# Forelesning IN1900 – 5 September 2023

Ole Christian Lingjærde
Institutt for Informatikk, Universitetet i Oslo

Uke: 4 September – 10 September, 2023

# Kort om min bakgrunn

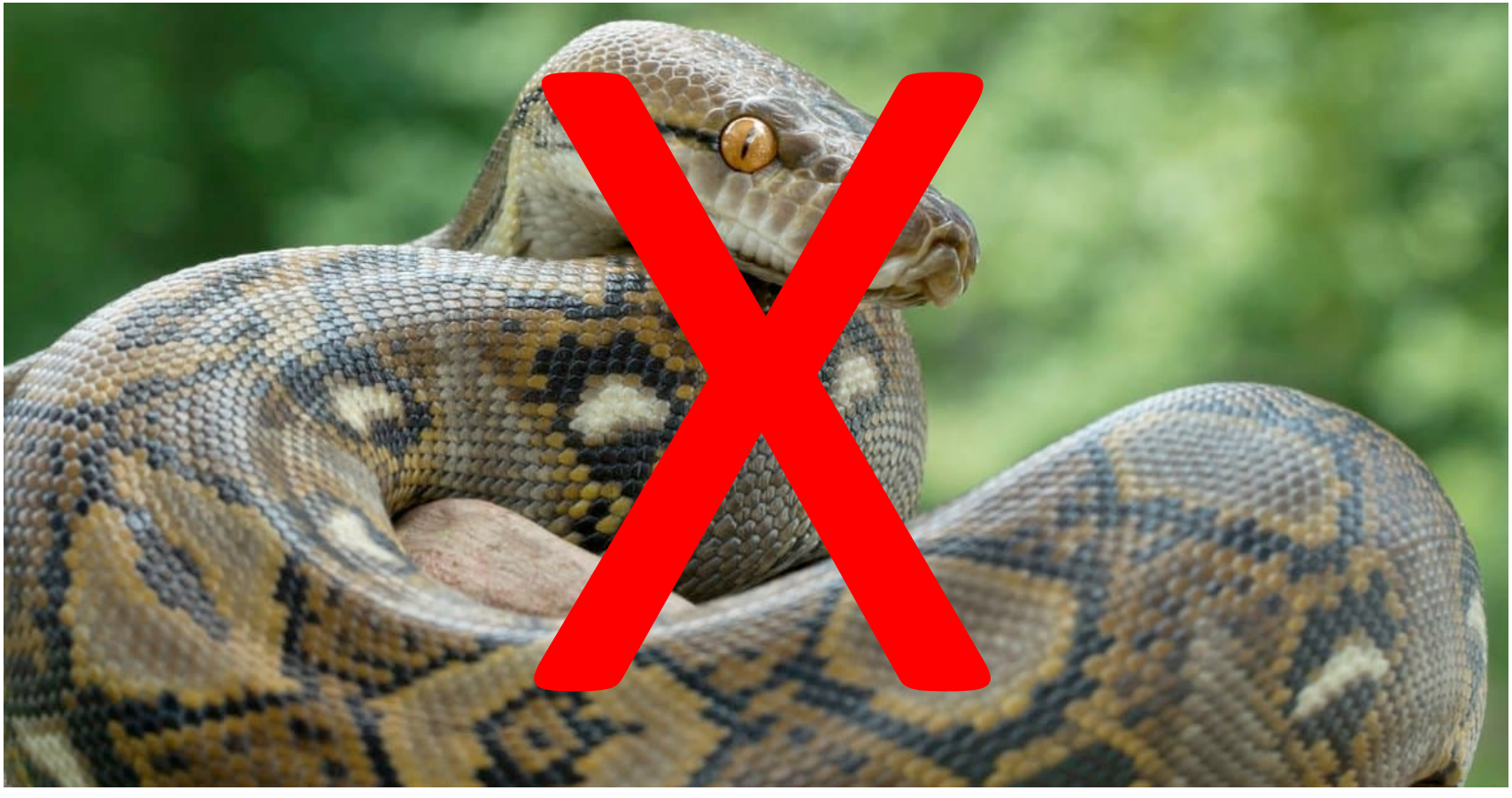
- Doktorgrad i anvendt matematikk fra Ifi, UiO
- Professor i maskinlæring og kunstig intelligens (AI) ved Ifi
- Også tilknyttet Oslo Universitetssykehus for å utvikle diagnoseverktøy for kreft basert på DNA-data
- Undervist i programmering i > 15 år

# Tema i dag

- Mer om lister
- Tupler (= lister som ikke kan endres)
- Funksjonen zip
- Lister av lister (tabeller)
- Mange eksempler på bruk av løkker og lister

# Hvorfor Python?

# Ikke slangen....



# ... men komikergruppen Monthy Python

# Noen fakta om Python

- Offisielt språk i Google (sammen med C++ og Java)
- Åpen kildekode og helt gratis
- Populært språk for maskinlæring/AI/big data
- Mange varianter av Python for spesielle formål:

| | |
|---|---|
| CPython | offisiell implementasjon av Python |
| PyPy | alternativ implementasjon for økt ytelse |
| Jython | for integrasjon med Java-kode |
| IronPython | for integrasjon med .NET |
| Brython/Skulpt | for å kjøre i nettlesere |
| MicroPython | for "embedded systems" |

# Lister i Python

Variabler har plass til en enkelt verdi:

```
C = -10
x = 2.255
s = 'Hello'
```

Lister kan holde flere verdier:

```
C = [-10, -5, 0, 5, 10]
x = [2.255, 3.634, 6.4]
s = ['Hello', 'my', 'friend']
u = [3, 3.14, 'pi']
```

Denne enkle utvidelsen fra en til mange verdier er utrolig nyttig og åpner en hel verden av nye muligheter!

# Konstruksjon av lister

Lister brukes veldig ofte i Python.

Ting vi kan gjøre med lister (og må lære):

- Lage dem
- Hente ut verdier fra dem (alle eller noen)
- Lete etter verdier i dem
- Forandre innholdet i dem
- Løpe systematisk gjennom dem

**Live programmering: lage lister**

# Oversikt over metoder for å lage lister

## Metode A: Angi listeelementer eksplisitt

```
a = [2, 3, 5, 7, 11, 13]
minliste = ['Kurs', 'i', 'programmering']
```

## Metode B: Angi et intervall

```
# Angi stopp:
a = list(range(10))                    # [0, 1, 2, ...., 9]

# Angi start og stopp:
a = list(range(3, 10))                 # [3, 4, 5, ...., 9]

# Angi start, stopp og steglengde:
a = list(range(3, 10, 2))              # [3, 5, 7, 9]

# Kan droppe list(..) hvis eneste bruk er for-løkke:
for i in range(10):
   print(i)                            # Skriver ut tallene 0-9
```

# Oversikt over metoder (forts.)

## Metode C: Angi startverdi og antall repetisjoner

```
a = [0]*10          # [0, 0, ..., 0]  (10 elementer)
b = [1]*25          # [1, 1, ..., 1] (25 elementer)
c = ['Hei']*3       #['Hei', 'Hei', 'Hei']
d = [0,1,2]*2       #[0, 1, 2, 0, 1, 2]
```

## Metode D: Angi en regel for å generere verdier

```
a = [i*i for i in range(5)]                  # [0, 1, 4, 9, 16]
b = [10-2*j for j in range(5)]               # [10, 8, 6, 4, 2]
c = [e**2 for e in a]                        # [0, 1, 16, 81, 256]
d = [e**2 for e in a if e%2==0]              # [0, 16, 256]
e = [s[0] for s in ['Hei', 'paa', 'deg']]    # ['H', 'p', 'd']
```

(Merk: e%2==0 tester om e er et partall)

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [1,2,3,4]
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [1,2,3,4]
```

**Svar:**

```
a = [1, 2, 3, 4]
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [-2, -2, -2, ...., -2]
```

(listen skal ha lengde 100)

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [-2, -2, -2, ...., -2]
```

(listen skal ha lengde 100)

**Svar:**

```
a = [-2] * 100
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [1, 2, 3, ..., 9999]
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [1, 2, 3, ..., 9999]
```

**Svar:**

```
a = list(range(1,10000))
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [2, 4, 6, 8, ...., 2000]
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [2, 4, 6, 8, ...., 2000]
```

**Svar:**

```
a = list(range(2,2001,2))
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [999, 998, 997, ..., 1]
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [999, 998, 997, ..., 1]
```

**Svar:**

```
a = list(range(999,0,-1))
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [0, 1, 0, 1, ....., 0, 1]
```

(listen skal ha lengde 100)

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [0, 1, 0, 1, ....., 0, 1]
```

(listen skal ha lengde 100)

**Svar:**

```
a = [0,1] * 50
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [1*2, 2*3, 3*4, 4*5, ..., 999*1000]
```

# Quiz

Foreslå en måte å lage denne listen på i Python:

```
a = [1*2, 2*3, 3*4, 4*5, ..., 999*1000]
```

**Svar:**

```
a = [i*(i+1) for i in range(1,1000)]
```

# Quiz

Hva blir resultatet?

```
a = [-1]*3 + [0]*3
```

# Quiz

Hva blir resultatet?

```
a = [-1]*3 + [0]*3
```

**Svar:**

```
a : [-1, -1, -1, 0, 0, 0]
```

# Quiz

Hva blir resultatet?

```
b = list(range(-1,4))
```

# Quiz

Hva blir resultatet?

```
b = list(range(-1,4))
```

**Svar:**

```
b : [-1, 0, 1, 2, 3]
```

# Quiz

Hva blir resultatet?

```
c = [k-1 for k in [1,2,3]]
```

# Quiz

Hva blir resultatet?

```
c = [k-1 for k in [1,2,3]]
```

**Svar:**

```
c : [0, 1, 2]
```

# Quiz

Hva blir resultatet?

```
d = [k**3 for k in range(1,4)]
```

# Quiz

Hva blir resultatet?

```
d = [k**3 for k in range(1,4)]
```

**Svar:**

```
d : [1, 8, 27]
```

# Quiz

Hva blir resultatet?

```
e = [1, 3, 5]
f = [e[i-1]+e[i] for i in range(0,3)]
```

# Quiz

Hva blir resultatet?

```
e = [1, 3, 5]
f = [e[i-1]+e[i] for i in range(0,3)]
```

**Svar:**

```
f : [6, 4, 8]
```

# Hente ut enkeltverdier i en liste

Vi kan hente ut enkeltverdier i en liste:

```
a = [3.14, 3.1415926, 999999]
print(len(a))
print(a[0])
print(a[1])
print(a[2])
```

```
3
3.14
3.1415926
999999
```

# Hente ut verdier fra slutten av en liste

Vi kan hente ut verdier fra slutten av en liste:

```
a = [-10, -5, 0, 5, 10]
print(a[-1])          # Siste element
print(a[-2])          # Nest siste element

somelist = ['eple', 'appelsin', 'ananas']
a, b, c = somelist  # Tilordne direkte til variabler
print(a)
print(b)
print(c)
```

```
10
5
eple
appelsin
ananas
```

# Lete etter verdier i en liste

Vi kan lete etter en bestemt verdi i en liste:

```
a = [-10, -5, 0, 5, 10]
print(a.index(10))          # Hvor ligger verdien 10?
print(5 in a)               # Finnes verdien 5 i listen?

b = [1, 2, 1]
print(b.index(1))           # Bare treff nr 1 rapporteres
print(b.count(1))           # Hvor mange ganger fins verdien 1 i listen?

k1 = b.index(1)             # Treff nr 1
k2 = b.index(1, k1+1)       # Treff nr 2
print(k1, k2)
```

Start å lete i posisjon k1+1 ( = 1)

```
4
True
0
2
0 2
```

# Forandre innholdet i en liste

Vi kan endre enkeltelementer i en liste:

```
a = [3.14, 3.1415926, 999999]
a[1] = 0.1
print(a)
```

```
[3.14, 0.1, 999999]
```

# Forandre innholdet i en liste (2)

Vi kan også utvide eller forkorte en liste:

```
a = [-10, -5, 0, 5, 10]
a.append(99)                # Legg til et nytt element på slutten
print(a)

a = a + [100, 200]          # Slå sammen (engelsk: concatenate) to lister
print(a)

a.insert(2, -99)            # Sett inn -99 slik at den får indeks 2
print(a)

del a[2]                    # Fjern elementet med indeks 2
print(a)
```

```
[-10, -5, 0, 5, 10, 99]
[-10, -5, 0, 5, 10, 99, 100, 200]
[-10, -5, -99, 0, 5, 10, 99, 100, 200]
[-10, -5, 0, 5, 10, 99, 100, 200]
```

# Løpe systematisk gjennom en liste

Vi kan bruke en for-løkke til å løpe gjennom alle verdiene i en liste:

```
CListe = [-20, -15, -10, 5, 0]
for C in CListe:
    F = (9.0/5) * C + 32
    print(f"{C:3.1f}  {F:3.1f}")
```

```
-20.0  -4.0
-15.0  5.0
-10.0  14.0
5.0  41.0
0.0  32.0
```

# Quiz

Lag en for-løkke som skriver ut tallene

1, 2, 4, 8, 16, 32, 64, 128, 256, 512

ved å bruke formelen $x_i = 2^i$ for $i = 0, 1, 2, \ldots, 9$.

# Quiz (svar)

```
for i in [0,1,2,3,4,5,6,7,8,9]:
    xi = 2**i
    print(xi)
```

Alternativ løsning:

```
for i in range(10):
    xi = 2**i
    print(xi)
```

# Beregninger av følger

Løkker kan brukes til å regne ut følger, f.eks. Fibonacci-følgen:

$$F_0 = 1$$
$$F_1 = 1$$
$$F_k = F_{k-2} + F_{k-1} \qquad k = 2, 3, \ldots$$

Vi regner ut de første leddene:

$F_0 = 1$
$F_1 = 1$
$F_2 = 1 + 1 = 2$
$F_3 = 1 + 2 = 3$
$F_4 = 2 + 3 = 5$
$F_5 = 3 + 5 = 8$
...

Live programmering: Fibonacci-følgen

# Bruk av løkke for å beregne Fibonacci-rekken

## Løsning A:

```
F = [1, 1]
for k in range(2, 100):
  F.append(F[k-1] + F[k-2])
print(F)
```

## Løsning B:

```
F = [1]*100
for k in range(2, 100):
   F[k] = F[k-2] + F[k-1]
print(F)
```

Spørsmål: Løsning A endrer størrelsen på listen F underveis. Kan du se noen potensielle ulemper med det?

# Beregning av summer (rekker)

Den harmoniske rekken (med 100 ledd) er:

$$s = \sum_{i=1}^{100} \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{100}$$

Denne kan vi lett beregne med en løkke.

Live programmering: harmonisk rekke

# Flere alternative løsninger

## Alternativ A: while-løkke

```
s = 0    # Startverdi
i = 1    # Teller
while i <= 100:
    s += 1/i
    i = i + 1
print(f"s = {s:5.2f}")
```

## Alternativ B: for-løkke

```
s = 0
for i in range(1, 101):
    s += 1/i
print(f"s = {s:5.2f}")
```

## Alternativ C: implisitt løkke

```
s = sum(1/i for i in range(1,101))
print(f"s = {s:5.2f}")
```

# Hente ut mange verdier fra en liste

Vi kan plukke ut mange verdier samtidig fra en liste:

```
# Lag en liste
a = [5, 10, 15, 20, 25, 30]

# Plukke ut første del av listen:
b = a[:4]       # [5, 10, 15, 20]  (indeks 0-3)

# Plukke ut siste del av listen:
c = a[3:]       # [20, 25, 30]  (indeks 3-5)

# Plukke ut midtparti av listen:
d = a[3:5]      # [20, 25]  (indeks 3-4)

# Alle elementer unntatt første og siste:
e = a[1:-1]     # (indeks 1-4)

# Alle elementer unntatt tre siste:
f = a[:-3]      # (indeks 0-2)

# Lage en kopi av hele listen:
g = a[:]        # (indeks 0-5)
```

# Quiz

Hva blir resultatet i hvert av disse tilfellene?

```
a = [0, 2, 4, 6, 8, 10, 12, 14]

x = a[3]
# Svar:

x = a[3:4]
# Svar:

x = a[3:3]
# Svar:

x = a[0:2] + a[3:5]
# Svar:

x = [2*e for e in a[0:2]]
# Svar:

x = a[-2:2]
# Svar:
```

# Quiz (svar)

Hva blir resultatet i hvert av disse tilfellene?

```
a = [0, 2, 4, 6, 8, 10, 12, 14]

x = a[3]
# Svar: x = 6

x = a[3:4]
# Svar:

x = a[3:3]
# Svar:

x = a[0:2] + a[3:5]
# Svar:

x = [2*e for e in a[0:2]]
# Svar:

x = a[-2:2]
# Svar:
```

# Quiz (svar)

Hva blir resultatet i hvert av disse tilfellene?

```
a = [0, 2, 4, 6, 8, 10, 12, 14]

x = a[3]
# Svar: x = 6

x = a[3:4]
# Svar: x = [6]

x = a[3:3]
# Svar:

x = a[0:2] + a[3:5]
# Svar:

x = [2*e for e in a[0:2]]
# Svar:

x = a[-2:2]
# Svar:
```

# Quiz (svar)

Hva blir resultatet i hvert av disse tilfellene?

```
a = [0, 2, 4, 6, 8, 10, 12, 14]

x = a[3]
# Svar: x = 6

x = a[3:4]
# Svar: x = [6]

x = a[3:3]
# Svar: x = []

x = a[0:2] + a[3:5]
# Svar:

x = [2*e for e in a[0:2]]
# Svar:

x = a[-2:2]
# Svar:
```

# Quiz (svar)

Hva blir resultatet i hvert av disse tilfellene?

```
a = [0, 2, 4, 6, 8, 10, 12, 14]

x = a[3]
# Svar: x = 6

x = a[3:4]
# Svar: x = [6]

x = a[3:3]
# Svar: x = []

x = a[0:2] + a[3:5]
# Svar: x = [0, 2, 6, 8]

x = [2*e for e in a[0:2]]
# Svar:

x = a[-2:2]
# Svar:
```

# Quiz (svar)

Hva blir resultatet i hvert av disse tilfellene?

```
a = [0, 2, 4, 6, 8, 10, 12, 14]

x = a[3]
# Svar: x = 6

x = a[3:4]
# Svar: x = [6]

x = a[3:3]
# Svar: x = []

x = a[0:2] + a[3:5]
# Svar: x = [0, 2, 6, 8]

x = [2*e for e in a[0:2]]
# Svar: x = [0, 4]

x = a[-2:2]
# Svar:
```

# Quiz (svar)

Hva blir resultatet i hvert av disse tilfellene?

```
a = [0, 2, 4, 6, 8, 10, 12, 14]

x = a[3]
# Svar: x = 6

x = a[3:4]
# Svar: x = [6]

x = a[3:3]
# Svar: x = []

x = a[0:2] + a[3:5]
# Svar: x = [0, 2, 6, 8]

x = [2*e for e in a[0:2]]
# Svar: x = [0, 4]

x = a[-2:2]
# Svar: x = []
```

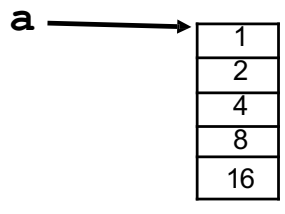
# En liste kan ha flere navn

Det neste vi skal se på er litt vanskelig, og ikke regn med at du skjønner det 100% nå i starten!

Når vi lager en liste må vi skille mellom **selve listen** (ofte kalt liste-objektet) og **navnet** vi bruker for å referere til listen. En liste kan faktisk ha flere navn samtidig!

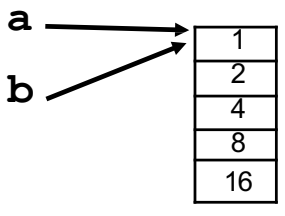
**Etter at vi har utført.....** **så er situasjonen slik:**

```
a = [1, 2, 4, 8, 16]
```



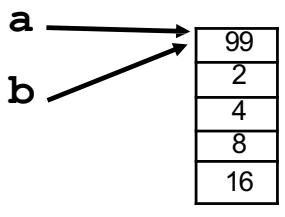

Ett dataobjekt, og ett navn a

```
b = a
```




Ett dataobjekt, og to navn a og b

```
a[0] = 99
```




Dataobjektet endres og både a og b «ser» denne endringen

# Hva betyr dette i praksis?

**Når en liste i Python har flere navn, må vi huske på at disse refererer til den *samme listen*.** Eksempel:

```
# Vi lager en liste med tre verdier
a = [1,2,4]

# Vi gir listen et ekstra navn:
b = a

# Vi endrer første verdien i b:
b[0] = 99

# Vi skriver ut innholdet av a og b:
print(f"a={a} og b={b}")
```

```
a=[99, 2, 4] og b=[99, 2, 4]
```

**Både a og b har blitt endret!**

# Å lage kopi av en liste

Det går an å lage en kopi av en liste - altså av selve listeobjektet. Da kan vi gjøre en endring i én av listene, uten at det forandrer den andre.

```
a = [1, 2, 4]
b = a[:]        ## Lag kopi av a
b[0] = 99       ## Nå endres b[0], men ikke a[0]
print(a)
print(b)

[1, 2, 4]
[99, 2, 4]
```