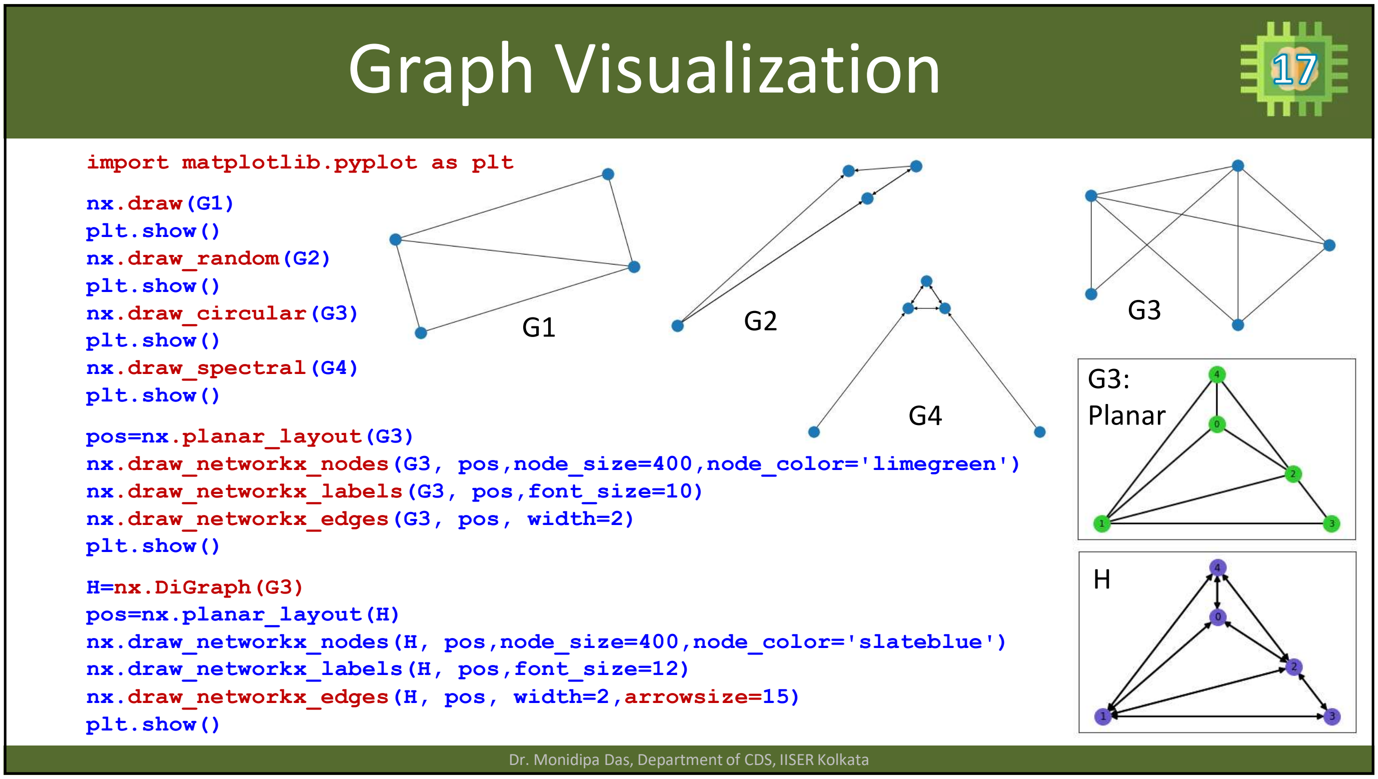


```
[0, 1, 2, 3, 4]
[(0, 1), (0, 2), (1, 0), (1, 2), (2, 0), (2, 1), (3, 1), (4, 0)]
5
8
[0, 2]
```

```
print([i for i in G4.neighbors(1)])
print(nx.adjacency_matrix(G4).toarray())
adjacency_matrix = nx.to_numpy_array(G4)
print(adjacency_matrix)
```

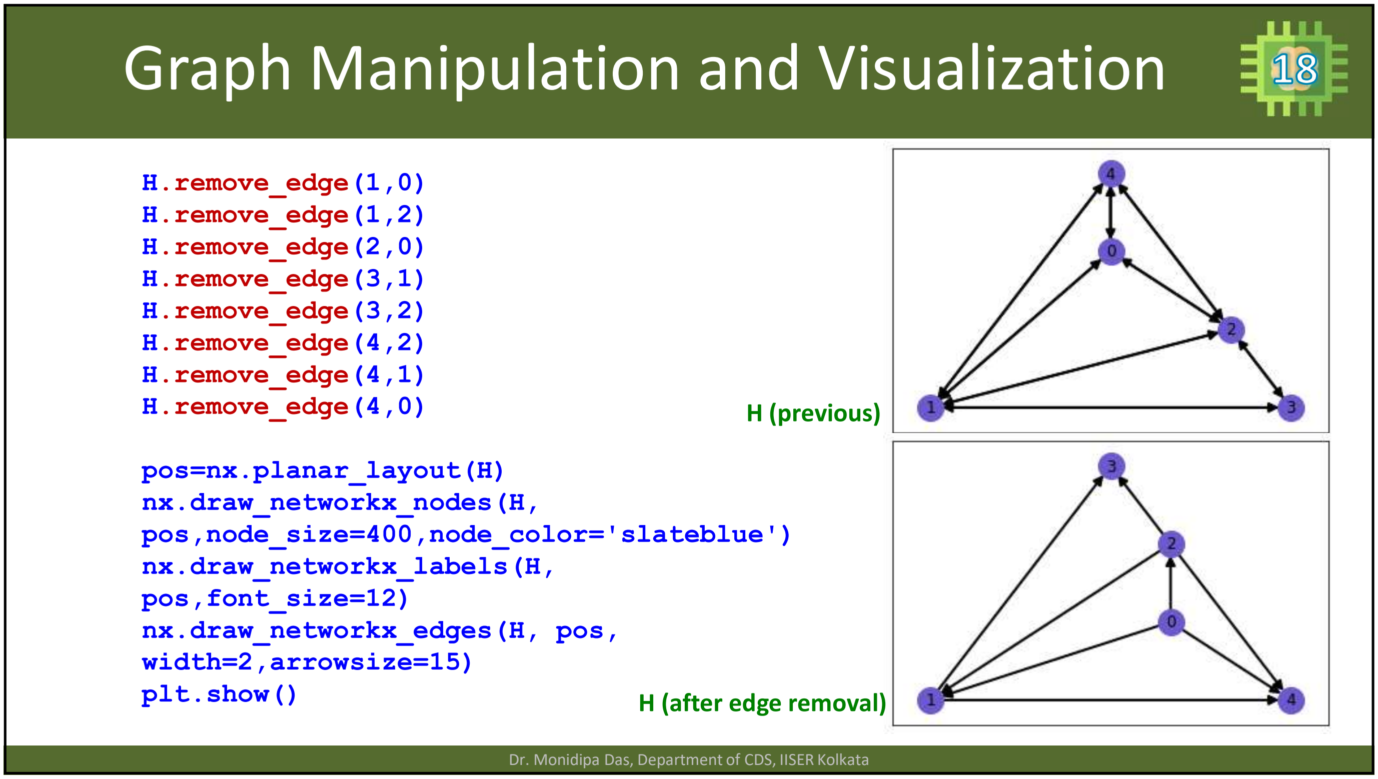


```
[[0 1 1 0 0]
 [1 0 1 0 0]
 [1 1 0 0 0]
 [0 1 0 0 0]
 [1 0 0 0 0]]
[[0. 1. 1. 0. 0.]
 [1. 0. 1. 0. 0.]
 [1. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [1. 0. 0. 0. 0.]]
```



Graph Visualization

```
import matplotlib.pyplot as plt

nx.draw(G1)
plt.show()
nx.draw_random(G2)
plt.show()
nx.draw_circular(G3)
plt.show()
nx.draw_spectral(G4)
plt.show()

pos=nx.planar_layout(G3)
nx.draw_networkx_nodes(G3, pos,node_size=400,node_color='limegreen')
nx.draw_networkx_labels(G3, pos,font_size=10)
nx.draw_networkx_edges(G3, pos, width=2)
plt.show()

H=nx.DiGraph(G3)
pos=nx.planar_layout(H)
nx.draw_networkx_nodes(H, pos,node_size=400,node_color='slateblue')
nx.draw_networkx_labels(H, pos,font_size=12)
nx.draw_networkx_edges(H, pos, width=2,arrowsize=15)
plt.show()
```



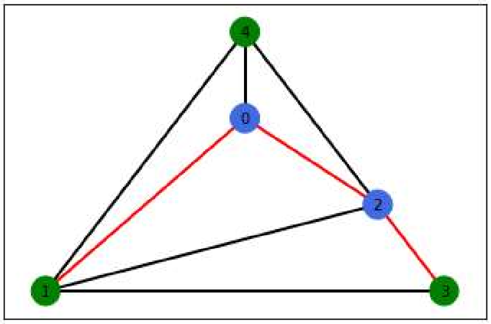
Graph Manipulation and Visualization

```
H.remove_edge(1,0)
H.remove_edge(1,2)
H.remove_edge(2,0)
H.remove_edge(3,1)
H.remove_edge(3,2)
H.remove_edge(4,2)
H.remove_edge(4,1)
H.remove_edge(4,0)

pos=nx.planar_layout(H)
nx.draw_networkx_nodes(H,
pos,node_size=400,node_color='slateblue')
nx.draw_networkx_labels(H,
pos,font_size=12)
nx.draw_networkx_edges(H, pos,
width=2,arrowsize=15)
plt.show()
```

Adding Attributes to Nodes and Edges

19

```
for i in range(5):
    if i in [0,2]:
        G3.nodes[i]['color'] = 'royalblue'
    else:
        G3.nodes[i]['color'] = 'green'
for i in G3.edges():
    if i in [(0,1),(0,2),(2,3)]:
        G3.edges[i[0],i[1]]['color'] = 'red'
    else:
        G3.edges[i[0],i[1]]['color'] = 'black'
```



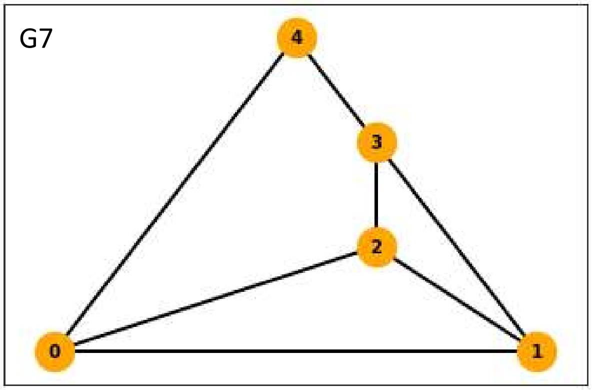
G3

```
node_colors = nx.get_node_attributes(G3, 'color').values()
edge_colors = nx.get_edge_attributes(G3, 'color').values()

pos=nx.planar_layout(G3)
nx.draw_networkx_nodes(G3, pos,node_size=400,node_color=node_colors)
nx.draw_networkx_labels(G3, pos,font_size=10)
nx.draw_networkx_edges(G3, pos, width=2, edge_color=edge_colors)
plt.show()
```

Creating Undirected Graph from Adjacency Matrix

20

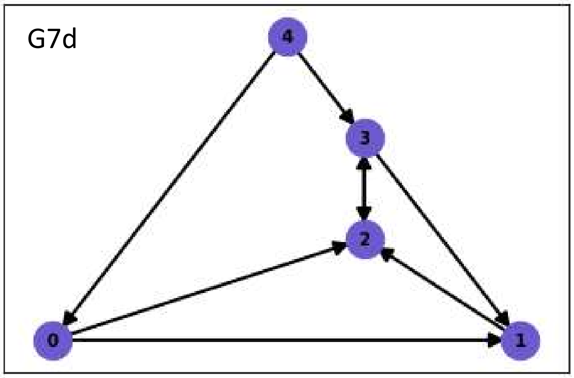
```
import numpy as np

a=np.array(
[[0,1,1,0,0],
 [0,0,1,0,0],
 [0,0,0,1,0],
 [0,1,1,0,0],
 [1,0,0,1,0]])

G7 = nx.from_numpy_array(a)
```



```
pos = nx.planar_layout(G7)
nx.draw_networkx_nodes(G7, pos,node_size=500,node_color='orange')
nx.draw_networkx_labels(G7, pos,font_size=10,font_weight='bold')
nx.draw_networkx_edges(G7, pos, width=2)
plt.show()
```

Creating Directed Graph from Adjacency Matrix

```
import numpy as np

a=np.array(
[[0,1,1,0,0],
 [0,0,1,0,0],
 [0,0,0,1,0],
 [0,1,1,0,0],
 [1,0,0,1,0]])

G7d = nx.from_numpy_array(a,
          create_using=nx.DiGraph())
```

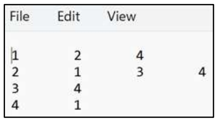


```
pos = nx.planar_layout(G7d)
nx.draw_networkx_nodes(G7d, pos,node_size=500,node_color='slateblue')
nx.draw_networkx_labels(G7d, pos,font_size=10,font_weight='bold')
nx.draw_networkx_edges(G7d, pos, width=2,arrowsize=18)
plt.show()
```

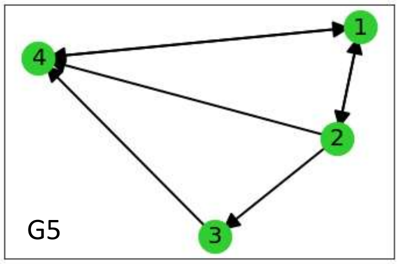
Creating Graphs from File Data

```
G5 =
nx.read_adjlist('adj.txt',create_using=nx.DiGraph())
pos = nx.spring_layout(G5)
nx.draw_networkx_nodes(G5,
pos,node_size=800,node_color='limegreen')
nx.draw_networkx_labels(G5, pos,font_size=20)
nx.draw_networkx_edges(G5, pos, width=2,arrowsize=30)
plt.show()
```

adj.txt

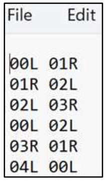


```
1    2    4
2    1    3    4
3    4
4    1
```

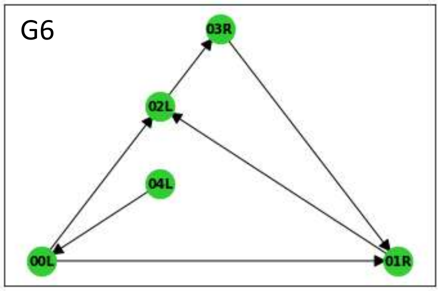


```
G6 =
nx.read_edgelist('edge.txt',create_using=nx.DiGraph())
pos = nx.planar_layout(G6)
nx.draw_networkx_nodes(G6,
pos,node_size=500,node_color='limegreen')
nx.draw_networkx_labels(G6,
pos,font_size=10,font_weight='bold')
nx.draw_networkx_edges(G6, pos,arrowsize=20)
plt.show()
```

edge.txt



```
00L 01R
01R 02L
02L 03R
00L 02L
03R 01R
04L 00L
```



Creating Graphs from File Data [contd.]

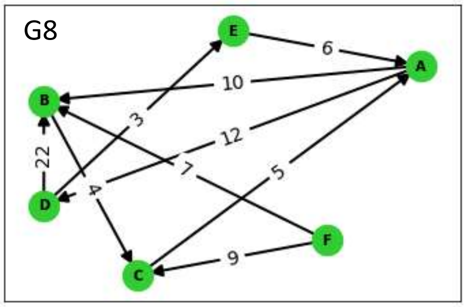
```
import pandas as pd
df=pd.read_excel('F:/CS4103/code/nodeposition.xlsx')
pos={}
for i in range(len(df)):
    pos[df['Node'][i]]=(df['X'][i],df['Y'][i])

df=pd.read_csv('F:/CS4103/code/graphdataset.csv')
G8=nx.from_pandas_edgelist(df, source='S', target='D',
edge_attr='W',create_using=nx.DiGraph())
edge_labels = {}
for u, v, d in G8.edges(data=True):
    edge_labels[(u, v)] = d['W’]

nx.draw_networkx_nodes(G8,
pos,node_size=500,node_color='limegreen')
nx.draw_networkx_labels(G8,
pos,font_size=10,font_weight='bold')
nx.draw_networkx_edges(G8, pos, width=2,arrowsize=18)
nx.draw_networkx_edge_labels(G8, pos,
edge_labels=edge_labels, font_size=14, font_color='k')
plt.show()
```

Node	X	Y
A	5	10
B	1	9
C	2	4
D	1	6
E	3	11
F	4	5

nodeposition.xlsx

S	D	W
A	B	10
A	D	12
B	C	4
C	A	5
D	E	3
E	A	6
F	B	7
F	C	9
D	B	22

graphdataset.csv



Creating Graphs from File Data [contd.]

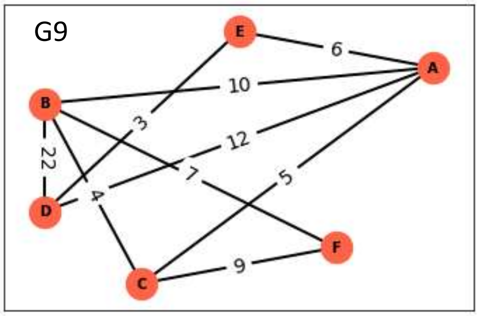
```
import pandas as pd
df=pd.read_excel('F:/CS4103/code/nodeposition.xlsx')
pos={}
for i in range(len(df)):
    pos[df['Node'][i]]=(df['X'][i],df['Y'][i])

df=pd.read_csv('F:/CS4103/code/graphdataset.csv')
G9=nx.from_pandas_edgelist(df, source='S', target='D',
edge_attr='W',create_using=nx.Graph())
edge_labels = {}
for u, v, d in G9.edges(data=True):
    edge_labels[(u, v)] = d['W’]

nx.draw_networkx_nodes(G9,
pos,node_size=500,node_color='tomato')
nx.draw_networkx_labels(G9,
pos,font_size=10,font_weight='bold')
nx.draw_networkx_edges(G9, pos, width=2,arrowsize=18)
nx.draw_networkx_edge_labels(G9, pos,
edge_labels=edge_labels, font_size=14, font_color='k')
plt.show()
```

Node	X	Y
A	5	10
B	1	9
C	2	4
D	1	6
E	3	11
F	4	5

nodeposition.xlsx

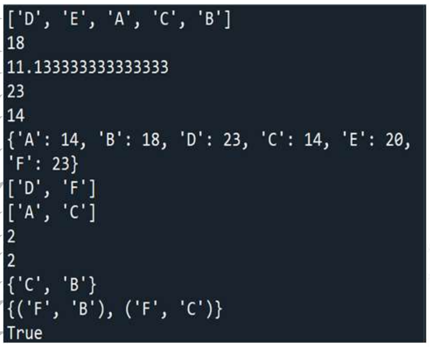
S	D	W
A	B	10
A	D	12
B	C	4
C	A	5
D	E	3
E	A	6
F	B	7
F	C	9
D	B	22

graphdataset.csv

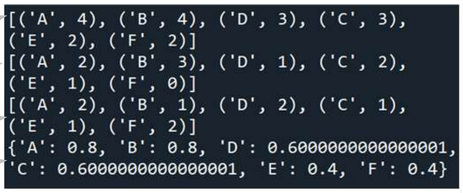


A Few Network Structure and Analysis Measures

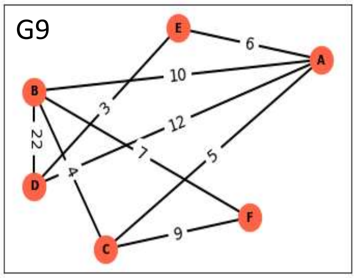
```
print(nx.shortest_path(G9,'D','B',
weight='W'))
print(nx.shortest_path_length(G9,'D',
'B',weight='W'))
print(nx.average_shortest_path_length
(G9,weight='W'))
print(nx.diameter(G9,weight='W'))
print(nx.radius(G9,weight='W'))
print(nx.eccentricity(G9,weight='W'))
print(nx.periphery(G9,weight='W'))
print(nx.center(G9,weight='W'))
print(nx.node_connectivity(G9))
print(nx.edge_connectivity(G9))
print(nx.minimum_node_cut(G9))
print(nx.minimum_edge_cut(G9))
print(nx.is_connected(G9))

print(G8.degree())
print(G8.in_degree())
print(G8.out_degree())
print(nx.degree_centrality(G8))
```

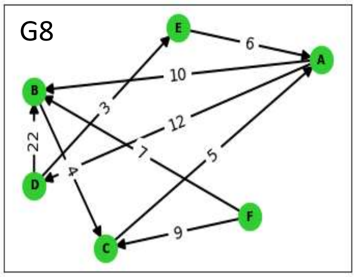


```
['D', 'E', 'A', 'C', 'B']
18
11.133333333333333
23
14
{'A': 14, 'B': 18, 'D': 23, 'C': 14, 'E': 20,
'F': 23}
['D', 'F']
['A', 'C']
2
2
{'C', 'B'}
{('F', 'B'), ('F', 'C')}
True
```



```
[('A', 4), ('B', 4), ('D', 3), ('C', 3),
('E', 2), ('F', 2)]
[('A', 2), ('B', 3), ('D', 1), ('C', 2),
('E', 1), ('F', 0)]
[('A', 2), ('B', 1), ('D', 2), ('C', 1),
('E', 1), ('F', 2)]
{'A': 0.8, 'B': 0.8, 'D': 0.6000000000000001,
'C': 0.6000000000000001, 'E': 0.4, 'F': 0.4}
```





Questions?