

Edge Attributes in NetworkX

```
G=nx.Graph()
G.add_edge('A','B', weight= 6, relation = 'family')
G.add_edge('B','C', weight= 13, relation = 'friend')
```

In: `G.edges()` #list of all edges

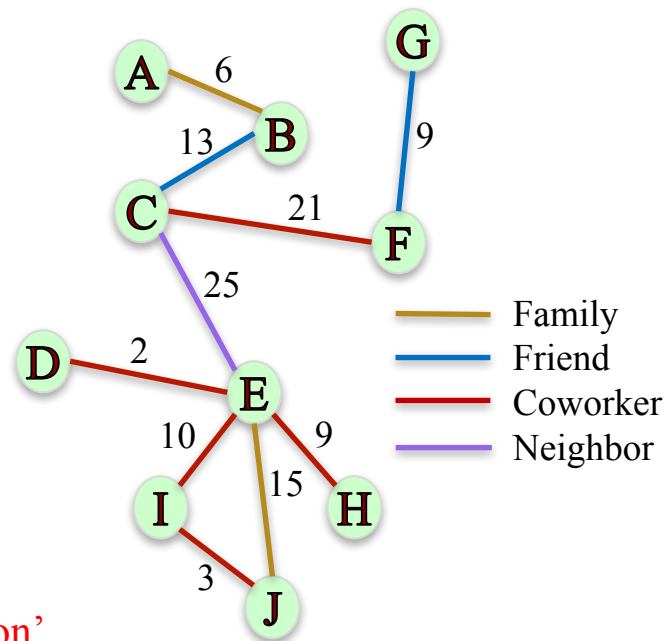
Out: `[('A', 'B'), ('C', 'B')]`

In: `G.edges(data= True)` #list of all edges with attributes

Out: `[('A', 'B', {'relation': 'family', 'weight': 6}),
('C', 'B', {'relation': 'friend', 'weight': 13})]`

In: `G.edges(data= 'relation')` #list of all edges with attribute 'relation'

Out: `[('A', 'B', 'family'), ('C', 'B', 'friend')]`



Number of times coworkers had
lunch together in one year

Edge Attributes in NetworkX

Accessing attributes of a specific edge:

In: `G.edge['A']['B']` # dictionary of attributes of edge (A, B)

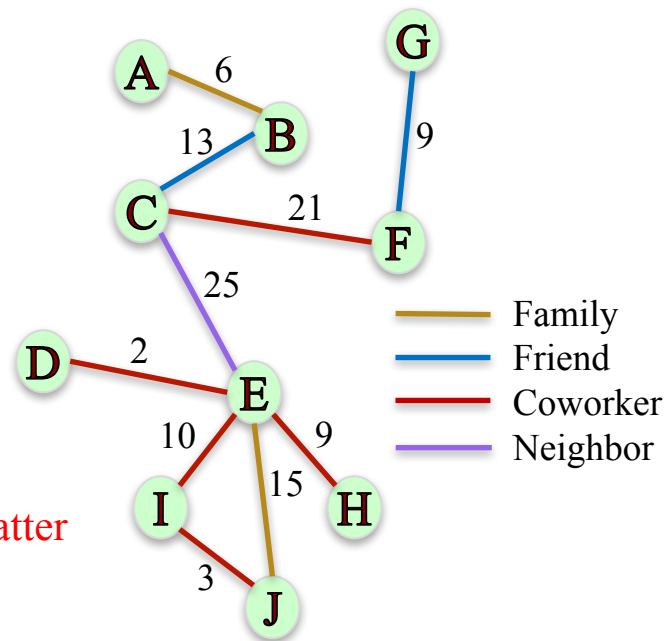
Out: `{'relation': 'family', 'weight': 6}`

In: `G.edge['B']['C']['weight']`

Out: 13

In: `G.edge['C']['B']['weight']` # undirected graph, order does not matter

Out: 13



Number of times coworkers had
lunch together in one year

Edge Attributes in NetworkX

Directed, weighted network:

```
G=nx.DiGraph()
```

```
G.add_edge('A','B', weight= 6, relation = 'family')
```

```
G.add_edge('C', 'B', weight= 13, relation = 'friend')
```

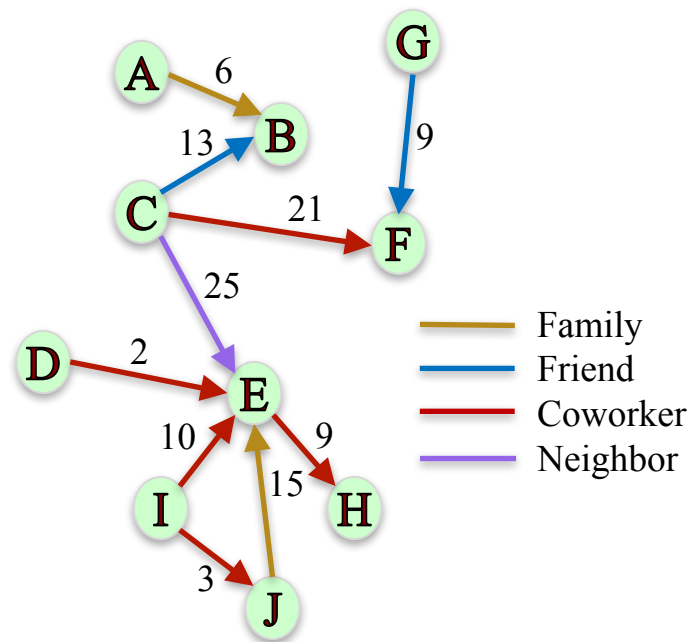
Accessing edge attributes:

```
In: G.edge['C']['B']['weight']
```

```
Out: 13
```

```
In: G.edge['B']['C']['weight'] # directed graph, order matters
```

```
Out: KeyError: 'C'
```



Edge Attributes in NetworkX

MultiGraph:

```
G=nx.MultiGraph()
```

```
G.add_edge('A','B', weight= 6, relation = 'family')
```

```
G.add_edge('A','B', weight= 18, relation = 'friend')
```

```
G.add_edge('C','B', weight= 13, relation = 'friend')
```

Accessing edge attributes:

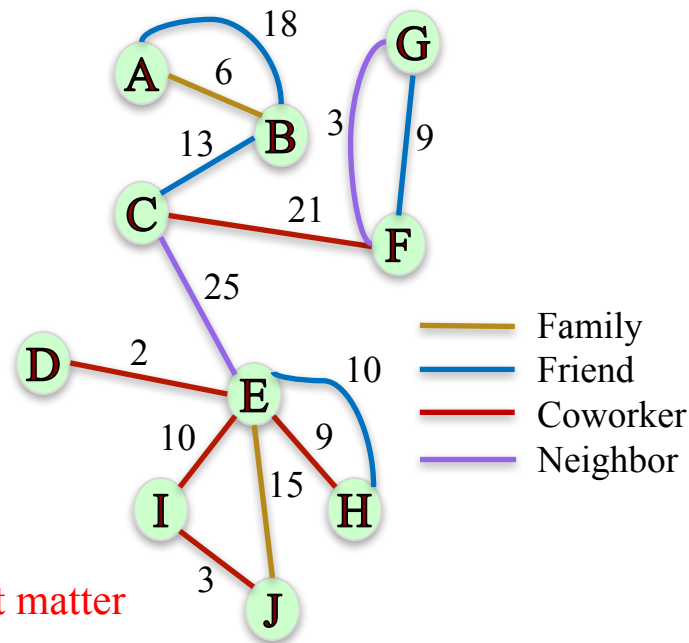
```
In: G.edge['A']['B'] # One dictionary of attributes per (A,B) edge
```

```
Out: {0: {'relation': 'family', 'weight': 6},
```

```
1: {'relation': 'friend', 'weight': 18}}
```

```
In: G.edge['A']['B'][0]['weight'] # undirected graph, order does not matter
```

```
Out: 6
```



Edge Attributes in NetworkX

Directed MultiGraph:

```
G=nx.MultiDiGraph()
```

```
G.add_edge('A','B', weight= 6, relation = 'family')
```

```
G.add_edge('A','B', weight= 18, relation = 'friend')
```

```
G.add_edge('C','B', weight= 13, relation = 'friend')
```

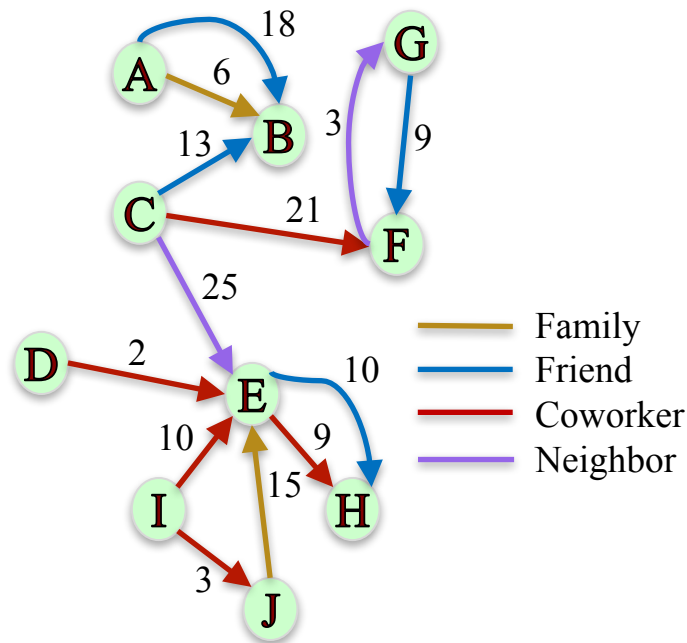
Accessing edge attributes:

```
In: G.edge['A']['B'][0]['weight']
```

```
Out: 6
```

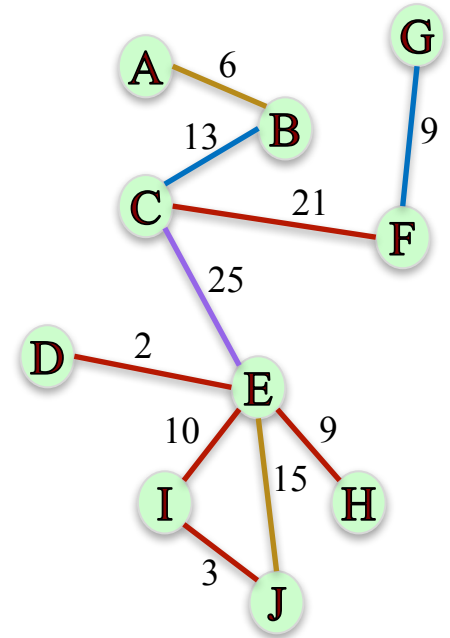
```
In: G.edge['B']['A'][0]['weight'] # directed graph, order matters
```

```
Out: KeyError: 'A'
```



Node Attributes in NetworkX

— Family
— Friend
— Coworker
— Neighbor



Number of times coworkers had lunch together in one year

Node Attributes in NetworkX

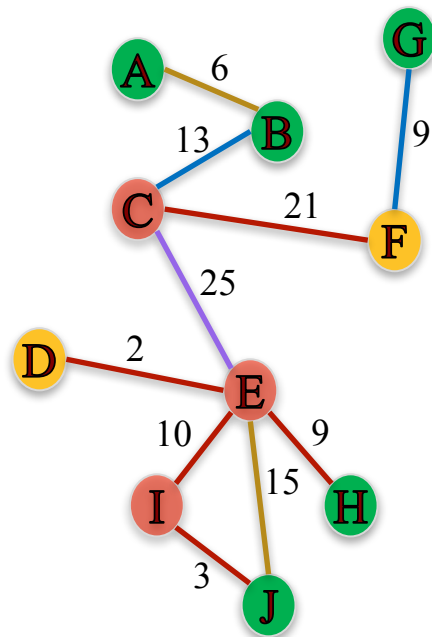
```
G=nx.Graph()
G.add_edge('A','B', weight= 6, relation = 'family')
G.add_edge('B','C', weight= 13, relation = 'friend')
```

Adding node attributes:

```
G.add_node('A', role = 'trader')
G.add_node('B', role = 'trader')
G.add_node('C', role = 'manager')
```

Family
Friend
Coworker
Neighbor

Manager
Trader
Analyst



Number of times coworkers had
lunch together in one year

Node Attributes in NetworkX

```
G=nx.Graph()
G.add_edge('A','B', weight= 6, relation = 'family')
G.add_edge('B','C', weight= 13, relation = 'friend')
```

Accessing node attributes:

```
In: G.nodes() # list of all nodes
```

```
Out: ['A', 'C', 'B']
```

```
In: G.nodes(data= True) #list of all nodes with attributes
```

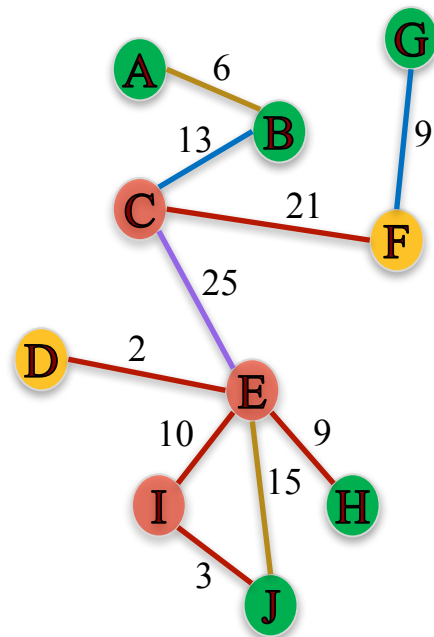
```
Out: [('A', {'role': 'trader'}), ('C', {'role': 'manager'})
, ('B', {'role': 'trader'})]
```

```
In: G.node['A']['role']
```

```
Out: 'manager'
```

Family
Friend
Coworker
Neighbor

Manager
Trader
Analyst



Number of times coworkers had lunch together in one year

Summary

Adding node and edge attributes:

```
G=nx.Graph()  
G.add_edge('A','B', weight= 6, relation = 'family')  
G.add_node('A', role = 'trader')
```

Accessing node attributes:

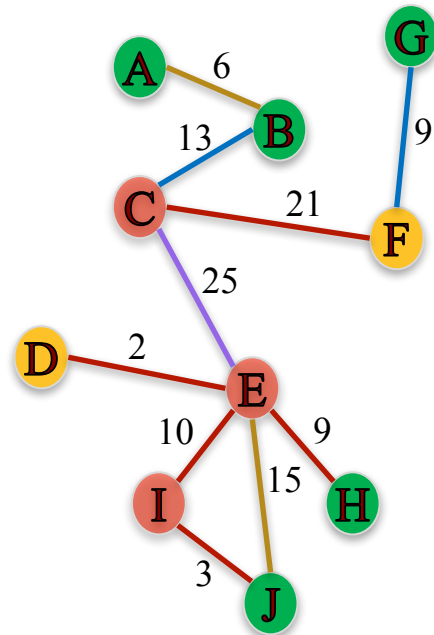
```
G.nodes(data= True) #list of all nodes with attributes  
G.node['A']['role'] #role of node A
```

Accessing Edge attributes:

```
In: G.edges(data= True) #list of all edges with attributes  
In: G.edges(data= 'relation') #list of all edges with attribute 'relation'  
G.edge['A']['B']['weight'] # weight of edge (A,B)
```

Family
Friend
Coworker
Neighbor

Manager
Trader
Analyst



Number of times coworkers had lunch together in one year