

# LISTER

---

## Programming E25

# SAMLINGER

---

Samlinger (= Collections) er indbyggede datatyper i Python

# SAMLINGER

---

Samlinger (= Collections) er indbyggede datatyper i Python

Der er fire forskellige indbyggede datatyper, der bruges til at opbevare samlinger af data

- list
- tuple
- set
- dictionary

# SAMLINGER

---

Samlinger (= Collections) er indbyggede datatyper i Python

Der er fire forskellige indbyggede datatyper, der bruges til at opbevare samlinger af data

- list (= liste)
- tuple (= tupel)
- set (= mængde)
- dictionary (= ordbog)

# SAMLINGER

---

Samlinger (= Collections) er indbyggede datatyper i Python

Der er fire forskellige indbyggede datatyper, der bruges til at opbevare samlinger af data

- list (= liste) `['æble', 'banan', 'citron', 'kiwi']`
- tuple (= tupel)
- set (= mængde)
- dictionary (= ordbog)

# SAMLINGER

---

Samlinger (= Collections) er indbyggede datatyper i Python

Der er fire forskellige indbyggede datatyper, der bruges til at opbevare samlinger af data

- list (= liste) `['æble', 'banan', 'citron', 'kiwi']`
- tuple (= tupel) `('æble', 'banan', 'citron', 'kiwi')`
- set (= mængde)
- dictionary (= ordbog)

# SAMLINGER

---

Samlinger (= Collections) er indbyggede datatyper i Python

Der er fire forskellige indbyggede datatyper, der bruges til at opbevare samlinger af data

- list (= liste) `['æble', 'banan', 'citron', 'kiwi']`
- tuple (= tupel) `('æble', 'banan', 'citron', 'kiwi')`
- set (= mængde) `{'æble', 'banan', 'citron', 'kiwi'}`
- dictionary (= ordbog)

# SAMLINGER

---

Samlinger (= Collections) er indbyggede datatyper i Python

Der er fire forskellige indbyggede datatyper, der bruges til at opbevare samlinger af data

- list (= liste) `['æble', 'banan', 'citron', 'kiwi']`
- tuple (= tupel) `('æble', 'banan', 'citron', 'kiwi')`
- set (= mængde) `{'æble', 'banan', 'citron', 'kiwi'}`
- dictionary (= ordbog) `{'æble':2, 'banan':5, 'citron':0, 'kiwi': 3}`



# SAMLINGER

---

Samlinger (= Collections) er indbyggede datatyper i Python

Der er fire forskellige indbyggede datatyper, der bruges til at opbevare samlinger af data

- list (= liste) `['æble', 'banan', 'citron', 'kiwi']`
- tuple (= tupel) `('æble', 'banan', 'citron', 'kiwi')`
- set (= mængde) `{'æble', 'banan', 'citron', 'kiwi'}`
- dictionary (= ordbog) `{'æble':2, 'banan':5, 'citron':0, 'kiwi': 3}`

Vi skal introducere dem allesammen, men vi skal i første omgang kigge på lister og lidt tupler

# LISTER

---

Datatypen liste (= list) er en følge af endeligt mange elementer med en bestemt rækkefølge

# LISTER

---

Datatypen liste (= list) er en følge af endeligt mange elementer med en bestemt rækkefølge

Eksempler på lister:

- Kongerækken i DK

Gorm den gamle, Harald Blåtand, Svend Tveskæg, ... Margrethe 2, Frederik 10

# LISTER

---

Datatypen liste (= list) er en følge af endeligt mange elementer med en bestemt rækkefølge

Eksempler på lister:

- Kongerækken i DK

Gorm den gamle, Harald Blåtand, Svend Tveskæg, ... Margrethe 2, Frederik 10

- Tocifrede kvadrattal

16, 25, 36, 49, 64, 81

# LISTER

---

Datatypen liste (= list) er en følge af endeligt mange elementer med en bestemt rækkefølge

Eksempler på lister:

- Kongerækken i DK

Gorm den gamle, Harald Blåtand, Svend Tveskæg, ... Margrethe 2, Frederik 10

- Tocifrede kvadrattal

16, 25, 36, 49, 64, 81

- Tætte venner

Peter, Ida, Jens, Anne, Christian, Signe, Mette, Ole, Morten, Mikkel, Asta

# LISTER

---

Datatypen liste (= list) er en følge af endeligt mange elementer med en bestemt rækkefølge

Eksempler på lister:

- Kongerækken i DK

Gorm den gamle, Harald Blåtand, Svend Tveskæg, ... Margrethe 2, Frederik 10

- Tocifrede kvadrattal

16, 25, 36, 49, 64, 81

- Tætte venner

Peter, Ida, Jens, Anne, Christian, Signe, Mette, Ole, Morten, Mikkel, Asta

- Dagens indtag af drikke

Kaffe, Vand, Kaffe, Kaffe, Pepsi Max, Kaffe, Vand, IPA, Kaffe

# LISTER

---

Datatypen liste (= list) er en følge af endeligt mange elementer med en bestemt rækkefølge

Eksempler på lister:

- Kongerækken i DK

Gorm den gamle, Harald Blåtand, Svend Tveskæg, ... Margrethe 2, Frederik 10

- Tocifrede kvadrattal

16, 25, 36, 49, 64, 81

- Tætte venner

Peter, Ida, Jens, Anne, Christian, Signe, Mette, Ole, Morten, Mikkel, Asta

- Dagens indtag af drikke

Kaffe, Vand, Kaffe, Kaffe, Pepsi Max, Kaffe, Vand, IPA, Kaffe

- Mine frugter

Bananer, Æbler, Appelsiner

# SYNTAKS OG INDEKSERING

---

## Eksempler

```
['Æble', 'Banan', 'Appelsin']
```

```
[1, 3, 7, 12, 19]
```

```
[2, 'bold', 17, True, 5, [2, 4, 6]]
```



# SYNTAKS OG INDEKSERING

## Eksempler

```
['Æble', 'Banan', 'Appelsin']
```

```
[1, 3, 7, 12, 19]
```

```
[2, 'bold', 17, True, 5, [2, 4, 6]]
```

bemærk de firkantede parenteser og  
kommaerne, der adskiller elementerne

# SYNTAKS OG INDEKSERING

## Eksempler

```
['Æble', 'Banan', 'Appelsin']  
[1, 3, 7, 12, 19]  
[2, 'bold', 17, True, 5, [2, 4, 6]]
```

bemærk de firkantede parenteser og  
kommaerne, der adskiller elementerne

## Generelt

[element<sub>1</sub>, element<sub>2</sub> ... element<sub>n</sub>]

hvor elementerne kan være vilkårlige datatyper

# SYNTAKS OG INDEKSERING

---

Nul-indeksering

```
Frugter = ['Æble', 'Banan', 'Appelsin']  
Frugter[0]
```

# SYNTAKS OG INDEKSERING

---

Nul-indeksering

```
Frugter = ['Æble', 'Banan', 'Appelsin']  
Frugter[0]
```

'Æble'

# SYNTAKS OG INDEKSERING

---

Nul-indeksering

Første element har indeks 0

andet element har indeks 1

osv...

```
Frugter = ['Æble', 'Banan', 'Appelsin']  
Frugter[0]
```

'Æble'

# SYNTAKS OG INDEKSERING

---

Nul-indeksering

Første element har indeks 0

andet element har indeks 1

osv...

```
Frugter = ['Æble', 'Banan', 'Appelsin']  
Frugter[0]
```

'Æble'

```
Frugter[2]
```

'Appelsin'

# SYNTAKS OG INDEKSERING

---

Nul-indeksering

Første element har indeks 0

andet element har indeks 1

osv...

```
Frugter = ['Æble', 'Banan', 'Appelsin']  
Frugter[0]
```

'Æble'

```
Frugter[2]
```

'Appelsin'

```
Frugter[3]
```

# SYNTAKS OG INDEKSERING

Nul-indeksering

Første element har indeks 0

andet element har indeks 1

osv...

```
Frugter = ['Æble', 'Banan', 'Appelsin']  
Frugter[0]
```

'Æble'

```
Frugter[2]
```

'Appelsin'

```
Frugter[3]
```

```
-----  
IndexError  
Cell In[7], line 1  
----> 1 Frugter[3]
```

**IndexError:** list index out of range



# SYNTAKS OG INDEKSERING

---

Negativ indeksering

Frugter

`['Æble', 'Banan', 'Appelsin']`

# SYNTAKS OG INDEKSERING

---

Negativ indeksering

```
Frugter
```

```
['Æble', 'Banan', 'Appelsin']
```

```
Frugter[-1]
```

# SYNTAKS OG INDEKSERING

---

Negativ indeksering

```
Frugter
```

```
['Æble', 'Banan', 'Appelsin']
```

```
Frugter[-1]
```

```
'Appelsin'
```

# SYNTAKS OG INDEKSERING

---

Negativ indeksering

Sidste element har indeks  $-1$

andensidste element har indeks  $-2$

osv...

```
Frugter
```

```
['Æble', 'Banan', 'Appelsin']
```

```
Frugter[-1]
```

```
'Appelsin'
```

# SYNTAKS OG INDEKSERING

Negativ indeksering

Sidste element har indeks  $-1$

andensidste element har indeks  $-2$

osv...

```
Frugter
```

```
['Æble', 'Banan', 'Appelsin']
```

```
Frugter[-1]
```

```
'Appelsin'
```

```
# len bestemmer længden af listen, dvs antallet af elementer  
len(Frugter)
```

```
3
```

# SYNTAKS OG INDEKSERING

Negativ indeksering

Sidste element har indeks -1

andensidste element har indeks -2

osv...

```
Frugter
```

```
['Æble', 'Banan', 'Appelsin']
```

```
Frugter[-1]
```

```
'Appelsin'
```

```
# len bestemmer længden af listen, dvs antallet af elementer  
len(Frugter)
```

```
3
```

```
Frugter[len(Frugter)-1]
```

```
'Appelsin'
```

# SYNTAKS OG INDEKSERING

Negativ indeksering

Sidste element har indeks -1

andensidste element har indeks -2

osv...

```
Frugter
```

```
['Æble', 'Banan', 'Appelsin']
```

```
Frugter[-1]
```

```
'Appelsin'
```

```
# len bestemmer længden af listen, dvs antallet af elementer  
len(Frugter)
```

```
3
```

ikke nødvendig

```
Frugter[len(Frugter)-1]
```

```
'Appelsin'
```

# SPECIELLE LISTER

---

```
range(8)
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```



# SPECIELLE LISTER

---

```
range(8)
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

bemærk, at det svarer til indeksene i en vilkårlig liste

# SPECIELLE LISTER

---

```
range(8)
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

bemærk, at det svarer til indeksene i en vilkårlig liste

```
range(3, 9)
```

```
[3, 4, 5, 6, 7, 8]
```

# SPECIELLE LISTER

---

```
range(8)
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

bemærk, at det svarer til indeksene i en vilkårlig liste

```
range(3, 9)
```

```
[3, 4, 5, 6, 7, 8]
```

```
range(1, 12, 2)
```

# SPECIELLE LISTER

---

```
range(8)
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

bemærk, at det svarer til indeksene i en vilkårlig liste

```
range(3, 9)
```

```
[3, 4, 5, 6, 7, 8]
```

```
range(1, 12, 2)
```

```
[1, 3, 5, 7, 9, 11]
```

# SPECIELLE LISTER

---

```
range(8)
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

bemærk, at det svarer til indeksene i en vilkårlig liste

```
range(3, 9)
```

```
[3, 4, 5, 6, 7, 8]
```

```
range(1, 12, 2)
```

```
[1, 3, 5, 7, 9, 11]
```

```
range(start, stop, trin)
```

start angiver starten – er automatisk **0**, hvis den ikke er angivet  
stop angiver afslutningen, men ikke medtaget  
trin angiver springet – er automatisk **1**, hvis den ikke er angivet

# SPECIELLE LISTER

---

```
range(8)
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

bemærk, at det svarer til indeksene i en vilkårlig liste

```
range(3, 9)
```

```
[3, 4, 5, 6, 7, 8]
```

```
range(1, 12, 2)
```

```
[1, 3, 5, 7, 9, 11]
```

```
range(start, stop, trin)
```

start angiver starten – er automatisk **0**, hvis den ikke er angivet  
stop angiver afslutningen, men ikke medtaget  
trin angiver springet – er automatisk **1**, hvis den ikke er angivet

```
Bestem range(2, 10, 3)[1]
```

# SPECIELLE LISTER

---

```
range(8)
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

bemærk, at det svarer til indeksene i en vilkårlig liste

```
range(3, 9)
```

```
[3, 4, 5, 6, 7, 8]
```

```
range(1, 12, 2)
```

```
[1, 3, 5, 7, 9, 11]
```

```
range(start, stop, trin)
```

start angiver starten – er automatisk **0**, hvis den ikke er angivet  
stop angiver afslutningen, men ikke medtaget  
trin angiver springet – er automatisk **1**, hvis den ikke er angivet

Bestem `range(2, 10, 3)[1]` = 5

# BESKÆRING AF LISTER

---



# BESKÆRING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']  
venner[1:]
```

# BESKÆRING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']  
venner[1:]
```

```
['Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

# BESKÆRING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']  
venner[1:]
```

```
['Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[2:6]
```

# BESKÆRING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']  
venner[1:]
```

```
['Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[2:6]
```

```
['Ole', 'Ida', 'Eva', 'Søren']
```

# BESKÆRING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']  
venner[1:]
```

```
['Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[2:6]
```

```
['Ole', 'Ida', 'Eva', 'Søren']
```

```
venner[:4]
```

# BESKÆRING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']  
venner[1:]
```

```
['Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[2:6]
```

```
['Ole', 'Ida', 'Eva', 'Søren']
```

```
venner[:4]
```

```
['Anne', 'Peter', 'Ole', 'Ida']
```

# BESKÆRING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']  
venner[1:]
```

```
['Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[2:6]
```

```
['Ole', 'Ida', 'Eva', 'Søren']
```

```
venner[:4]
```

```
['Anne', 'Peter', 'Ole', 'Ida']
```

```
venner[:]
```

# BESKÆRING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']  
venner[1:]
```

```
['Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[2:6]
```

```
['Ole', 'Ida', 'Eva', 'Søren']
```

```
venner[:4]
```

```
['Anne', 'Peter', 'Ole', 'Ida']
```

```
venner[:]
```

```
['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```



# BESKÆRERING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']  
venner[1:]
```

```
['Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[2:6]
```

```
['Ole', 'Ida', 'Eva', 'Søren']
```

```
venner[:4]
```

```
['Anne', 'Peter', 'Ole', 'Ida']
```

```
venner[:]
```

```
['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[2:6:2]
```

# BESKÆRING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']  
venner[1:]
```

```
['Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[2:6]
```

```
['Ole', 'Ida', 'Eva', 'Søren']
```

```
venner[:4]
```

```
['Anne', 'Peter', 'Ole', 'Ida']
```

```
venner[:]
```

```
['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[2:6:2]
```

```
['Ole', 'Eva']
```

# ÆNDRING AF LISTER

---

venner

['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']

# ÆNDRING AF LISTER

---

```
venner
```

```
['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[3] = 'Signe'  
venner
```

# ÆNDRING AF LISTER

---

```
venner
```

```
['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[3] = 'Signe'  
venner
```

```
['Anne', 'Peter', 'Ole', 'Signe', 'Eva', 'Søren', 'Mette']
```

# ÆNDRING AF LISTER

---

```
venner
```

```
['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[3] = 'Signe'  
venner
```

```
['Anne', 'Peter', 'Ole', 'Signe', 'Eva', 'Søren', 'Mette']
```

```
venner.append('Bo')  
venner
```

```
['Anne', 'Peter', 'Ole', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']
```

# ÆNDRING AF LISTER

---

```
venner
```

```
['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[3] = 'Signe'  
venner
```

```
['Anne', 'Peter', 'Ole', 'Signe', 'Eva', 'Søren', 'Mette']
```

```
venner.append('Bo')  
venner
```

```
['Anne', 'Peter', 'Ole', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']
```

```
venner.remove('Ole')  
venner
```

```
['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']
```

# ÆNDRING AF LISTER

bemærk at  
lister erforanderlige

på engelsk – **mutable** eller  
**changeable**

**.append()** og **.remove()** er  
eksempler på metoder,  
der hører til typen list

```
venner
```

```
['Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette']
```

```
venner[3] = 'Signe'  
venner
```

```
['Anne', 'Peter', 'Ole', 'Signe', 'Eva', 'Søren', 'Mette']
```

```
venner.append('Bo')  
venner
```

```
['Anne', 'Peter', 'Ole', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']
```

```
venner.remove('Ole')  
venner
```

```
['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']
```



# GENNEMLØB AF LISTER

---

Lister er ordnede, og derfor kan de gennemløbes med **for** – løkker

```
venner = ['Signe', 'Peter', 'Eva']  
  
for ven in venner:  
    print(ven)
```

# GENNEMLØB AF LISTER

---

Lister er ordnede, og derfor kan de gennemløbes med **for** – løkker

```
venner = ['Signe', 'Peter', 'Eva']  
  
for ven in venner:  
    print(ven)
```

Signe  
Peter  
Eva

# GENNEMLØB AF LISTER

---

Lister er ordnede, og derfor kan de gennemløbes med **for** – løkker

```
venner = ['Signe', 'Peter', 'Eva']  
  
for ven in venner:  
    print(ven)
```

Signe  
Peter  
Eva

```
for i in range(len(venner)):  
    print(venner[i])
```

Signe  
Peter  
Eva

# GENNEMLØB AF LISTER

---

Lister er ordnede, og derfor kan de gennemløbes med **for** – løkker

```
venner = ['Signe', 'Peter', 'Eva']  
  
for ven in venner:  
    print(ven)
```

Signe  
Peter  
Eva

```
for i in range(len(venner)):  
    print(venner[i])
```

Signe  
Peter  
Eva

# GENNEMLØB AF LISTER

---

Lister er ordnede, og derfor kan de gennemløbes med **for** – løkker

```
venner = ['Signe', 'Peter', 'Eva']  
  
for ven in venner:  
    print(ven)
```

Signe  
Peter  
Eva

range(3) =

```
for i in range(len(venner)):  
    print(venner[i])
```

Signe  
Peter  
Eva

# GENNEMLØB AF LISTER

---

Lister er ordnede, og derfor kan de gennemløbes med **for** – løkker

```
venner = ['Signe', 'Peter', 'Eva']  
  
for ven in venner:  
    print(ven)
```

Signe  
Peter  
Eva

$\text{range}(3) = [0, 1, 2]$

```
for i in range(len(venner)):  
    print(venner[i])
```

Signe  
Peter  
Eva

# GENNEMLØB AF LISTER

---

Lister er ordnede, og derfor kan de gennemløbes med **for** – løkker

Vi siger, at lister er **iterable**

```
venner = ['Signe', 'Peter', 'Eva']  
  
for ven in venner:  
    print(ven)
```

Signe  
Peter  
Eva

$\text{range}(3) = [0, 1, 2]$

```
for i in range(len(venner)):  
    print(venner[i])
```

Signe  
Peter  
Eva

# GENNEMLØB AF LISTER

Lister er ordnede, og derfor kan de gennemløbes med **for** – løkker

Vi siger, at lister er **iterable**

```
venner = ['Signe', 'Peter', 'Eva']
```

```
for ven in venner:  
    print(ven)
```

den smarteste måde

```
Signe  
Peter  
Eva
```

```
for i in range(len(venner)):  
    print(venner[i])
```

```
Signe  
Peter  
Eva
```



# SORTERING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']  
venner.sort()  
venner
```

# SORTERING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']  
venner.sort()  
venner
```

```
['Anne', 'Bo', 'Eva', 'Mette', 'Peter', 'Signe', 'Søren']
```

# SORTERING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']  
venner.sort()  
venner
```

```
['Anne', 'Bo', 'Eva', 'Mette', 'Peter', 'Signe', 'Søren']
```

```
tal = [3, 7, 1, 5, 9, 2]  
tal.sort(reverse = True)  
tal
```

# SORTERING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']  
venner.sort()  
venner
```

['Anne', 'Bo', 'Eva', 'Mette', 'Peter', 'Signe', 'Søren']

```
tal = [3, 7, 1, 5, 9, 2]  
tal.sort(reverse = True)  
tal
```

[9, 7, 5, 3, 2, 1]

# SORTERING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']  
venner.sort()  
venner
```

```
['Anne', 'Bo', 'Eva', 'Mette', 'Peter', 'Signe', 'Søren']
```

```
tal = [3, 7, 1, 5, 9, 2]  
tal.sort(reverse = True)  
tal
```

```
[9, 7, 5, 3, 2, 1]
```

```
blandet = [1, 'Peter', 7, 'Ida', 2]  
blandet.sort()  
blandet
```

# SORTERING AF LISTER

```
venner = ['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']  
venner.sort()  
venner
```

```
['Anne', 'Bo', 'Eva', 'Mette', 'Peter', 'Signe', 'Søren']
```

```
tal = [3, 7, 1, 5, 9, 2]  
tal.sort(reverse = True)  
tal
```

```
[9, 7, 5, 3, 2, 1]
```

```
blandet = [1, 'Peter', 7, 'Ida', 2]  
blandet.sort()  
blandet
```

```
-----  
TypeError                                Traceback (most recent call last)  
Cell In[50], line 2  
      1 blandet = [1, 'Peter', 7, 'Ida', 2]  
----> 2 blandet.sort()  
      3 blandet
```

```
TypeError: '<' not supported between instances of 'str' and 'int'
```

# SORTERING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']  
venner.sort()  
venner
```

```
['Anne', 'Bo', 'Eva', 'Mette', 'Peter', 'Signe', 'Søren']
```

```
tal = [3, 7, 1, 5, 9, 2]  
tal.sort(reverse = True)  
tal
```

```
[9, 7, 5, 3, 2, 1]
```

```
blandet = [1, 'Peter', 7, 'Ida', 2]  
blandet.sort()  
blandet
```

```
blandet = ['1', 'Peter', '7', 'Ida', '2']  
blandet.sort()  
blandet
```

# SORTERING AF LISTER

---

```
venner = ['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']  
venner.sort()  
venner
```

```
['Anne', 'Bo', 'Eva', 'Mette', 'Peter', 'Signe', 'Søren']
```

```
tal = [3, 7, 1, 5, 9, 2]  
tal.sort(reverse = True)  
tal
```

```
[9, 7, 5, 3, 2, 1]
```

```
blandet = [1, 'Peter', 7, 'Ida', 2]  
blandet.sort()  
blandet
```

```
blandet = ['1', 'Peter', '7', 'Ida', '2']  
blandet.sort()  
blandet
```

```
['1', '2', '7', 'Ida', 'Peter']
```



# SORTERING AF LISTER

```
venner = ['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']  
venner.sort()  
venner
```

['Anne', 'Bo', 'Eva', 'Mette', 'Peter', 'Signe', 'Søren']

```
tal = [3, 7, 1, 5, 9, 2]  
tal.sort(reverse = True)  
tal
```

[9, 7, 5, 3, 2, 1]

```
blandet = [1, 'Peter', 7, 'Ida', 2]  
blandet.sort()  
blandet
```

```
blandet = ['1', 'Peter', '7', 'Ida', '2']  
blandet.sort()  
blandet
```

['1', '2', '7', 'Ida', 'Peter']

ASCII-tabellen

# SORTERING AF LISTER

```
venner = ['Anne', 'Peter', 'Signe', 'Eva', 'Søren', 'Mette', 'Bo']
venner.sort()
venner
```

['Anne', 'Bo', 'Eva', 'Mette', 'Peter', 'Signe', 'Søren']

tal =	0	NUL	16	DLE	32		48	0	64	@	80	P	96	`	112	p
tal.so	1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
tal	2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
[9, 7,	3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
	4	EOT	20	DC4	36	\$	52	4	68	D	84	T	100	d	116	t
	5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
blande	6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
blande	7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
blande	8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
	9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
	10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
blande	11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
blande	12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	
blande	13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
['1',	14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
	15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL



# KOPIERING AF LISTER

---

```
mineVenner = ['Anne', 'Peter', 'Bo']
```

# KOPIERING AF LISTER

---

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner
```

# KOPIERING AF LISTER

---

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])
```

# KOPIERING AF LISTER

---

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

# KOPIERING AF LISTER

---

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```


```
Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']
```

# KOPIERING AF LISTER

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']

mineVenner



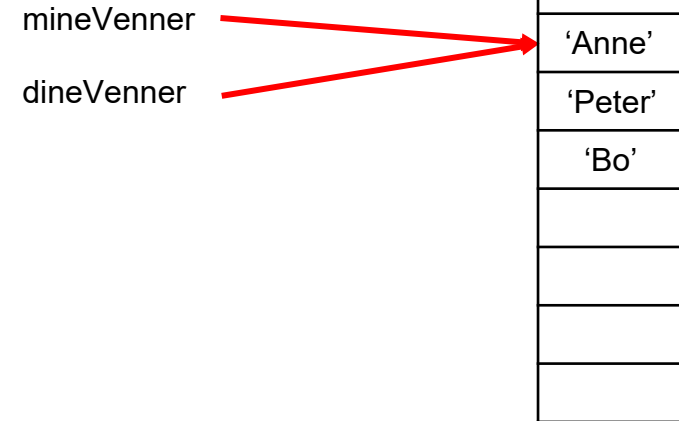
'Anne'
'Peter'
'Bo'



# KOPIERING AF LISTER

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']



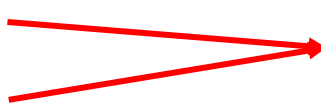
# KOPIERING AF LISTER

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']

mineVenner

dineVenner



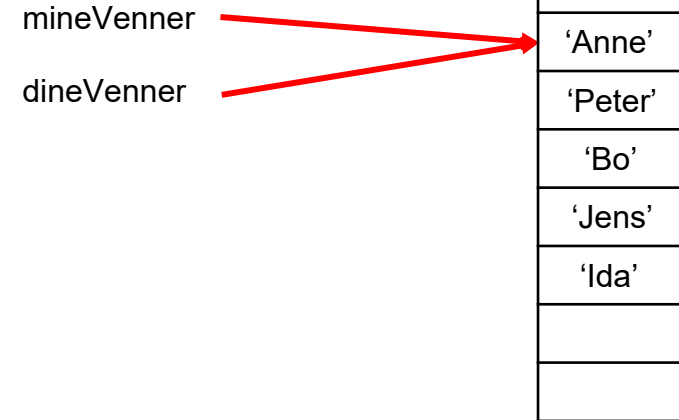
'Anne'
'Peter'
'Bo'
'Jens'
'Ida'

# KOPIERING AF LISTER

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

```
Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']
```

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner.copy()  
mineVenner.extend(['Jens', 'Ida'])
```

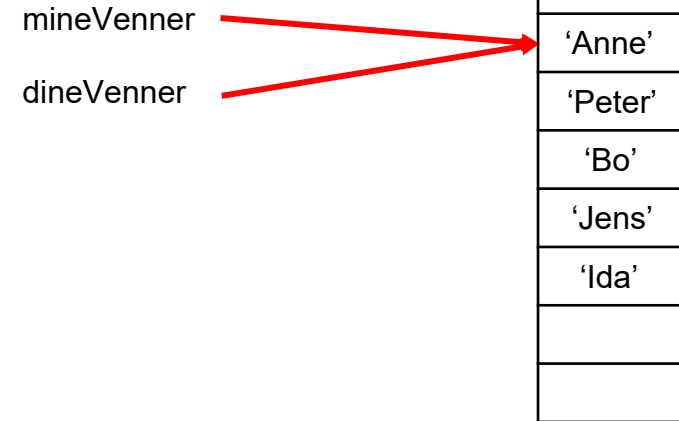


# KOPIERING AF LISTER

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

```
Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']
```

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner.copy()  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```



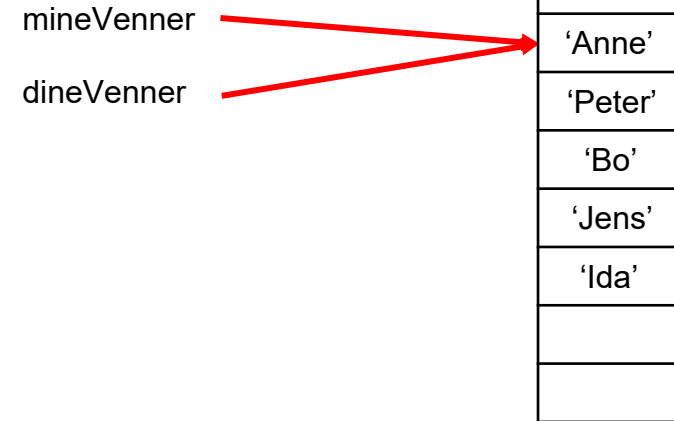
# KOPIERING AF LISTER

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

```
Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']
```

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner.copy()  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

```
Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo']
```



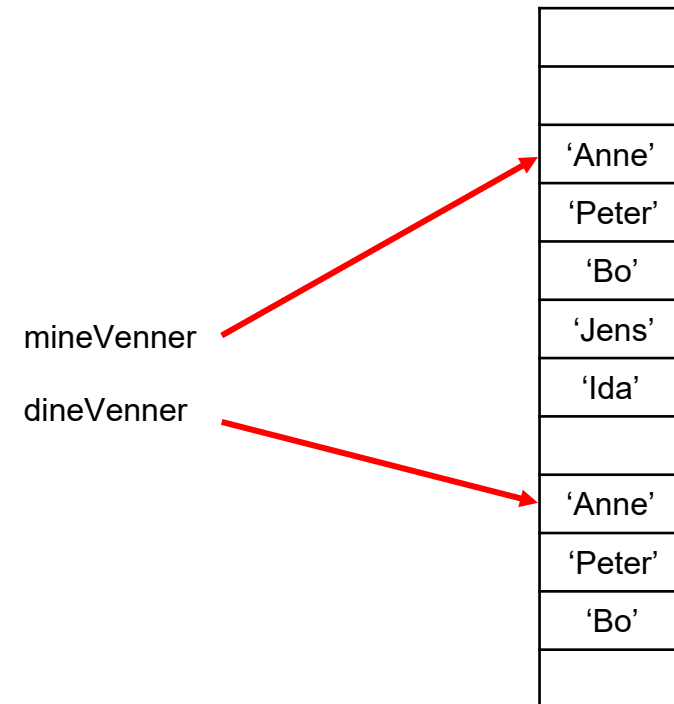
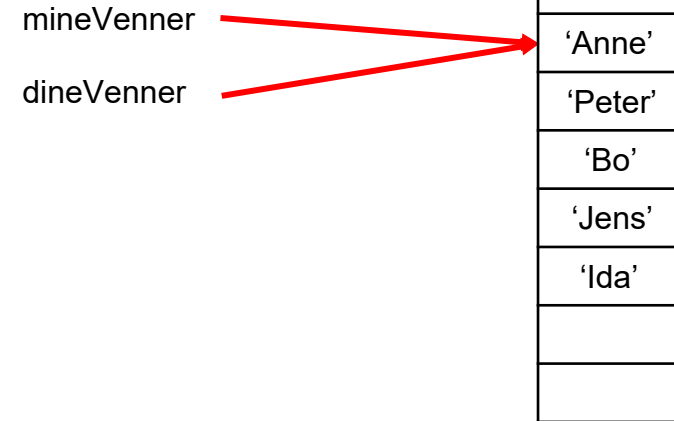
# KOPIERING AF LISTER

```
mineVenner = ['Anne', 'Peter', 'Bo']
dineVenner = mineVenner
mineVenner.extend(['Jens', 'Ida'])
print('Mine venner: ', mineVenner)
print('Dine venner: ', dineVenner)
```

Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']

```
mineVenner = ['Anne', 'Peter', 'Bo']
dineVenner = mineVenner.copy()
mineVenner.extend(['Jens', 'Ida'])
print('Mine venner: ', mineVenner)
print('Dine venner: ', dineVenner)
```

Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo']



# KOPIERING AF LISTER

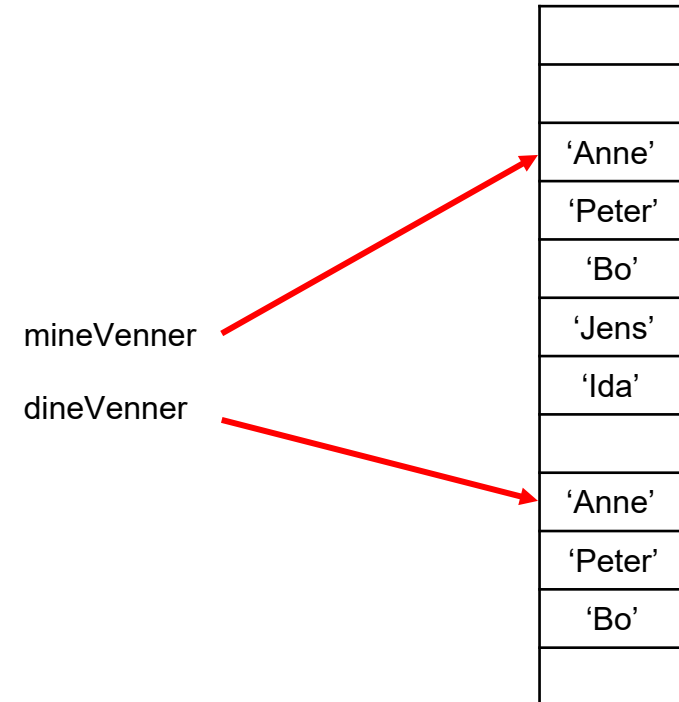
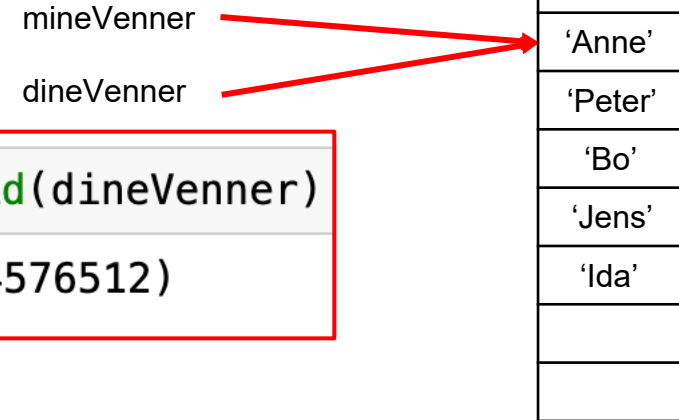
```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

```
id(mineVenner), id(dineVenner)  
(4424576512, 4424576512)
```

```
Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']
```

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner.copy()  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

```
Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo']
```



# KOPIERING AF LISTER

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

```
id(mineVenner), id(dineVenner)  
(4424576512, 4424576512)
```

```
Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']
```

```
mineVenner = ['Anne', 'Peter', 'Bo']  
dineVenner = mineVenner.copy()  
mineVenner.extend(['Jens', 'Ida'])  
print('Mine venner: ', mineVenner)  
print('Dine venner: ', dineVenner)
```

```
id(mineVenner), id(dineVenner)  
(4424508288, 4424573952)
```

```
Mine venner: ['Anne', 'Peter', 'Bo', 'Jens', 'Ida']  
Dine venner: ['Anne', 'Peter', 'Bo']
```

mineVenner

dineVenner

'Anne'
'Peter'
'Bo'
'Jens'
'Ida'

mineVenner

dineVenner

'Anne'
'Peter'
'Bo'
'Jens'
'Ida'
'Anne'
'Peter'
'Bo'



# METODER LISTER

---

Metode	Beskrivelse
<code>append( )</code>	Tilføjer et element i slutningen af listen
<code>clear( )</code>	Fjerner alle elementer fra liste
<code>copy( )</code>	Returnerer en kopi af listen
<code>count( )</code>	Returnerer antallet af elementer med den specificerede værdi
<code>extend( )</code>	Tilføjer elementer fra en liste til slutningen af listen
<code>index( )</code>	Returnerer indekset af det første element med den specificerede værdi
<code>insert( )</code>	Tilføjer et element i den specificerede position
<code>pop( )</code>	Fjerner elementet i den specificerede position
<code>remove( )</code>	Fjerne elementet med den specificerede værdi
<code>sort( )</code>	Sorterer listen

# LISTER VS VEKTORER

---

Lister ligner meget vektorer, som bruges i matematik, men...

$$[2, 3, 7] + [1, 8, 4]$$

# LISTER VS VEKTORER

---

Lister ligner meget vektorer, som bruges i matematik, men...

```
[2, 3, 7] + [1, 8, 4]
```

```
[2, 3, 7, 1, 8, 4]
```

# LISTER VS VEKTORER

---

Lister ligner meget vektorer, som bruges i matematik, men...

```
[2, 3, 7] + [1, 8, 4]
```

```
[2, 3, 7, 1, 8, 4]
```

```
3 * [5, 1]
```

```
[5, 1, 5, 1, 5, 1]
```

# LISTER VS VEKTORER

---

Lister ligner meget vektorer, som bruges i matematik, men...

```
[2, 3, 7] + [1, 8, 4]
```

```
[2, 3, 7, 1, 8, 4]
```

```
3 * [5, 1]
```

```
[5, 1, 5, 1, 5, 1]
```

```
import numpy as np
```

```
np.array([2, 3, 7]) + np.array([1, 8, 4])
```

# LISTER VS VEKTORER

---

Lister ligner meget vektorer, som bruges i matematik, men...

```
[2, 3, 7] + [1, 8, 4]
```

```
[2, 3, 7, 1, 8, 4]
```

```
3 * [5, 1]
```

```
[5, 1, 5, 1, 5, 1]
```

```
import numpy as np
```

```
np.array([2, 3, 7]) + np.array([1, 8, 4])
```

```
array([ 3, 11, 11])
```

# LISTER VS VEKTORER

---

Lister ligner meget vektorer, som bruges i matematik, men...

```
[2, 3, 7] + [1, 8, 4]
```

```
[2, 3, 7, 1, 8, 4]
```

```
3 * [5, 1]
```

```
[5, 1, 5, 1, 5, 1]
```

```
import numpy as np
```

```
np.array([2, 3, 7]) + np.array([1, 8, 4])
```

```
array([ 3, 11, 11])
```

Lister skal laves om til såkaldte arrays for at kunne virke som matematiske vektorer

# TUPLER

---

Datatypen tuple er en følge af endeligt mange elementer med en bestemt rækkefølge



# TUPLER

---

Datatypen tuple er en følge af endeligt mange elementer med en bestemt rækkefølge

Svarer til datatypen list – der er dog vigtige forskelle

# TUPLER

---

Datatypen tuple er en følge af endeligt mange elementer med en bestemt rækkefølge

Svarer til datatypen list—der er dog vigtige forskelle

```
('Æble', 'Banan', 'Appelsin')  
  
(1, 3, 7, 12, 19)  
  
(2, 'bold', 17, True, 5, (2, 4, 6))
```

# TUPLER

---

Datatypen tuple er en følge af endeligt mange elementer med en bestemt rækkefølge

Svarer til datatypen list—der er dog vigtige forskelle

```
('Æble', 'Banan', 'Appelsin')  
(1, 3, 7, 12, 19)  
(2, 'bold', 17, True, 5, (2, 4, 6))
```

bemærk de almindelige parenteser og  
kommaerne, der adskiller elementerne

# TUPLER

Datatypen tuple er en følge af endeligt mange elementer med en bestemt rækkefølge

Svarer til datatypen list—der er dog vigtige forskelle

```
('Æble', 'Banan', 'Appelsin')  
(1, 3, 7, 12, 19)  
(2, 'bold', 17, True, 5, (2, 4, 6))
```

bemærk de almindelige parenteser og  
kommaerne, der adskiller elementerne

Generelt

(element<sub>1</sub>, element<sub>2</sub> ... element<sub>n</sub>)

hvor elementerne kan være vilkårlige datatyper

# TUPLER

---

Datatypen tuple er en følge af endeligt mange elementer med en bestemt rækkefølge

Svarer til datatypen list – der er dog vigtige forskelle

# TUPLER

---

Datatypen tuple er en følge af endeligt mange elementer med en bestemt rækkefølge

Svarer til datatypen list—der er dog vigtige forskelle

men først lighederne:

- indeksering (nul-indeksering og negativ indeksering)

- beskæring

- gennemløb

# TUPLER

---

Datatypen tuple er en følge af endeligt mange elementer med en bestemt rækkefølge

Svarer til datatypen list—der er dog vigtige forskelle

men først lighederne:

- indeksering (nul-indeksering og negativ indeksering)

- beskæring

- gennemløb

forskellene:

- almindelige parenteser fremfor firkantede parenteser

# TUPLER

---

Datatypen tuple er en følge af endeligt mange elementer med en bestemt rækkefølge

Svarer til datatypen list—der er dog vigtige forskelle

men først lighederne:

- indeksering (nul-indeksering og negativ indeksering)

- beskæring

- gennemløb

forskellene:

- almindelige parenteser fremfor firkantede parenteser

- kan ikke ændres

```
venner = ('Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette')  
venner[3] = 'Signe'
```



# TUPLER

---

Datatypen tuple er en følge af endeligt mange elementer med en bestemt rækkefølge

Svarer til datatypen list—der er dog vigtige forskelle

men først lighederne:

indeksering (nul-indeksering og negativ indeksering)

beskæring

gennemløb

forskellene:

almindelige parenteser fremfor firkantede parenteser

kan ikke ændres

```
venner = ('Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette')
venner[3] = 'Signe'
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[8], line 2
      1 venner = ('Anne', 'Peter', 'Ole', 'Ida', 'Eva', 'Søren', 'Mette')
----> 2 venner[3] = 'Signe'
```

```
TypeError: 'tuple' object does not support item assignment
```

# METODER TUPLER

---

Metode	Beskrivelse
count( )	Returnerer antallet af elementer med den specificerede værdi
index( )	Returnerer indekset af det første element med den specificerede værdi

# METODER TUPLER

---

Metode	Beskrivelse
<code>count( )</code>	Returnerer antallet af elementer med den specificerede værdi
<code>index( )</code>	Returnerer indekset af det første element med den specificerede værdi

Hvorfor kan det alligevel være en god ide at bruge tupler fremfor lister?

# METODER TUPLER

---

Metode	Beskrivelse
<code>count( )</code>	Returnerer antallet af elementer med den specificerede værdi
<code>index( )</code>	Returnerer indekset af det første element med den specificerede værdi

Hvorfor kan det alligevel være en god ide at bruge tupler fremfor lister?

- tupler bruger mindre lager og er derfor hurtigere at tilgå

# METODER TUPLER

---

Metode	Beskrivelse
<code>count( )</code>	Returnerer antallet af elementer med den specificerede værdi
<code>index( )</code>	Returnerer indekset af det første element med den specificerede værdi

Hvorfor kan det alligevel være en god ide at bruge tupler fremfor lister?

- tupler bruger mindre lager og er derfor hurtigere at tilgå
- for at undgå utilsigtede ændringer i data

# LISTER I SCRATCH

Under **Variable** findes **Lav en liste**

## Variable

Lav en variabel

☐

my variable

sæt my variable ▼ til 0

ændre my variable ▼ med 1

vis variabel my variable ▼

skjul variabel my variable ▼

Lav en liste

## Mine brikker

Lav en brik

# LISTER I SCRATCH

Under **Variable** findes **Lav en liste**  
hvorefter man kan give listen et navn, fx **Venner**  
og tilføje elementer ved at klikke på **+**



# LISTER I SCRATCH

---

Under **Variable** findes **Lav en liste**  
hvorefter man kan give listen et navn, fx **Venner**  
og tilføje elementer ved at klikke på **+**

Bemærk, at lister i Scratch ikke er nulindekserede





# LISTER I SCRATCH

Under **Variable** findes **Lav en liste**  
hvorefter man kan give listen et navn, fx **Venner**  
og tilføje elementer ved at klikke på **+**

Bemærk, at lister i Scratch ikke er nulindekserede

Der opstår en række forskellige muligheder, når  
der er oprettet en liste

