

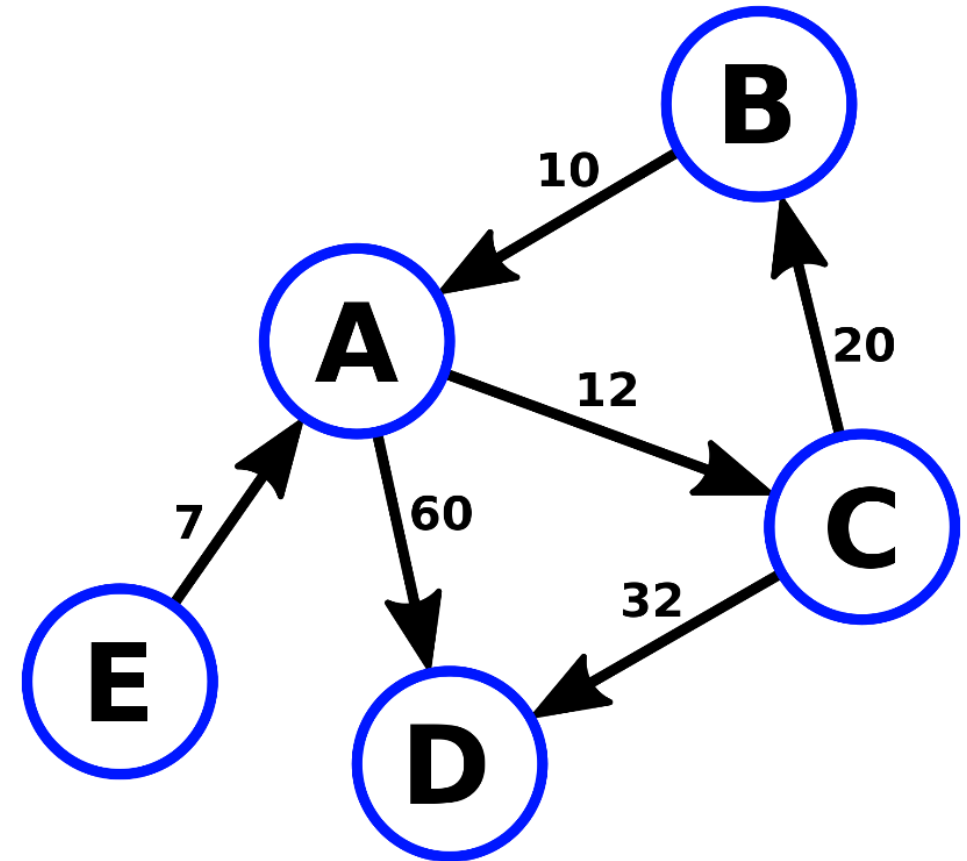
# Graphentheorie I

Kantonsschule Sursee

Dr. Philipp Hurni

# Agenda

- Was ist ein Graph?
- Darstellung eines Graphen
- Eigenschaften von Graphen
- Grad eines Knotens
- Übungen

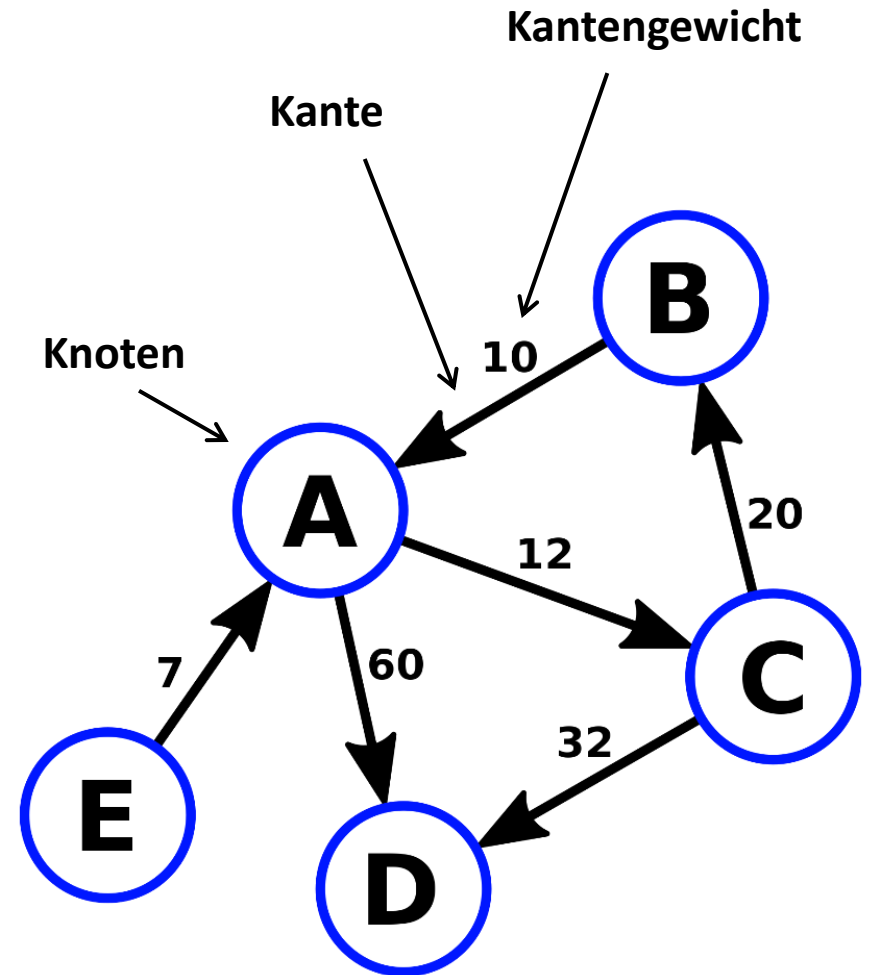


# Was ist ein Graph?

## Definition (Wikipedia):

Ein Graph ist eine abstrakte Struktur, die eine Menge von Objekten zusammen mit den zwischen diesen Objekten bestehenden Verbindungen repräsentiert.

- Die Objekte werden dabei **Knoten** des Graphen genannt.
- Die paarweisen Verbindungen zwischen Knoten heissen **Kanten**



# Darstellung eines Graphen – mit Mengen

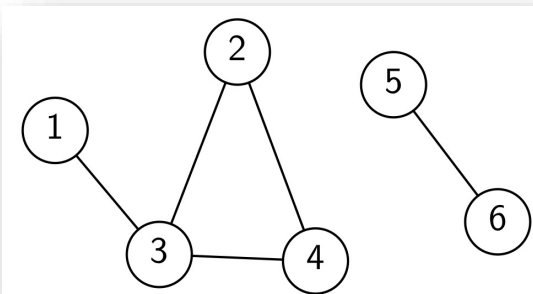
## Mathematisch formale Beschreibung:

### Ein Graph besteht aus:

- einer Menge von Knoten **V** (engl. Vertices)
- einer Menge von Kanten **E** (engl. Edges)

Anmerkung: hier ist der Graph ungerichtet – das heisst, eine Verbindung von 1 nach 3 bedeutet auch eine Verbindung von 3 nach 1. Das kann man auch anders definieren.

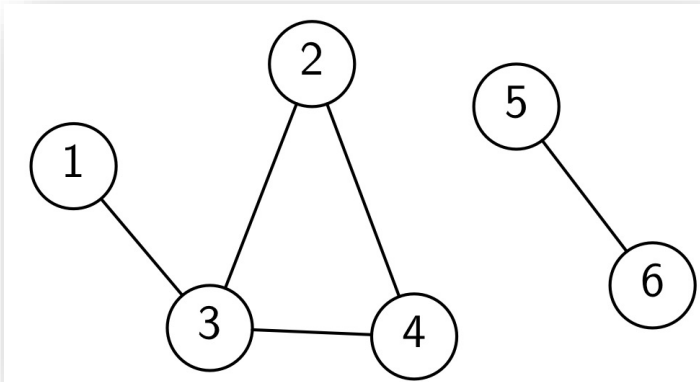
## Datenstruktur für einen Graphen



$$V = \{1,2,3,4,5,6\}$$

$$E = \{(1,3); (2,3); (2,4); (3,4); (5,6)\}$$

# Darstellung eines Graphen – mit Adjazenzliste/matrix



## Adjazenzliste

1: 3  
2: 3,4  
3: 1,2,4  
4: 2,3  
5: 6  
6: 5

2D-Liste. Für jeden Knoten ist gespeichert, zu welchem Knoten er eine Kante besitzt

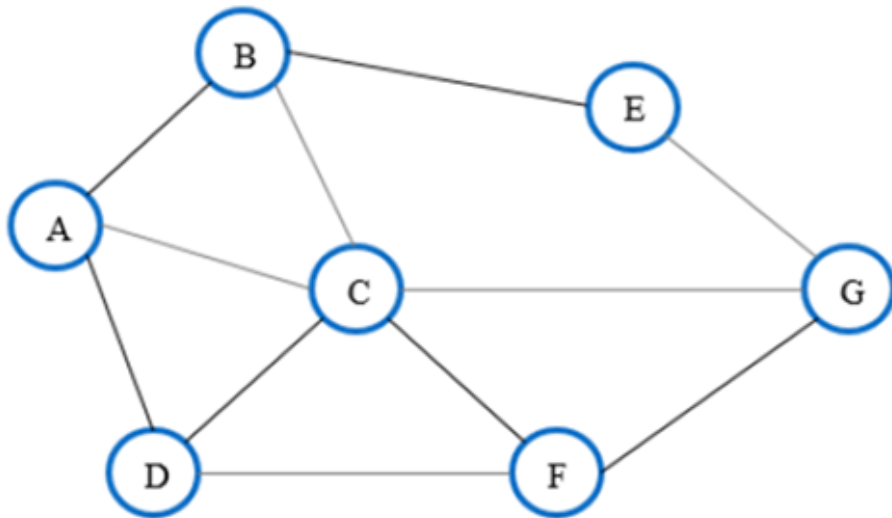
## Adjazenzmatrix

0	0	1	0	0	0
0	0	1	1	0	0
1	1	0	1	0	0
0	1	1	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0

n x n Matrix mit Werten falls Werte

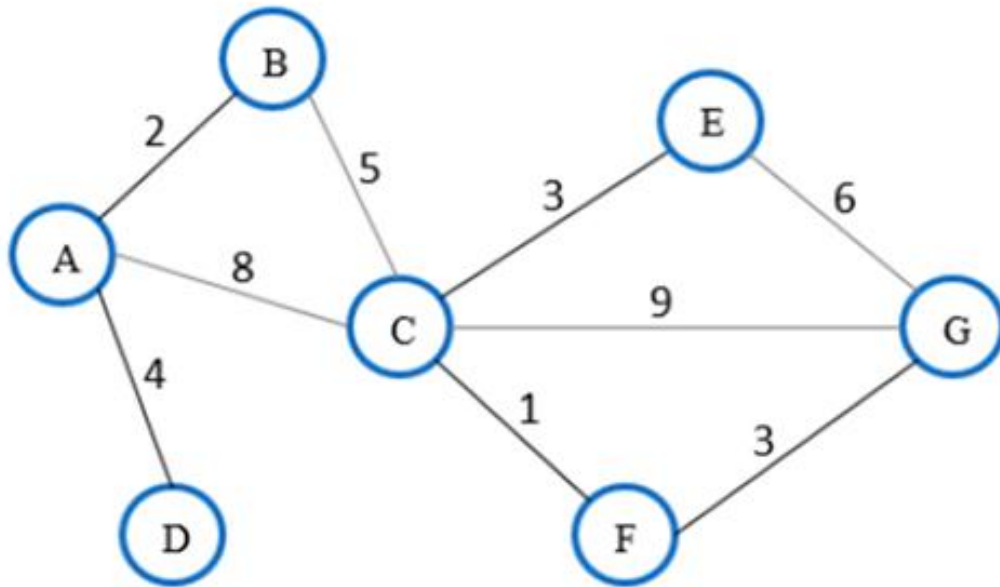
- >0 → es gibt eine Kante
- 0 → keine Kante)

# Aufgabe 1 – Schreibe die Adjazenzmatrix auf



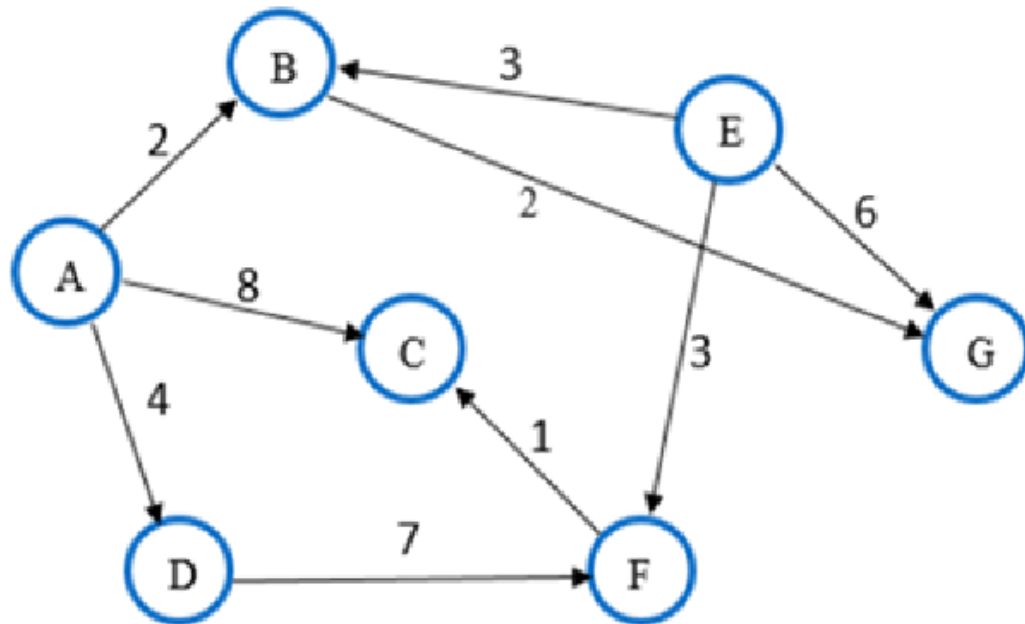
0111000  
1010100  
1111011  
1010010  
0100001  
0011001  
0010110

## Aufgabe 2 – Fülle die Adjazenzmatrix



000000C

# Aufgabe 3– Fülle die Adjazenzmatrix

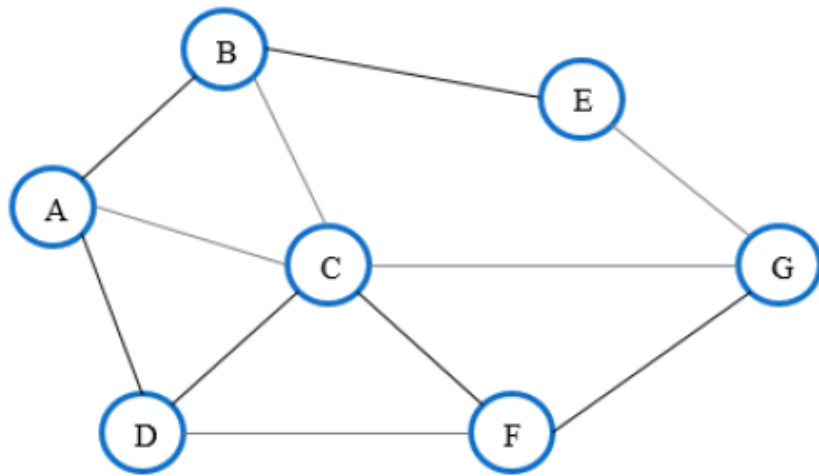


0284000  
0000002  
0000000  
0000070  
0300036  
0010000  
0000000

# Lösungen zu Darstellung Graphen



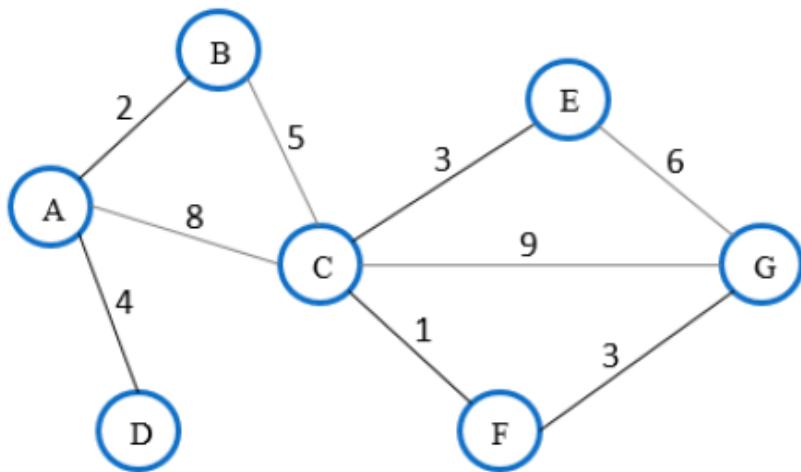
# Aufgabe 1 – Fülle die Adjazenzmatrix



Graph 1

	A	B	C	D	E	F	G
A	0	1	1	1	0	0	0
B	1	0	1	0	1	0	0
C	1	1	0	1	0	1	1
D	1	0	1	0	0	1	0
E	0	1	0	0	0	0	1
F	0	0	1	1	0	0	1
G	0	0	1	0	1	1	0

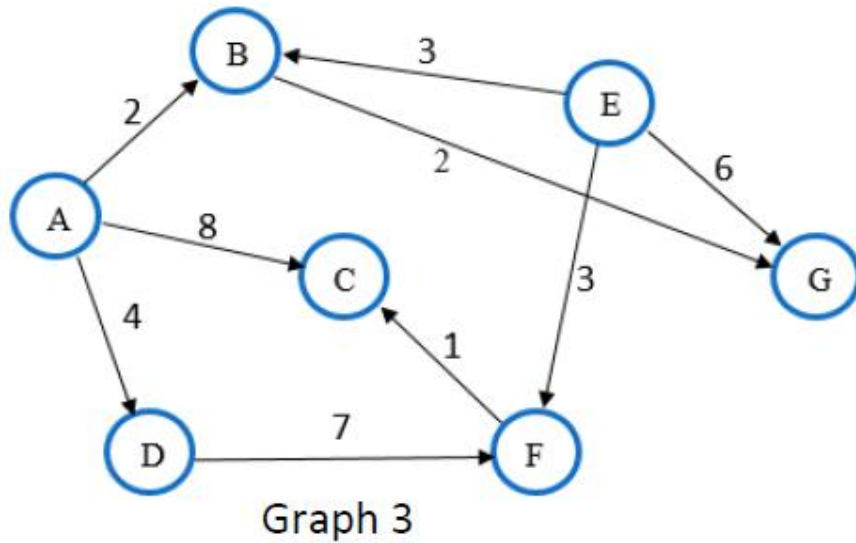
## Aufgabe 2 – Fülle die Adjazenzmatrix



Graph 2

	A	B	C	D	E	F	G
A	0	2	8	4	0	0	0
B	2	0	5	0	0	0	0
C	8	5	0	0	3	1	9
D	4	0	0	0	0	0	0
E	0	0	3	0	0	0	6
F	0	0	1	0	0	0	3
G	0	0	9	0	6	3	0

# Aufgabe 3– Fülle die Adjazenzmatrix



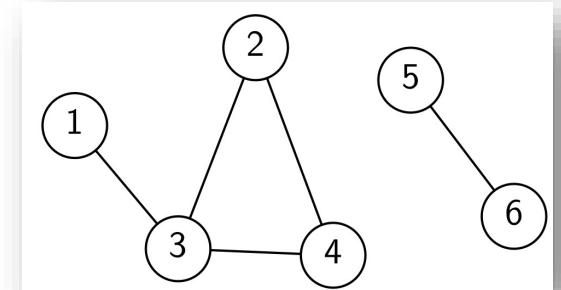
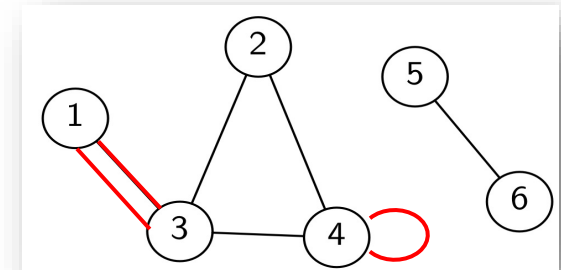
	A	B	C	D	E	F	G
A	0	2	8	4	0	0	0
B	0	0	0	0	0	0	2
C	0	0	0	0	0	0	0
D	0	0	0	0	0	7	0
E	0	3	0	0	0	3	6
F	0	0	1	0	0	0	0
G	0	0	0	0	0	0	0

# Eigenschaften von Graphen



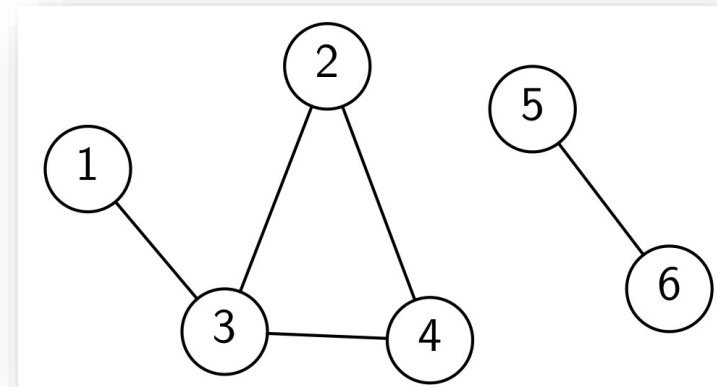
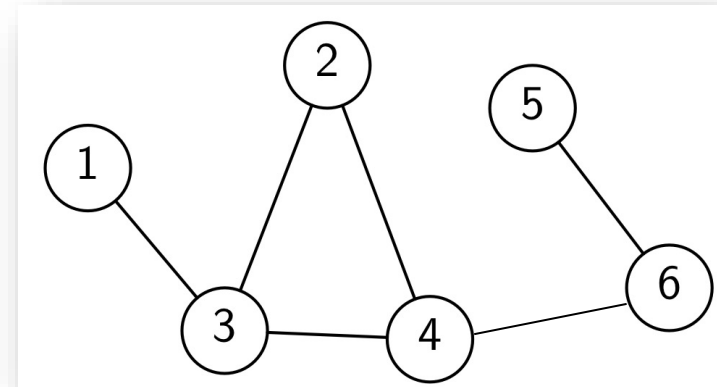
# Eigenschaften von Graphen

- Multigraph: Der Graph hat Schlingen (Kanten zu sich selbst) oder Mehrfachkanten
- Einfacher Graph: Keine Schlingen & Mehrfachkanten



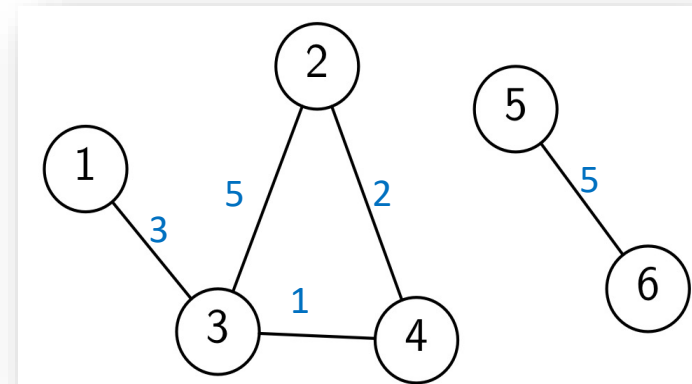
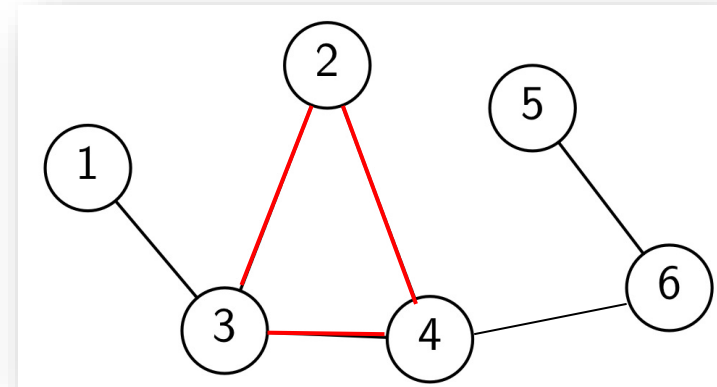
# Eigenschaften von Graphen

- Zusammenhängender Graph: Es gibt einen Pfad zwischen allen Paaren von Knoten
- Nicht zusammenhängender Graph: Es gibt nicht immer einen Pfad zwischen Paaren von Knoten



# Eigenschaften von Graphen

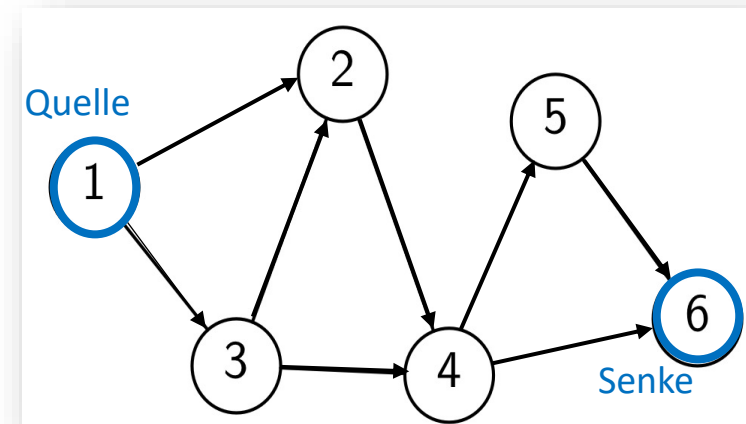
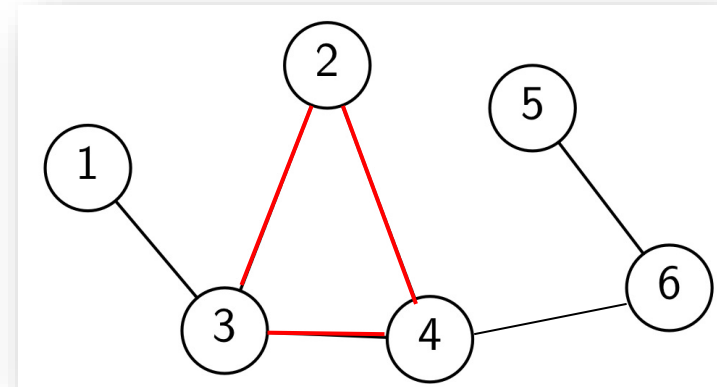
- Zyklisch – Azyklisch: Der Graph enthält einen / keinen Zyklus
- Gewichtet: Jede Kante hat ein Gewicht (Kosten)



# Eigenschaften von Graphen

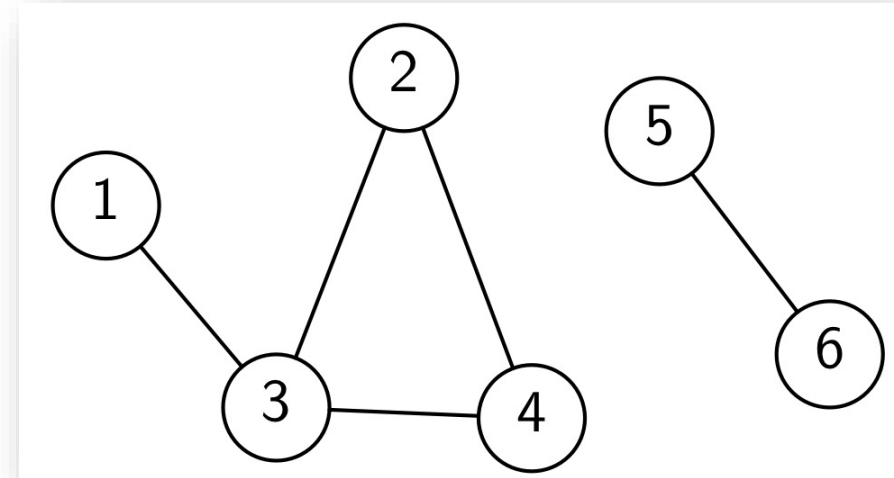
- Zyklisch – Azyklisch: Der Graph enthält einen / keinen Zyklus
- Gerichteter Graph: Kanten besitzen eine eindeutige Richtung

DAG (Directed Acyclic Graph) hat Quelle und Senke



# Der Grad eines Knotens

- Anzahl Kanten, die zu ihm führen
  - $\deg(3)=3$
  - $\deg(4)=2$
  - $\deg(6)=?$
  - $\deg(1)=?$

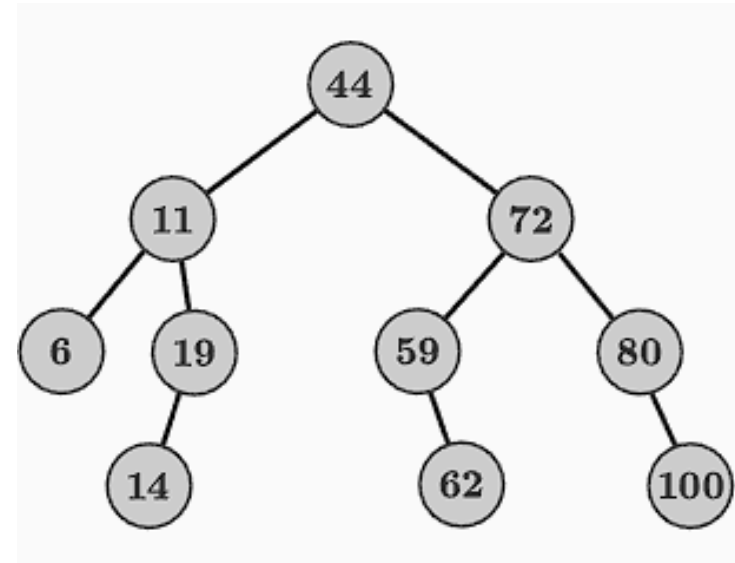


# Spezielle Klassen von Graphen

## Ein Baum

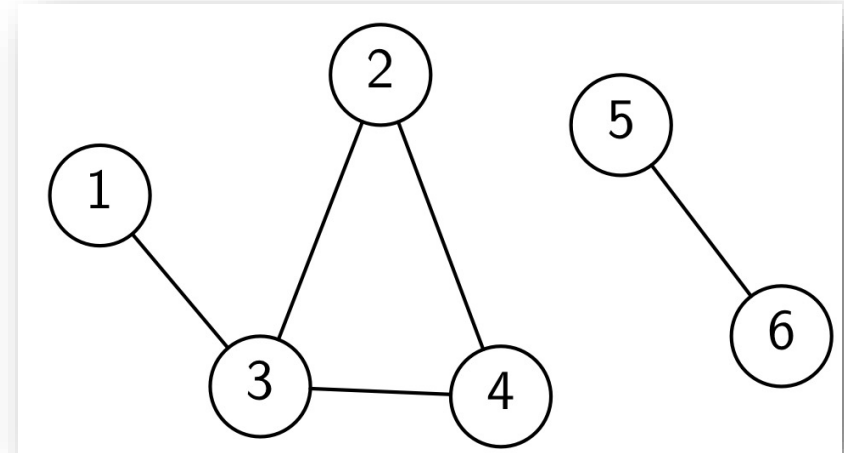
- Einfacher Graph
- Zusammenhängender Graph
- Azyklischer Graph

**Anzahl Kanten:** Anzahl Knoten - 1



# Einfacher Graph mit $n$ Knoten und $m$ Kanten

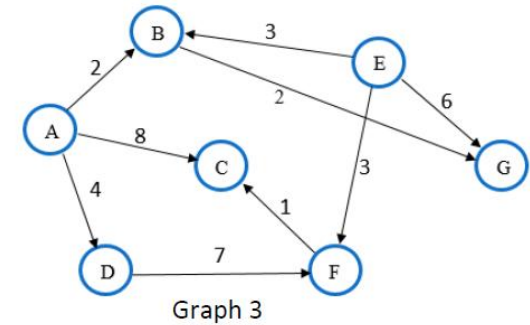
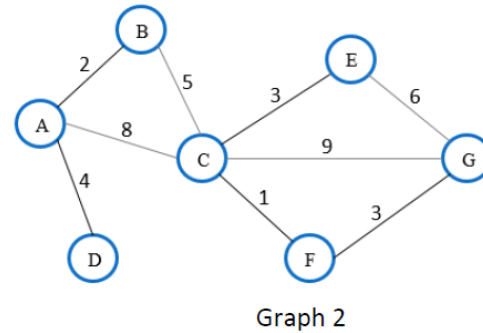
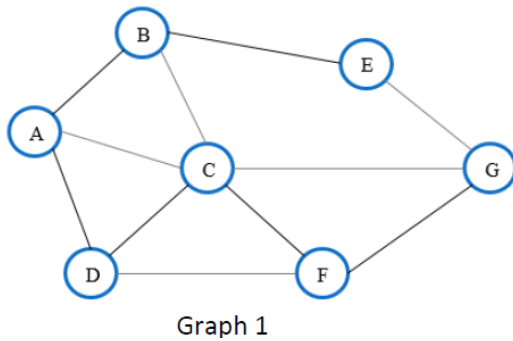
- Summe der Grade (= Anz. Kanten, die zum Knoten führen) aller Knoten ergibt  $2m$
- Für jeden zusammenhängenden Graph gilt:  
 $m \geq n - 1$
- Ein einfacher Graph kann nie mehr als  $\frac{n(n-1)}{2}$  Kanten haben



# Übungen zu Graphtheorie I - Theorieteil



# Aufgabe 4 – Beurteile ob die Aussagen zutreffen



	Graph 1	Graph 2	Graph 3
zusammenhängend	ja	ja	ja
zyklisch	ja	ja	nein
gewichtet	nein	ja	ja
gerichtet	nein	nein	ja

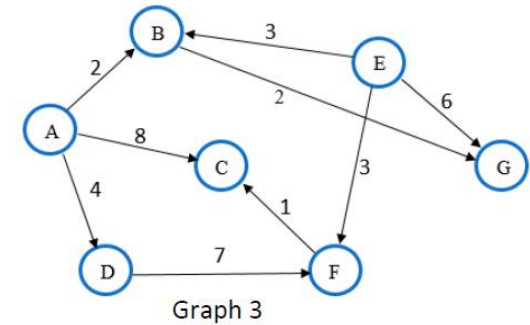
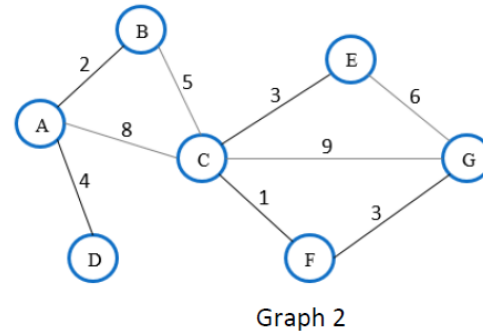
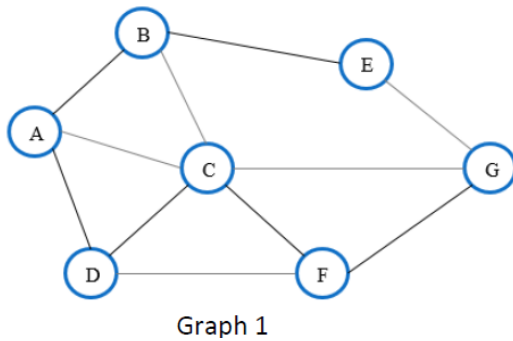
# Aufgabe 5 – Beantworte die Fragen

- Wie viele und welche Kanten müsste man beim Graph 1 mindestens entfernen, so dass der Graph nicht mehr zusammenhängend ist?

2: B,E und E,G

- Wie viele und welche Kanten müsste man beim Graph 2 mindestens entfernen, so dass der Graph nicht mehr zyklisch ist?

# Aufgabe 4 – Beurteile ob die Aussagen zutreffen



	Graph 1	Graph 2	Graph 3
zusammenhängend	Ja	Ja	Ja
zyklisch	Ja	Ja	Nein
gewichtet	Nein	Ja	Ja
gerichtet	Nein	Nein	Ja

## Aufgabe 5 – Beantworte die Fragen

- Wie viele und welche Kanten müsste man beim Graph 1 mindestens entfernen, so dass der Graph nicht mehr zusammenhängend ist?

Man müsste 2 Kanten entfernen:  $(B,E)$  und  $(E,G)$ . Einen einzelnen Knoten abzuspalten reicht.

- Wie viele und welche Kanten müsste man beim Graph 2 mindestens entfernen, so dass der Graph nicht mehr zyklisch ist?

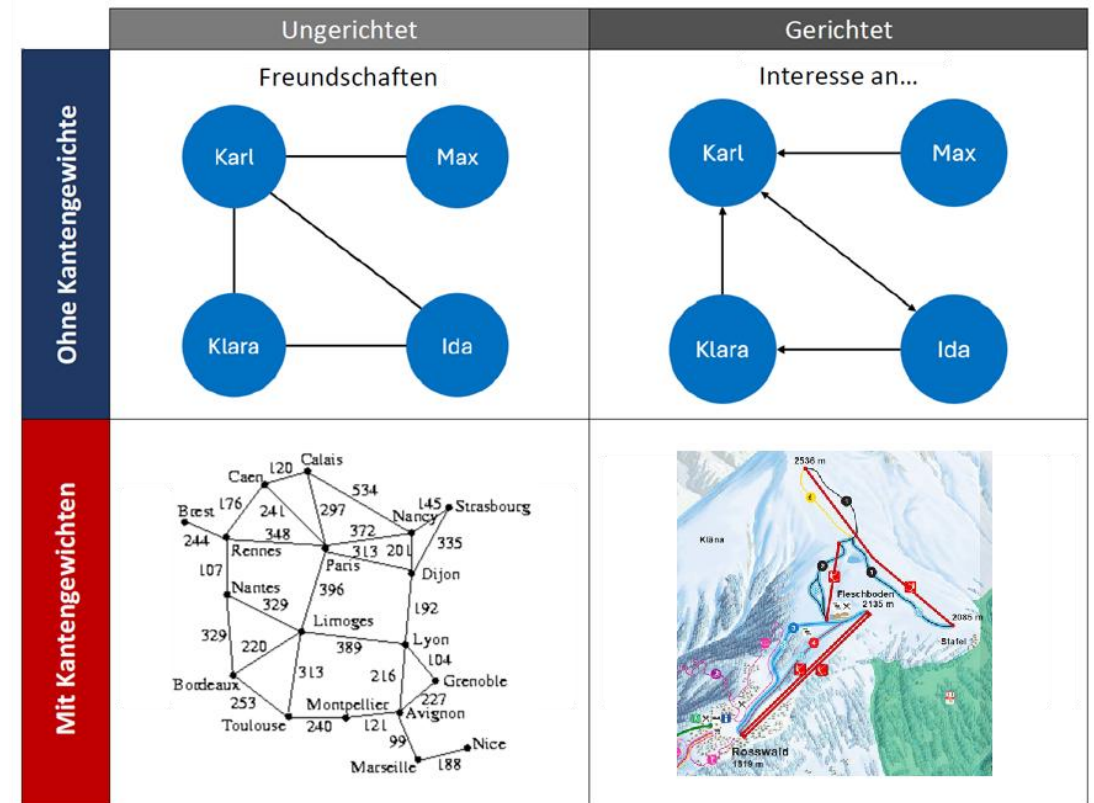
Es gibt mehrere Lösungen. Man müsste 3 Kanten entfernen:  $(A,B)$ ,  $(E,G)$  und  $(F,G)$ .

# Wozu das Ganze?

The slide features a dark green background. A thick green horizontal bar spans the width of the slide, positioned below the title. Below this bar, there are several thin, light green horizontal lines of varying lengths, creating a stepped or layered effect on the right side of the slide.

# Anwendungen von Graphentheorie

- Anwendungen sind sehr vielfältig/unterschiedlich
- Je nach Anwendung kommen unterschiedliche Graphen zum Einsatz
- Algorithmen (und deshalb auch Programme/Bibliotheken) bleiben aber zumeist die gleichen

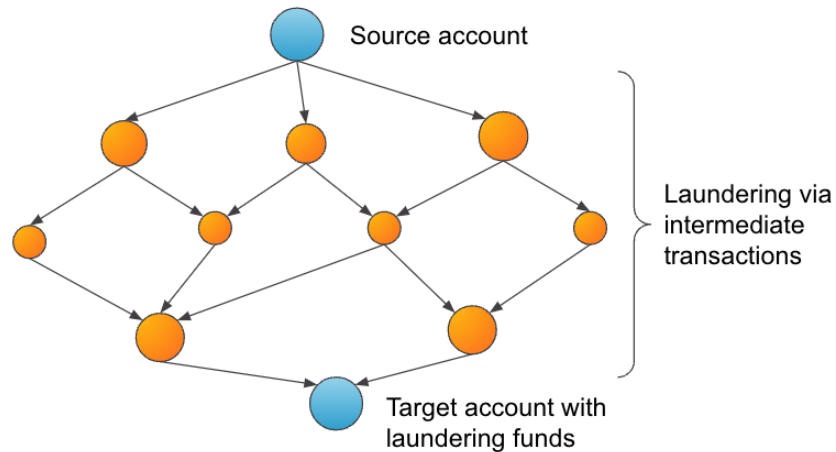


# Graphentheorie in Python

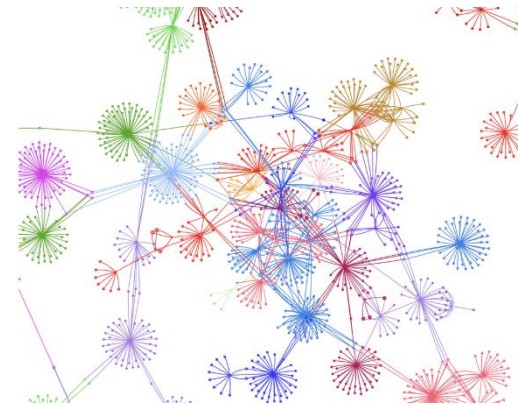


# Graphen in Python mit NetworkX

- Es gibt zahlreiche Möglichkeiten, einen Graph in Python abzubilden
- Eine der meistbenutzten Bibliotheken ist NetworkX
- Wird in vielen anderen Szenarien verwendet



Geldwäscherei-Erkennung



Gesetzestexte (USA)

# Graphen in Python mit NetworkX

Graphentheorie\_Aufgabe\_A.py

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

G = nx.Graph() # Graph Objekt (Behälter)

# Definiere Node IDs
nodes = ["A", "B", "C", "D", "E", "F", "G"]

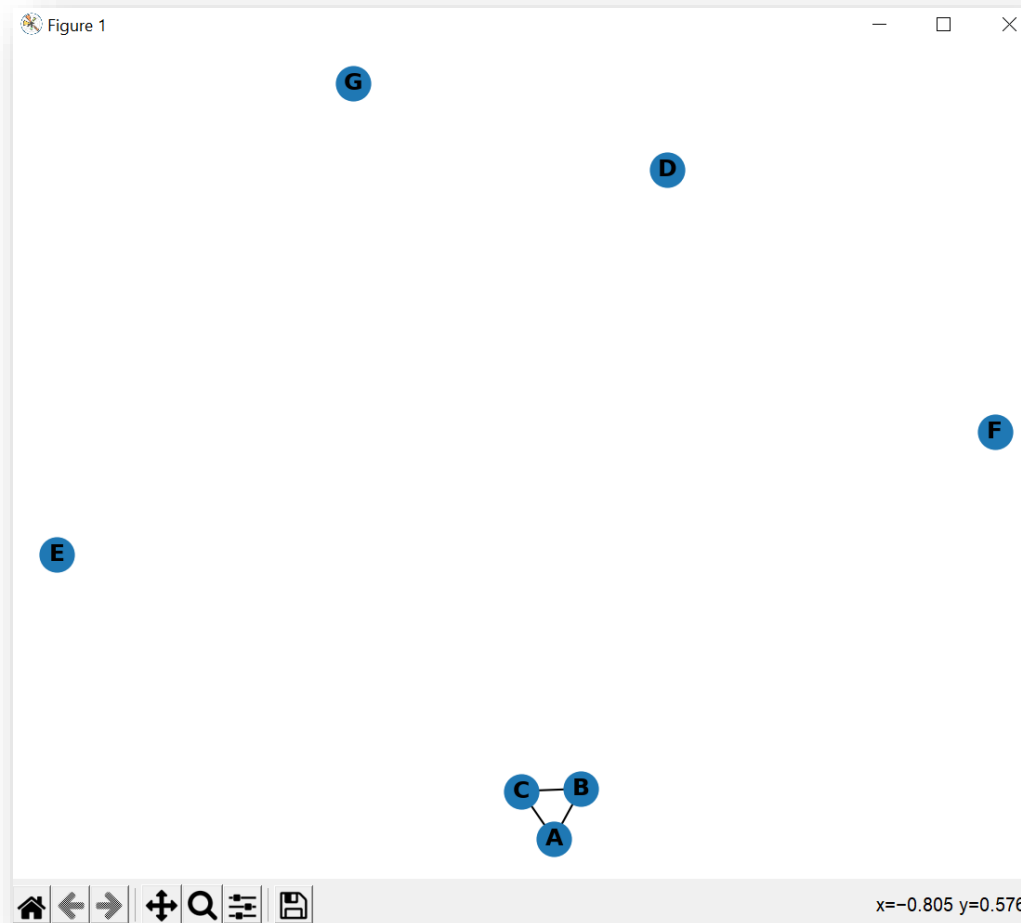
# Definiere Kanten - tuple (id_1, id_2) heisst dass id_1 und id_2 miteinander durch eine Kante verbunden sind
edges = [("A", "B"), ("A", "C"), ("B", "C")]

# füge Knoten und Kanten zu G
G.add_nodes_from(nodes)
G.add_edges_from(edges)

# Zeichne den Graph G
nx.draw(G, with_labels=True, font_weight='bold')
plt.show()
```

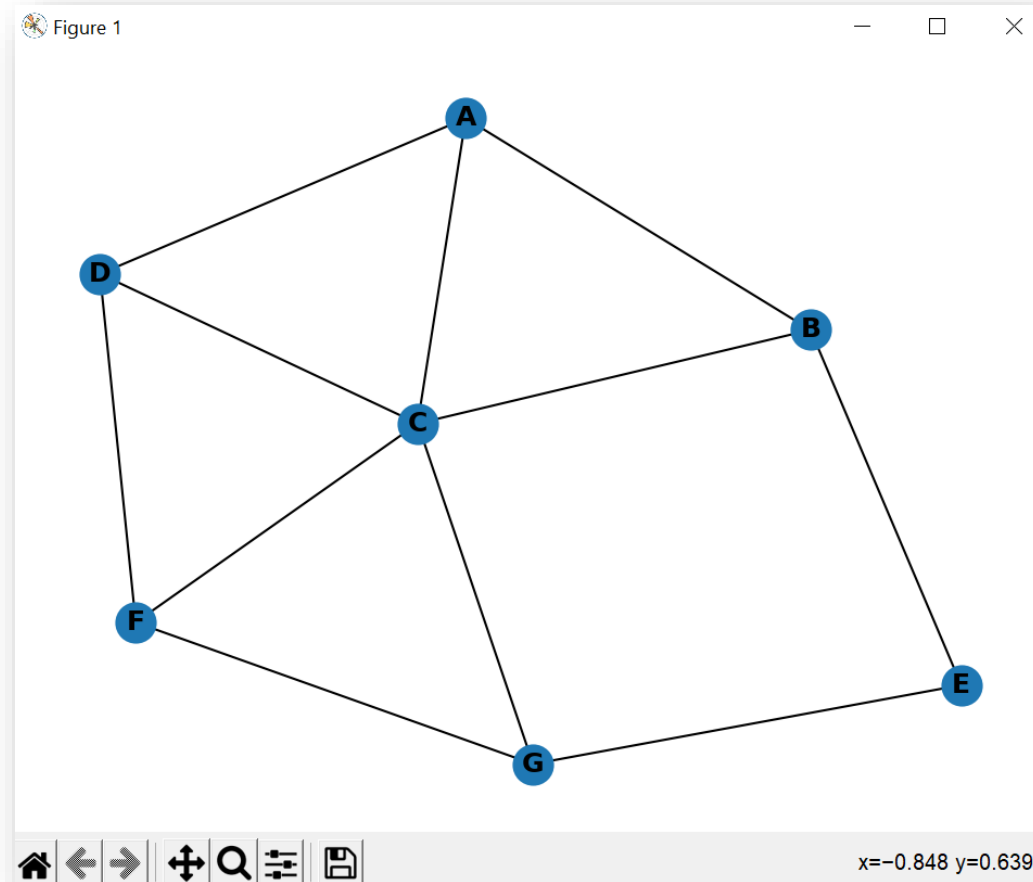
# Graphen in Python mit NetworkX

Graphentheorie\_Aufgabe\_A.py



# Aufgabe A: Zeichne den Graph 1

Graphentheorie\_Aufgabe\_A\_Loesung.py



# Aufgabe B1: Gib die Adjazenzmatrix aus

- Finde heraus, wie du aus NetworkX die Adjazenzmatrix bekommst - als 2-Dimensionale Liste – das heisst eine Liste von Zeilen, und diese Zeilen sind wiederum Listen.
- Gib sie aus – Verifiziere mit deiner Adjazenz-Matrix oben

```
Python 3.7.9 (bundled)
>>> %Run Graphentheorie_Aufgabe_B.py

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

# Aufgabe B2: Gib die Adjazenzmatrix aus

- Versuche, mit einer Adjazenzmatrix den gleichen Graph zu definieren wie in A und B1
- Betrachte, was du dafür am Code ändern musst

```
Graphentheorie_Aufgabe_B2.py x
1 import matplotlib.pyplot as plt
2 import pandas
3 import networkx as nx
4
5 G = nx.Graph() # Graph Objekt (Behälter)
6
7 AdjMatrix = [[0,1,1,1,1,1,1],
8              [1,0,1,1,1,1,1],
9              [1,1,0,1,1,1,1],
10             [1,1,1,0,1,1,1],
11             [1,1,1,1,0,1,1],
12             [1,1,1,1,1,0,1],
13             [1,1,1,1,1,1,0]]
14
15 # node labels
16 labels = ["A", "B", "C", "D", "E", "F", "G"]
17
18 # wir brauchen sogenannte "Pandas"-Dataframes
19 AdjMatrix = pandas.DataFrame(AdjMatrix, index=labels, columns=labels)
20
21 # Zeichne den Graph
22 nx.draw_networkx(nx.from_pandas_adjacency(AdjMatrix))
23 plt.show()
```

# Aufgabe C: 7x7 Graph-Generator

- Du hast ein Python Programm welches zuerst eine 7x7 Adjazenzmatrix generiert, und diese dann als Graph druckt.
- Nutze dazu die folgende Funktion, welche mit Wahrscheinlichkeit 0-100% eine 1 generiert (sonst 0).
- Generiere einen zufälligen Graph – dazu sollst du für jede Zeile und jede Spalte den Wert auf random01() setzen

```
# funktion für zufällig 0 oder 1 - mit 40% Wahrscheinlichkeit für eine 1
def random01():
    if random.random()<0.4:
        return 1
    else:
        return 0
```

# Zusätzliche Übungsaufgaben



# Zusatzaufgabe I

Übersetze folgende Adjazenzmatrix in einen Graphen und beurteile seine Eigenschaften

Eigenschaft	Ja/Nein
zusammenhängend	
zyklisch	
gewichtet	
gerichtet	

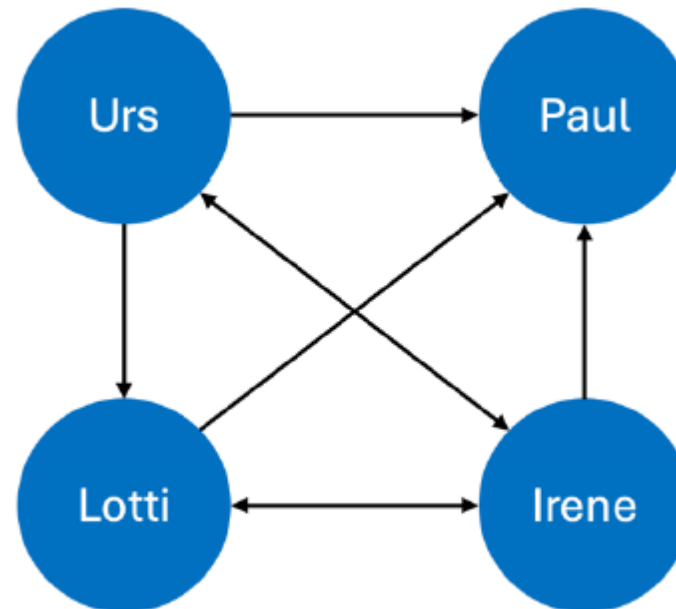
Adjazenzmatrix:

	Urs	Paul	Lotti	Irene
Urs	0	1	1	1
Paul	0	0	0	0
Lotti	0	1	0	1
Irene	1	1	1	0

# Lösung Zusatzaufgabe I

Folgender Graph entsteht

Eigenschaften	Ja/Nein
zusammenhängend	Ja
zyklisch	Ja
gewichtet	Nein
gerichtet	Ja



# Zusatzaufgabe II

Finde eine Anwendung von Graphen mit folgenden Eigenschaften

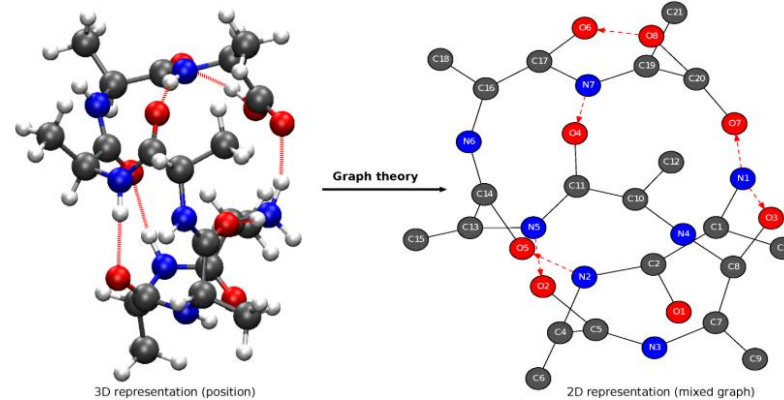
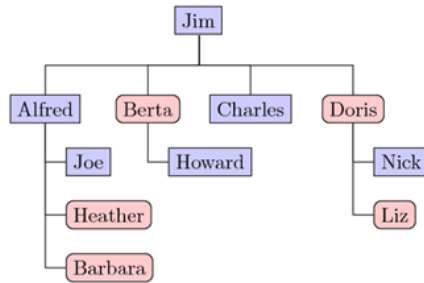
Eigenschaften	Ja/Nein
zusammenhängend	Ja
zyklisch	Ja
gewichtet	Nein
gerichtet	Ja

Eigenschaften	Ja/Nein
zusammenhängend	Nein
zyklisch	Ja
gewichtet	Nein
gerichtet	Nein

# Lösung Zusatzaufgabe II

Finde eine Anwendung von Graphen mit folgenden Eigenschaften

"Descendants of Jim"



Eigenschaften	Ja/Nein
---------------	---------

zusammenhängend	Ja
-----------------	----

zyklisch	Nein
----------	------

gewichtet	Nein
-----------	------

gerichtet	Ja
-----------	----

Eigenschaften	Ja/Nein
---------------	---------

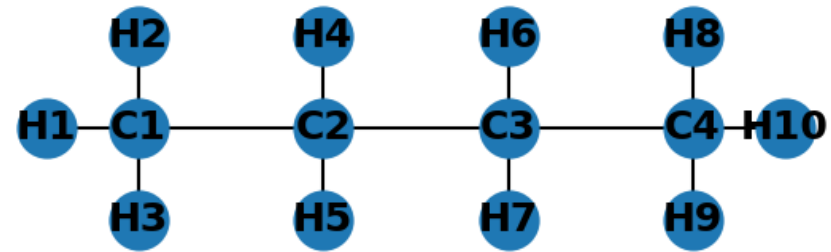
zusammenhängend	Nein
-----------------	------

zyklisch	Ja
----------	----

gewichtet	Nein
-----------	------

gerichtet	Nein
-----------	------

# Zusatzaufgabe III



- In der Datei Graphentheorie\_Zusatzaufgabe\_Butan.py wird das Molekül Butan gezeichnet.
- Lasse dir von NetoworkX zuerst die Adjazenzmatrix liefern
- Danach: Prüfe mit einem Algorithmus, ob jedes C-Atom 4 Verbindungen hat, und jedes H-Atom eine.
- Mit deinem Graph-Prüfalgorithmus solltest du nun beliebige Kohlenwasserstoffe aus C und H prüfen können.