Objektorientert programmering i Python

IN1000

Høst 2025 – uke 10

Siri Moe Jensen og Geir Kjetil Sandve

Innhold uke 10

- Noen sentrale datastrukturer for programmering
  - lenkede lister
  - trær
  - grafer

- Eksempler med livekoding: Offentlig transport som
  - lenket liste
  - graf

- Input til innlevering 8
  - To- og flerdimensjonale strukturer

Semesteret videre

Uke	Forelesninger	Gruppetimer	Større egne oppgaver
10	Sentrale datastrukturer for programmering	Fra ukens forelesning, oblig 8-trening	Oblig 8
11	Live-koding av et større, objektorientert program	Eksamensoppgaver mm	
12	Pensum-gjennomgang og introduksjon til Inspera		Prøveeksamen i Inspera (rettes ikke)
13	Gjennomgang prøveeksamen, eksamenstips	Repetisjonskurs	
14-15	Fredagspython Ingen forelesninger eller ordinære gruppetimer		
16	Eksamen (5.12)		

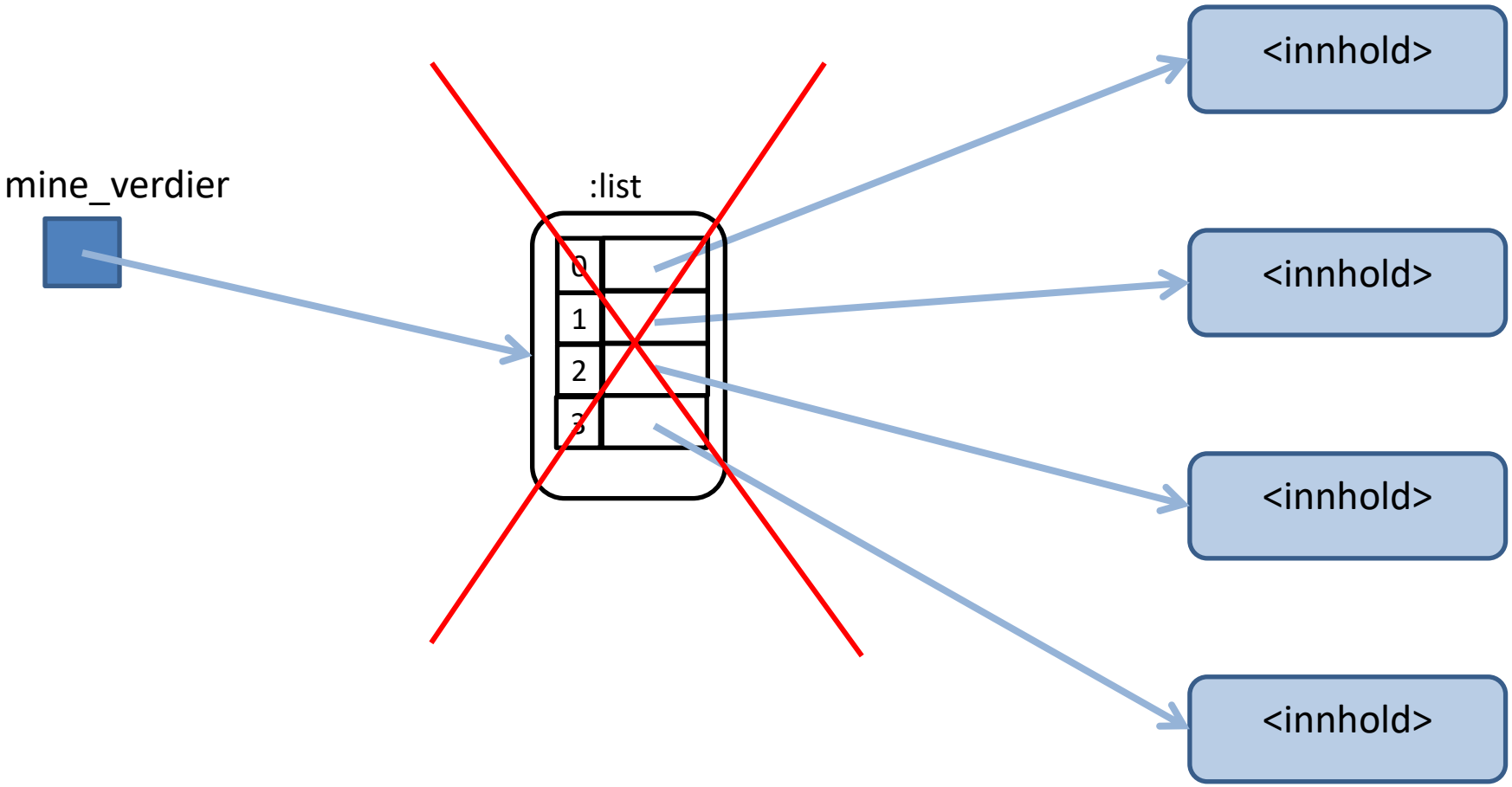
- Fredagspython fortsetter til og med uken før eksamen

Å lage egne beholdere (containers)

- Hvordan sørge for at vi kan lagre alt vi trenger av data?
- Trenger datastrukturer for å lagre ukjent antall objekter. I Python har vi **lister**, **ordbøker** og **mengder** – hva om disse ikke fantes?
- Vi kan gå i løkke og opprette objekter etter behov – disse kan brukes fra ulike steder i programmet vårt, bare vi har en referanse til objektet.
- Men hvordan sørge for nok referansevariabler til et ukjent antall objekter, slik at vi kan finne dem igjen og bruke dem senere?

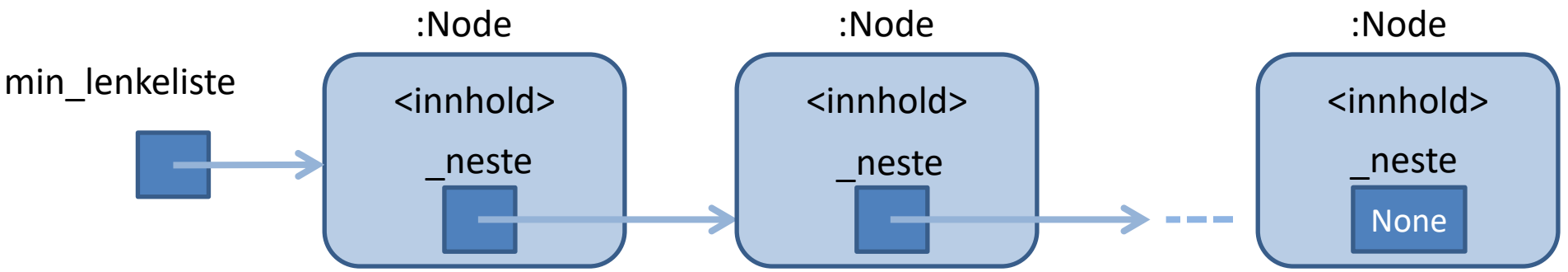
$\Rightarrow$ en klassisk datastruktur for et ukjent antall elementer er en *lenket liste*

Egne beholdere



Lenkede lister

- Poenget med en lenket liste er at for hvert nye objekt vi lager – så lager vi samtidig en referansevariabel som kan referere til et annet, nytt objekt
- dvs hvert objekt må kunne referere til et annet objekt i listen
- dermed får vi en lenket liste av objekter – og trenger bare én variabel som refererer til det første objektet

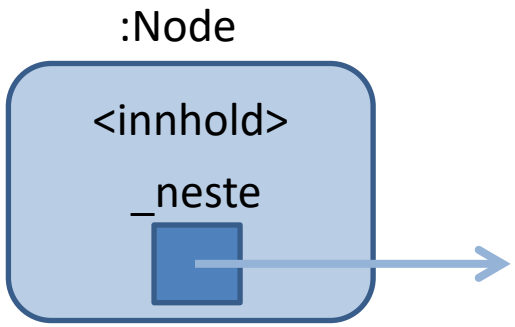


Lenket liste

Vi lager en generell klasse for objektene i lenkelisten

- Klassen Node

- **Grensesnitt** for klassen Node:
  - lese, endre, skrive ut, ... innhold
  - konstruktør med parameter innhold
  - legg til ny etterfølger
  - hent etterfølger
  - (fjern etterfølger)



Klassen Node: Implementasjon

- Datastruktur Node-klassen
  - innholdet objektene skal representere (her: en streng)
  - en referansevariabel så vi får tak i neste objekt

```
class Node :
    def __init__(self, nytt) :
        self._innhold = nytt
        self._neste = None

    def ny_etterfolger (self, ny) :
        self._neste = ny

    def hent_neste (self) :
        return self._neste

     # + metoder for manipulering av innholdet
```

Skriv et program som oppretter en lenket liste med 5 noder, og deretter skriver ut innholdet i alle nodene.

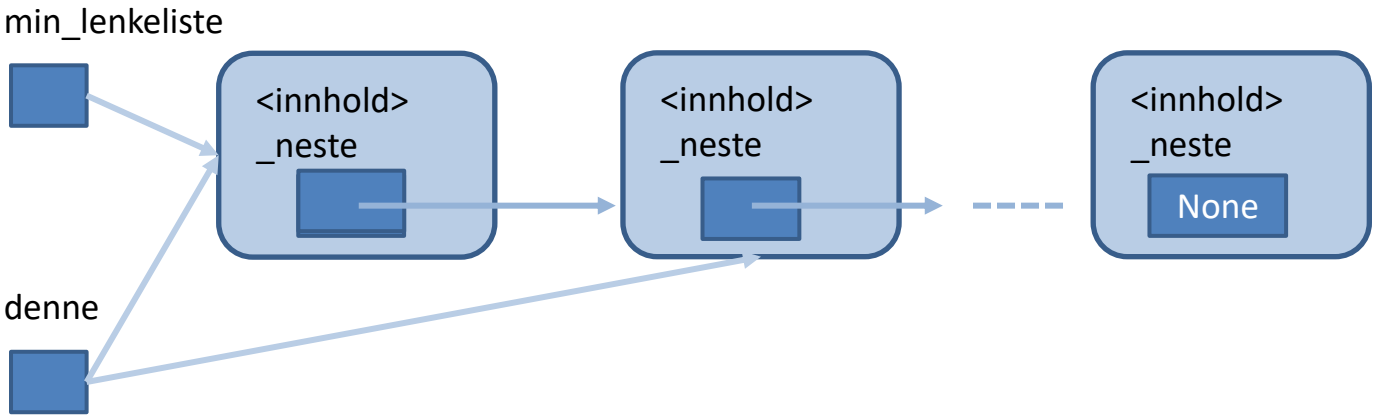
Bygge opp en lenkeliste

```
class Node :
    def __init__(self, nytt) :
        self._innhold = nytt
        self._neste = None

    def ny_etterfolger (self, ny) :
        self._neste = ny

    def hent_neste (self) :
        return self._neste

    def hent_innhold(self):
        return self._innhold
```

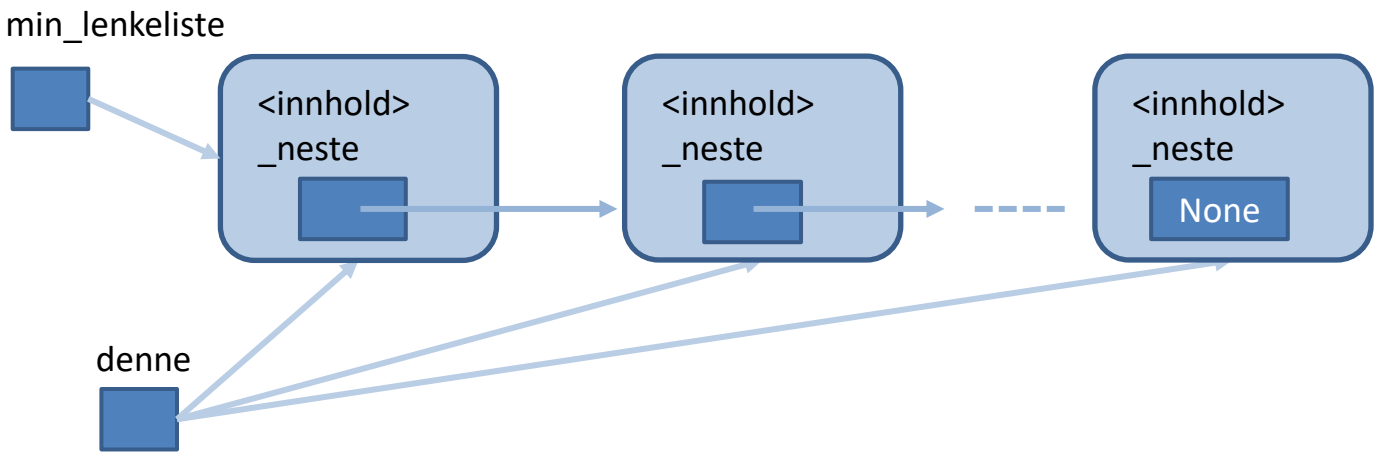



```
min_lenkeliste = Node("1")
denne = min_lenkeliste
for teller in range(2,6):
    denne.ny_etterfolger(Node(str(teller)))
    denne = denne.hent_neste()
```

Flytte seg gjennom nodene i en liste

```
min_lenkeliste = Node("1")
denne = min_liste
for i in range(2,6):
    ny = Node(str(i))
    denne.ny_etterfolger(ny)
    denne = denne.hent_neste()

denne = min_lenkeliste
while denne is not None:
    print(denne.hent_innhold())
    denne = denne.hent_neste()
```



Live eksempel I - trikk

Et eksempel der lenkene (referansene) representerer viktig informasjon om objektene våre (ikke bare hjelper oss å lagre et ukjent antall objekter).

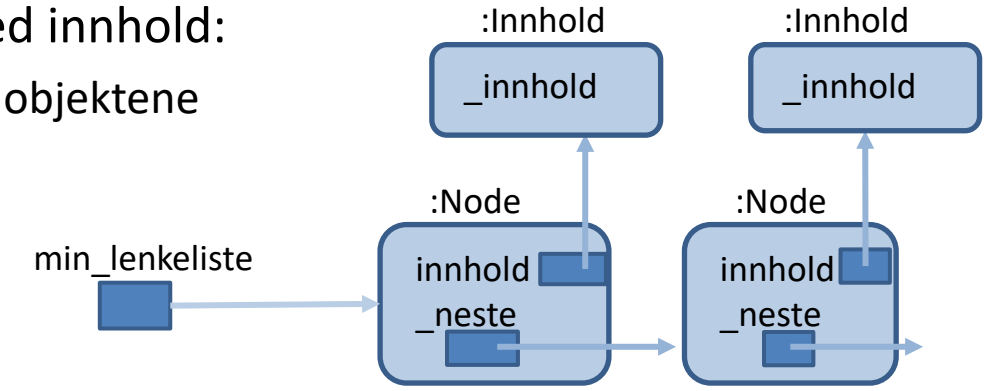
Her er nodene *stasjoner på en trikkerute*, dvs. den lenkede listen består av objekter av klassen Stasjon som inneholder stasjonsnavn og referanse til neste stasjon.

<se egen presentasjon og kildekode>

Innkapsling og generalisering av lenkeliste

- Vi kan organisere et «uendelig» antall noder (f eks stasjoner) ved å opprette og lenke inn nye objekter av klassene Node (eller Stasjon, eller noe annet) ved behov.

- Designmessige ulemper ved at lenkene ligger i objektene med innhold:
  - den som skal bruke klassen må kjenne til sammenlenkingen av objektene
  - liste-håndteringen må skrives på nytt for hver klasse



- Tiltak:
  - Node-klassen refererer til innholdsobjekter og neste Node
  - Skjule klassen Node (Stasjon) inne i en klasse Lenkeliste (LagRute).
  - Denne vil fungere som en beholder som håndterer (og skjuler) alle lenke-operasjoner, og kun tilbyr brukervennlige tjenester for å opprette og administrere noder (for eksempel Stasjoner) i sitt grensesnitt
  - Ta emnet IN1010 til våren ☺

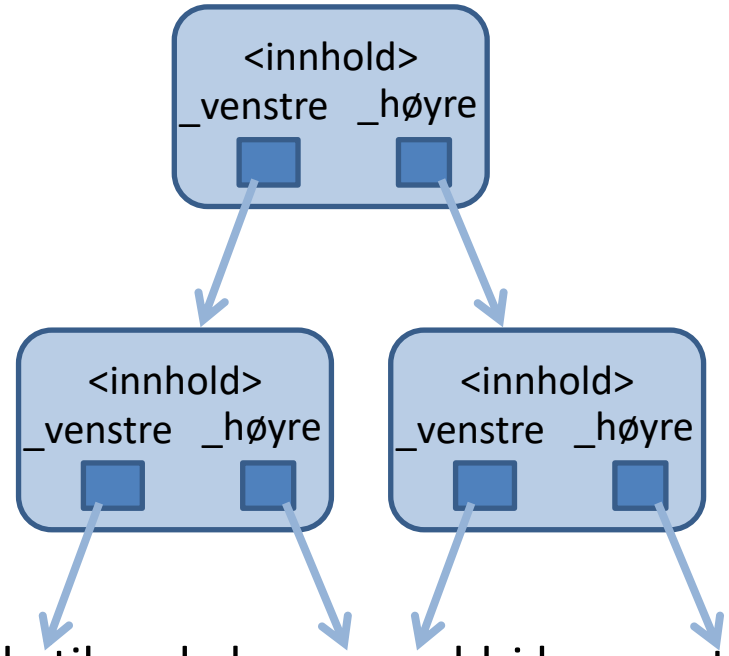
Andre sentrale datastrukturer

- Lenkede lister har noen ulemper
  1. Hvis man har mange noder, blir veien gjennom strukturen veldig lang, spesielt til siste node
  2. Hvis man skal representere noder med flere enn en relasjon til andre noder, holder det ikke med én neste-referanse

1. Punkt 1 kan avhjelpes noe vha endringer i implementasjon av nodene og lenkene. Mer om dette i IN1010.

2. Kan være nyttig med andre strukturer, for eksempel *trær* (hver node kan referere til flere noder, ingen sykler) eller *grafer.* Trær og grafer er sentralt innhold i IN2010 «Algoritmer og datastrukturer».

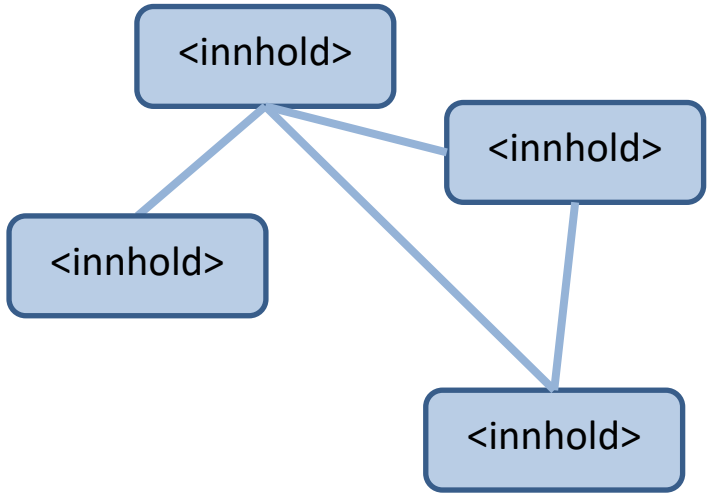
Trær



- Et **tre** har en node definert som **roten** i treet.
- Hver node kan ha referanser til flere andre noder (barn)
- Trær har ikke sykler, dvs om man følger referanser fra node til node kan man aldri komme tilbake til en node man har vært i tidligere
- Spesielt binære trær (hver node har maks to barn) brukes ofte til å strukturere data for effektive oppslag, sortering og traversering
- Trær kan representere hierarkiske strukturer som familietrær, organisasjonskart

Generelt: Grafer

- En graf består av en mengde noder, og en mengde **kanter**, der hver kant forbinder to noder med hverandre.
- Kantene kan, men trenger ikke, ha en retning (grafen kan være **rettet** eller **urettet**)
- Kan representere mange ulike strukturer der referanser til andre noder angir en forbindelse
- Eksempel: Kollektivnettet i Oslo. Nodene er holdeplasser på en eller flere ruter, en referanse fra én node til en annen viser at det er mulig å reise fra den ene holdeplassen til den andre.



Live eksempel II - kollektivnettet

Innlevering 8

1. Denne har tidligere vært obligatorisk – og pensum og krav er de samme som før

2. Samme oppgave var i fjor første obligatoriske oppgave i IN1010 (men i Java)

- Conway's *Game of Life* - se f eks Wikipedia
- Husk:
  - Test for hver metode
  - Hvilken type har variablene du bruker til enhver tid?
  - Grensesnitt og innkapsling:
    - Celle håndterer sine data og operasjoner på disse
    - Rutenett håndterer rutenettet og operasjoner på dette

- Et mindre program med et rutenett gjennomgås på gruppene denne uken (Bondesjakk/ Tre på rad)

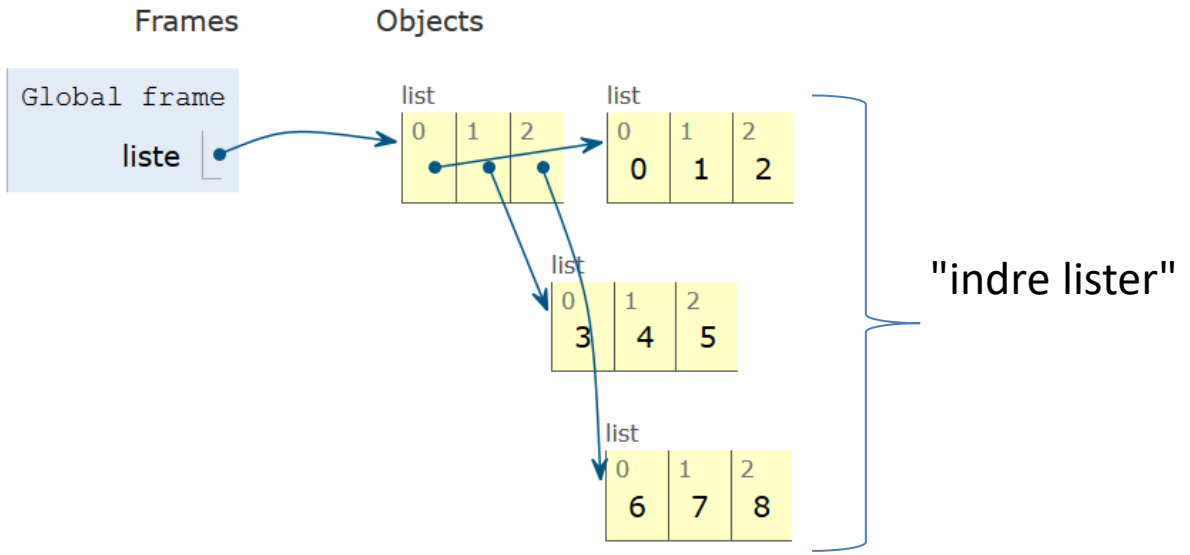
Liste av lister

- En liste inneholder referanser til elementer
- Elementene i (den ytre) listen kan være andre (indre) lister, som igjen kan ha lister som elementer, etc

- Python notasjon for å aksessere innholdet i lister:
  - liste[indeks]
  - For nøstede lister: liste[indeks1][indeks2][indeks3]

liste = [[0,1,2],[3,4,5],[6,7,8]]

Hva skrives ut her?
print(liste[0][1])



fra pythontutor.com

Mange anvendelser for 2-dimensjonale strukturer!

Eksempler med fysisk 2-dimensjonal utstrekning

- sjakkbrett
- kart
- tabell

eller mer abstrakte (skoleklasser 2A, 5C, 7B..) som det ikke er naturlig å illustrere todimensjonalt

Innholdet i de "innerste" listene er verdiene vi ønsker å representere og som kan slås opp – det kan være typer av f.eks. int, boolean, eller referanser til objekter av klasser vi har definert selv.

Kan tenke oss innholdet i en tabell

	0	1	2	3	4	5	..	..
0		[0][1]						
1					[1][4]			
2								
3				[3][3]				
4		[4][1]						
5	[5][0]							
..								
..								

- Her er elementene nummerert som om vi leser en bokside (med latinsk alfabet!):
  Vi leser en rad av gangen fra øverst til nederst, tegn for tegn fra venstre mot høyre.

- NB: Python bryr seg ikke om hvordan vi *tegner* denne strukturen – om første indeks viser rad eller kolonne! Men vi må selvsagt holde rede på dette så vi er konsistente gjennom hele programmet og i eventuelle visualiseringer.

Liste av lister: Indeksering

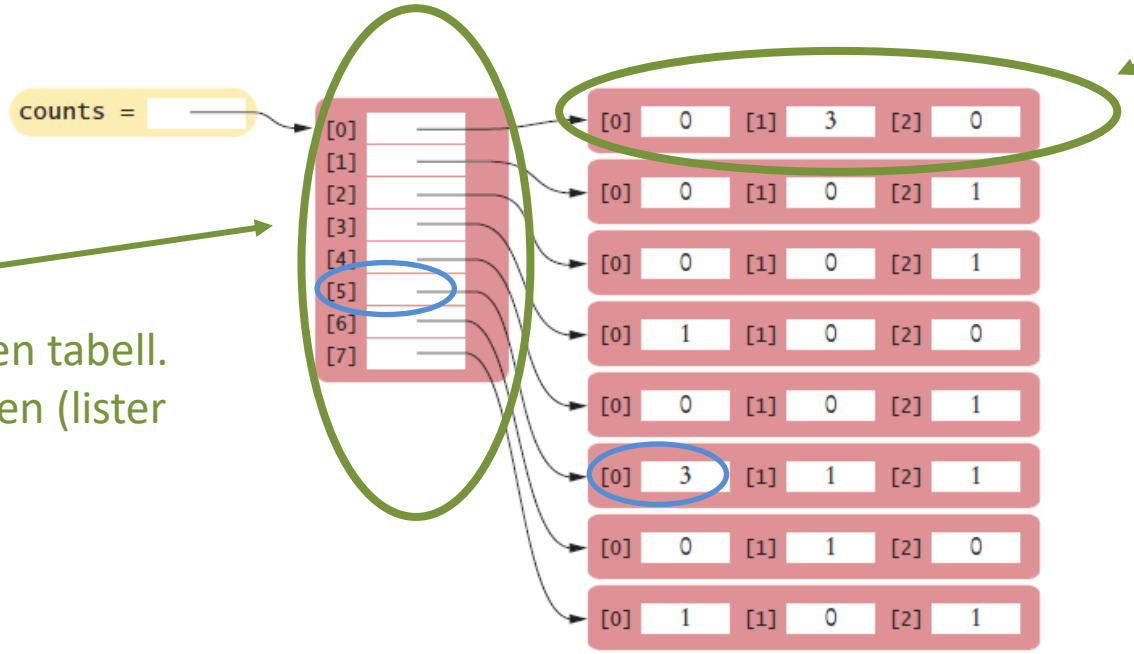
- En tabell med generiske indekser:

[i - 1][j - 1]	[i - 1][j]	[i - 1][j + 1]
[i][j - 1]	[i][j]	[i][j + 1]
[i + 1][j - 1]	[i + 1][j]	[i + 1][j + 1]

Liste av lister for en tabell

Fra læreboken:

- **Her bruker vi en liste av lister til å representere en tabell**
- **Har valgt å la den ytterste liste representere rader**
- **Da blir verdiene i de innerste listene det som er i en gitt kolonne på hver rad**
- innholdet i disse listene hentes ut med f eks counts[5][0]



Denne listen er én rad. Elementene er verdien i hver kolonne for denne raden

Denne listen representerer en tabell. Elementene er rader i tabellen (lister med verdiene i en rad)