INF100 Krasjkurs

...

How to: Eksamen i INF100

•••

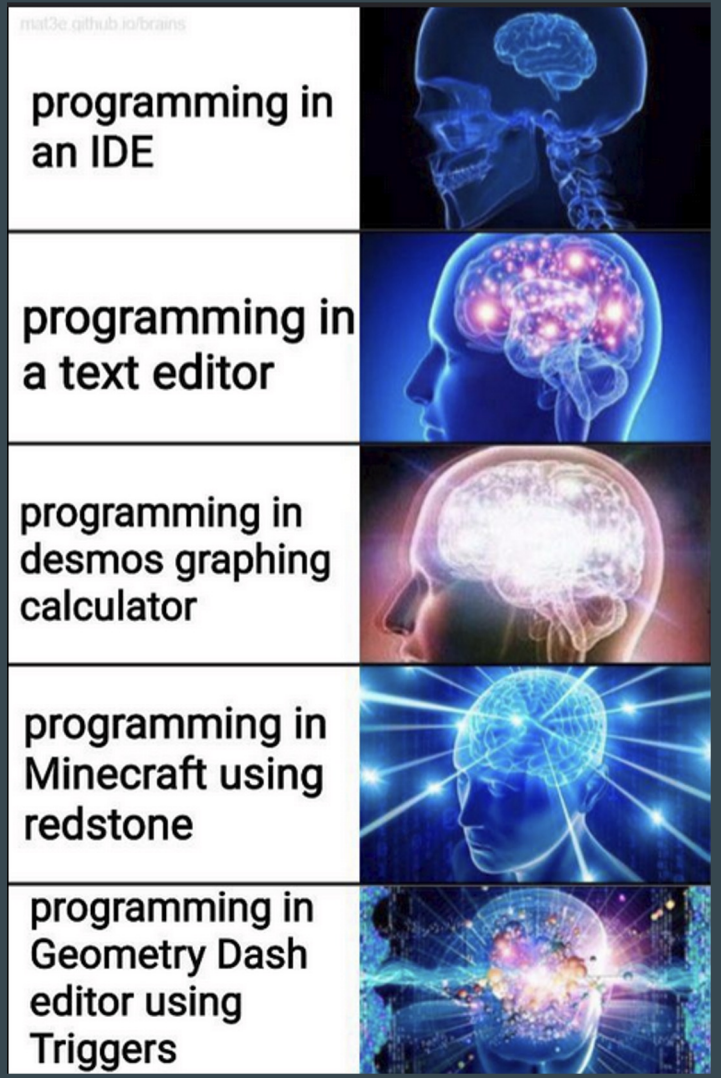
med Kristian

Om eksamen i INF100

- Ingen mulighet for å kjøre kode
- Syntax-feil og slurvefeil?
- Sensor kommer til å lete etter hva du har forstått
- Hva slags oppgaver kan dere forvente?

Tips og triks: Før eksamen

- Øv på å skrive kode uten hjelp fra vs code
- Gjør deg kjent med forelesningsnotatene
  - Har tilgang til disse under eksamen!
- Gjør oppgaver -> Les deg opp om du står fast
- GJØR GAMLE EKSAMENSOPPGAVER!!



Tips og triks: Under eksamen

- Hvis du står fast -> gå videre
- Gjør alle oppgavene
  - Prøv ditt beste
  - Skriv kommentarer
  - Skriv pseudokode
- Kjør koden i hodet ditt

Typer og variabler

...

med Kristian

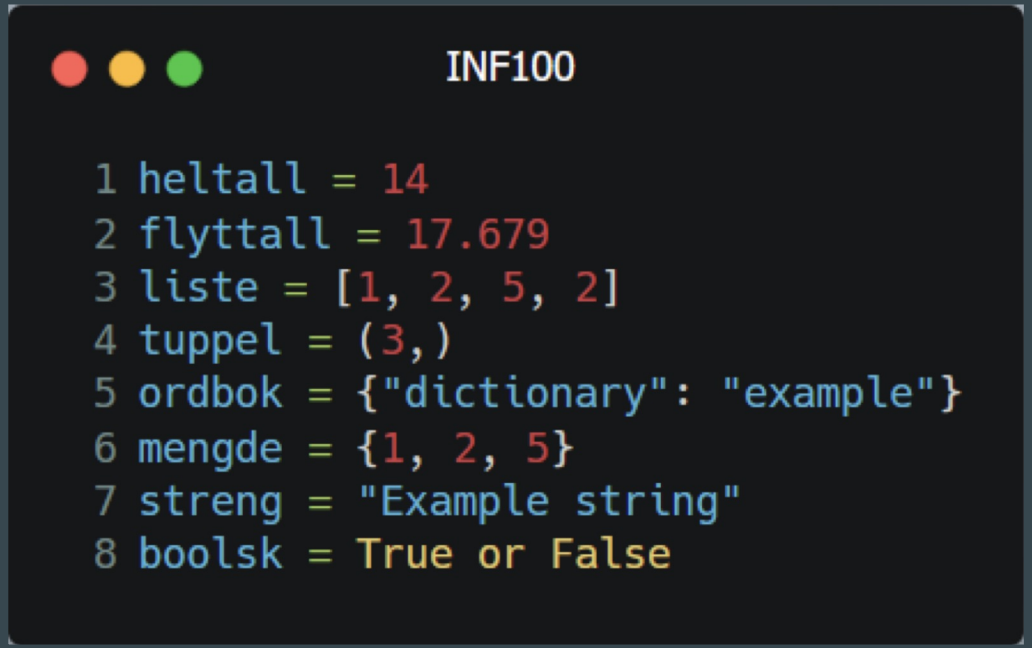
Variabler

- Holder en verdi
- Tildeling av et navn til en kjent eller ukjent informasjon
- ‘=’ brukes for å tildele en verdi til en variabel

```
6    navn = "Kristian"
7    alder = 23
8    gruppeleder = True
```

Typer i python

- Numeriske typer
  - int
  - float
- Collections
  - list
  - tuple
  - set
  - dict
- bool
- str
- NoneType



Type conversion

```
heltall = 10
tekst = 'hei:)'
print(type(heltall))          # int
print(type(str(heltall)))     # str
print(type(float(heltall)))   # float
print(type(int(tekst)))       # ERROR
```

Eksamensoppgave

Variabler

```
a = 420

b = 'True'

c = ['hei', 10, [False]]

d = 6.9

e = 'Gleder meg til jul<333'

f = {
    'key1' : 2,
    '2' : True,
    3 : [4, 3.0]
    'f' : '2'
    }
```

Hvilken type får de følgende uttrykkene?

```
a

b

c[2]

a * b

e * d

c[2][0] == a

f[2]

f[3.0]

f[f['f']]
```

Fasit

Fasit

- int
- str
- list
- str
- Error
- bool
- Error
- list

Booleans & if statements

•••

med Izaak

Boolske verdier og uttrykk

Boolske verdier : True / False

Boolske uttrykk: Uttrykk som evaluerer til enten True eller False

Sammenligning: ==, !=, >, <, >=, <=

Logiske operatorer: and, or, not

python

```
5 > 3
10 == 10
7 != 2
4 < 2
```

python

```
(5 > 3) and (10 < 15)
(5 > 3) or (10 < 8)
not (5 == 5)
```

Presedens

Tenk PEMDAS men med boolske verdier

1. Parenteser
2. Sammenligning (in)
3. Not
4. And
5. Or

```
expression = True or not False and False
```

```
expression = True or True and False
```

```
expression = True or False
```

```
expression = True
```

PSNAO?

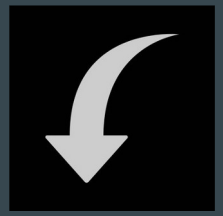
Hvordan plassere parenteser for å få et uttrykk *identisk* med

x and y or z in a

Velg ett alternativ:

- x and (y or (z in a))
- x and ((y or z) in a)
- (x and y) or (z in a)
- (x and (y or z)) in a
- ((x and y) or z) in a

Generalisert if setning



Programmet krasjer om det boolske uttrykket ikke kan evalueres til True/False

```
if <boolsk uttrykk>:
    <kodeblokk>
```

else?

```
weather
if weather == “sunny”:
        go_outside()
else:
        stay_inside()
```

```
if weather == “sunny”:
        go_swim()
elif weather == “cloudy”:
        go_walk()
else:
        stay_inside()
```

Kombinere boolske operatorer med if setninger

```
1   alder = 22
2   har_førerkort = True
3
4   if alder >= 18 and har_førerkort:
5       print("Du kan kjøre")
6   else:
7       print("Du kan ikke kjøre")
8
```

Løkker

•••

med Kristian

While - løkker

```
while condition:
    print("Utfør følgende kode.")
```

```
i = 0
while i < 100:
    if i % 2 == 0:
        print(i, " er et partall.")
    i += 1
```

```
i = 0
while True:
    if i % 2 == 0:
        print(i, " er et partall.")
    i += 1
    if 1 == 100:
        break
```

For - løkker

```
for variable in sequence:
    print("Utfør følgend kodeblokk.")
```

```
for i in range(10):
    print(i*10)
```

```
matvarer = ["tomat", "banan", "eple", "melk", "pasta", "pære", "apelsin"]
frukt = ["banan", "eple", "pære", "apelsin"]
for vare in matvarer:
    if vare in frukt:
        print(vare)
```

Finn feilen

```
def four_letter_capitals():
    capitals = ["London", "Paris", "Oslo", "Berlin", "Roma", "Kiev"]
    for i in capitals:
        if len(capitals[i]) < 5:
            print(capitals[i])
```

Finn feilen

```
def x_squared():
    x = 0
    while x < 10:
        squared = x ** 2
        print(squared)
        x + 1
```

Finn feilen

```
def make_multiplication_table(cols, rows):
    table = []
    for i in range(rows):
        row = []
        for j in range(cols):
            row.append(i * j)
    table.append(row)
    return table
```

Finn feilen

```
def remove_zeros():
    l = [0,0,5,6,2,1,0,1,0,4,5]
    for i in range(len(l)):
        if l[i] == 0:
            l.pop(i)
```

Funksjoner

med Izaak

Hva er funksjoner og hvorfor brukes de?

Funksjoner er kodeblokker vi kan lett bruke om igjen.
De hjelper å organisere koden, gjør den lettere å lese og lar oss unngå å repetere kode

```
1   def greet():
2       print("Hei og velkommen til kræsjkurs")
3
4   greet()
```

Parameter vs argument

Parameter!

```
6   def plus_one_times_two(number):
7       return (number + 1) * 2
8
9   tall = 5
10  oppdatert_tall = plus_one_times_two(tall)
11
12  print(oppdatert_tall) # 12
13
```

Argument!
(Parameteret "number" får tilegnet verdien 5, som er argumentet)

Return vs print

Print:

Print brukes for å VISE noe i terminalen

Return:

Return brukes for å lagre et resultat (fra funksjoner)

Bruk print når du vil vise noe på skjermen (terminal), bruk return i funksjoner for å returnere en verdi som du vil bruke et annet sted i programmet.

```
15
16  def lag_en_streng():
17      streng = "Dette er et eksemepel"
18      return streng
19
20  streng = lag_en_streng()
21  print(streng) # "Dette er et eksemepel"
22
```

```
22
23  def lag_en_streng_med_navn(navn):
24      streng = "Hei " + navn
25      return streng
26
27  streng = lag_en_streng_med_navn("Izaak")
28  print(streng) # Hei Izaak
29
```

OBS!

En funksjon kan kun returnere én gang på en kjøretid

Så snart datamaskinen leser en return-statement stoppes funksjonskallet

```
30  def count(l):
31      sum = 0
32      for row in l:
33          for number in row:
34              sum += number
35          return sum
36
37  l = [[1,2], [3,4]]
38  print(count(l)) # 3
```

```
40  def count(l):
41      sum = 0
42      for row in l:
43          for number in row:
44              sum += number
45      return sum
46
47  l = [[1,2], [3,4]]
48  print(count(l)) # 10
```

Lister og tupler

med Kristian

INF100

```python
liste = [4, 6, "potet", [2, 4]]

print(liste[-1])
print(liste[2:4:1])

liste.append([7,11])
liste.pop(-2)
print(liste)

nyliste = liste.remove("potet")
print(nyliste)

print(liste + ["litt", "ekstra"])
print(liste)

liste.sort()
print(liste)
```

Tupler

- Indexert samling av elementer
- Parallel tilordning av verdier
- Ikke-muterbar

```
# Tupler

t = ("Kristian", 23, "Datateknologi")
print(t[0])     # Kristian
print(t[:2])    # ('Kristian', 23)


navn, alder, studie = t
print(navn)                 # 'Kristian'
print(alder)                # 23
print(studie)               # 'Datateknologi


t[1] = 24       # ERROR
```

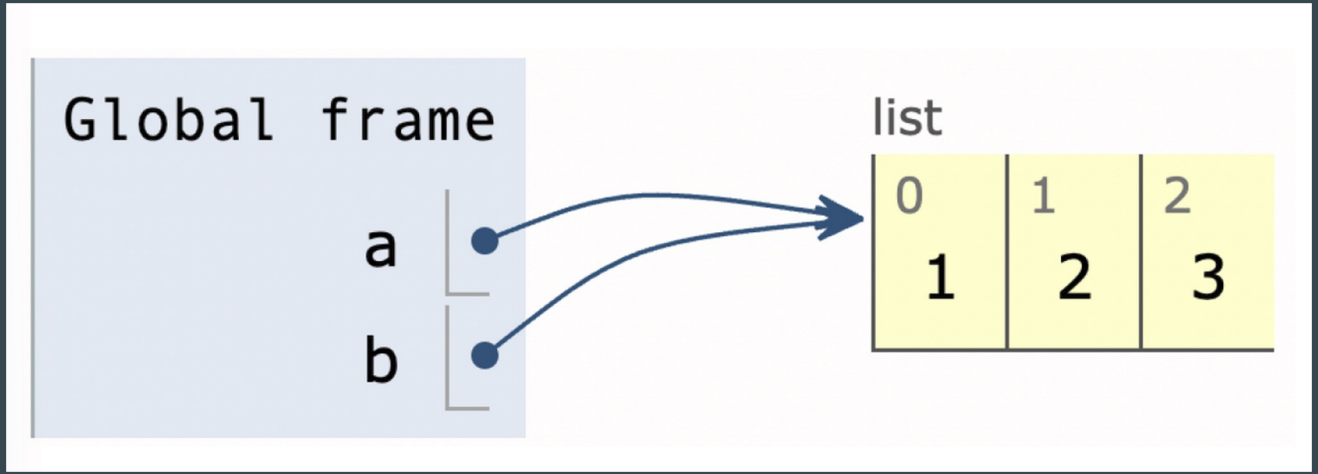
```python
my_list = [1, 2, 3, 4]
another_list = my_list
another_list[0] = 'Lykke til på eksamen'

print(my_list[0])           # Lykke til på eksamen
```

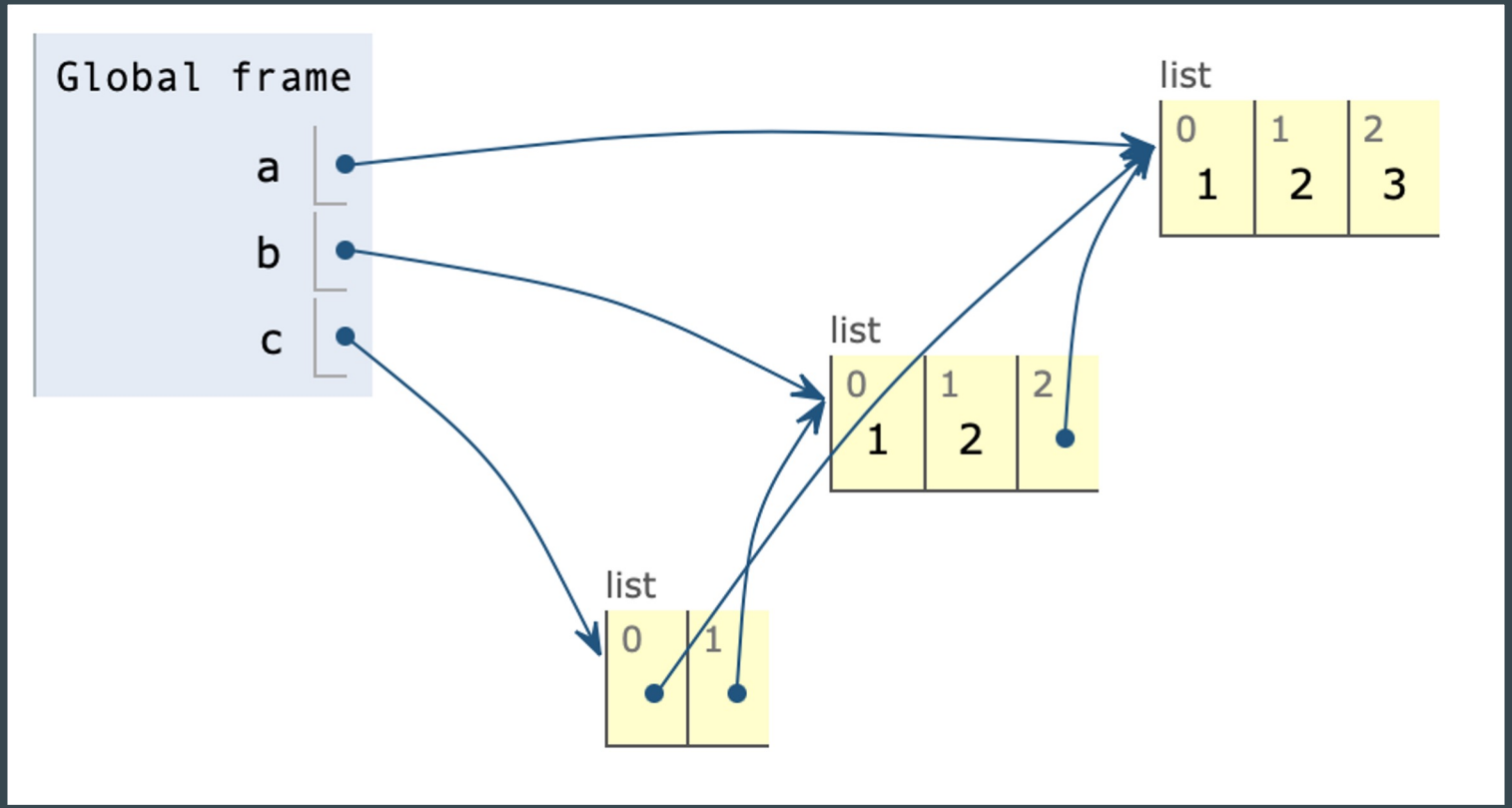
```
my_tuple = (1, 2, 3, 4)
another_tuple = my_tuple
another_tuple[0] = 'Lykke til på eksamen'

print(my_tuple[0])
```

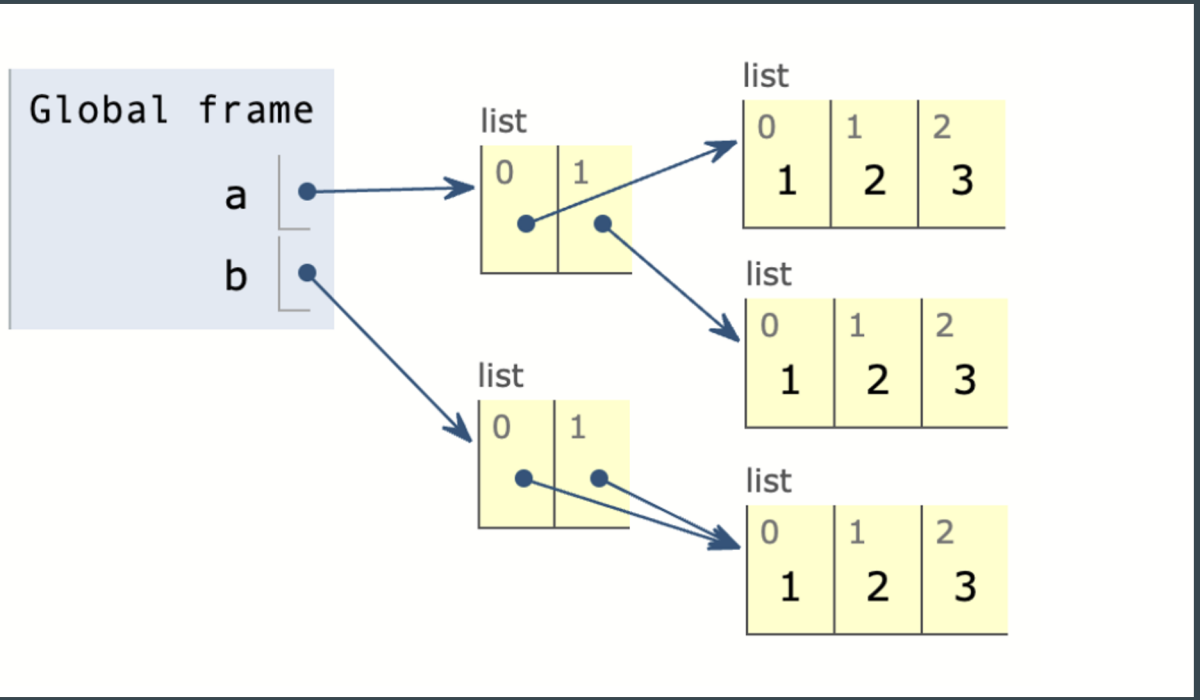
```python
my_tuple = (1, 2, 3, 4)
another_tuple = my_tuple
first, second, third, fourth = another_tuple
another_tuple = ('Lykke til på eksamen', second, third, fourth)

print(my_tuple[0])          # 1
print(another_tuple[0])     # Lykke til på eksamen
```

Destruktive vs ikke-destruktive funksjoner på lister

```
def destructive_remove_zeros(l):
    while 0 in l:
        l.remove(0)



def non_destructive_remove_zeros(l):
    res = []
    for e in l:
        if e != 0:
            l.append(e)
    return res
```


Global frame
a
b
list
0
1
1
2
2
3


Global frame
a
b
c
list
0
1
2
1
2
3
list
0
1
2
1
2
list
0
1


Global frame
a
b
list
0
1
list
0
1
2
1
2
3
list
0
1
2
1
2
3
list
0
1
list
0
1
2
1
2
3

Fasit 1

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

Fasit 2

```
a = [1, 2, 3]
b = [1, 2, a]
c = [a, b]
```

Fasit 3

```
a = [[1, 2, 3], [1, 2, 3]]
c = [1,2,3]
b = [c, c]
```

Set og oppslagsverk

•••

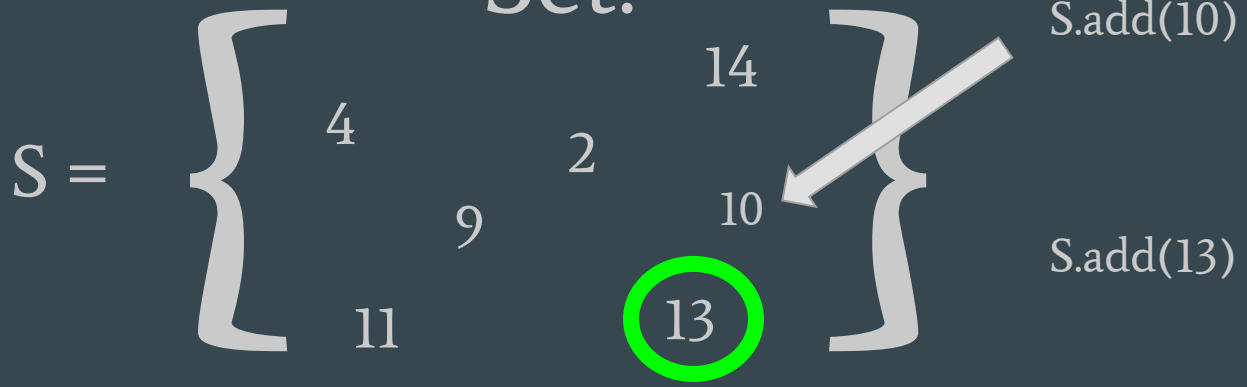
med Izaak

Lister:

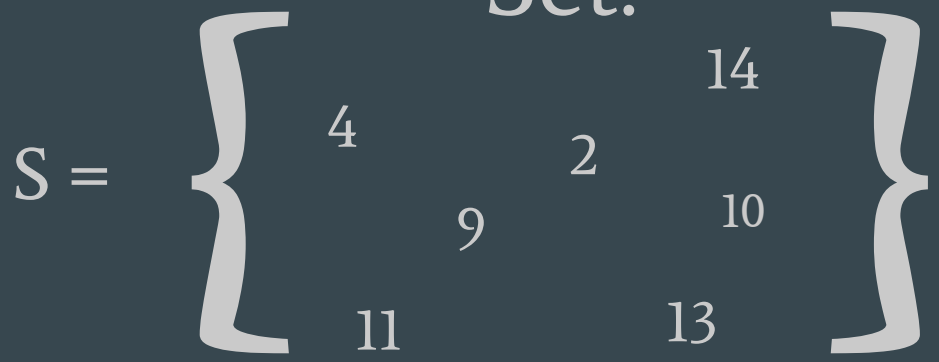
l = [“a” , 5 , True]

l[0] l[1] l[2]

Set:



Set:



Do:

```
s = set()
s.add(x)
s.remove(x)
if x in s: ... # -> True/False
for item in s: ...
len(s)
```

Do not:

```
s.append(x)
s.pop(x)
s[3]
s.count(x)
```

Dictionaries

Kandidatnummer	Karakter
133	A
134	B
135	C
136	B
137	B

Dictionaries

Kandidatnummer	Karakter
133	A
134	B
135	C
136	B
137	B

Nøkler

Verdier

Dictionaries

	List	Dictionary
Opprette	l = []	d = {}
Legge til element	l.append(x)	d[k] = x
Endre element	l[i] = y	d[k] = y
Hente ut verdi	l[i]	d[k]
Fjerne verdi	l.remove(x) l.pop(i)	del d[k]

Dictionaries

```
d = {
    113 : "A",
    114 : "B",
    115 : "C",
    116 : "B",
    117 : "B"
}
```

```
keys = d.keys() # -> [113, 114, 115, 116, 117]
values = d.values() # -> [“A”, “B”, “C”, “B”, “B”]
```

```
for key, value in d.items():
    print(f"Key: {key}, value: {value}")
```

```
Key: 113, value: A
Key: 114, value: B
Key: 115, value: C
Key: 116, value: B
Key: 117, value: B
```

Live eksempel av en counter funksjon

Filer

•••

med Izaak

Filbehandling

Hvorfor?

- Lagre informasjon i datamaskinens filsystem
- Hente inn informasjon fra filer

Hvordan?

- Tekstfiler
- CSV

Hovedsakelig to måter å håndtere filer på...

```python
filnavn = "eksempelfil.txt"

with open(filnavn, "r", encoding='utf-8') as f:
    innhold = f.read()
    print(innhold)

with open(filnavn, "w", encoding='utf-8') as f:
    ny_tekst = "Ny tekst"
    f.write(ny_tekst)
```

```python
from pathlib import Path

filnavn = "eksempelfil.txt"

filsti = Path(filnavn)
filsti.write_text("Hei, dette er tekst.")

innhold = filsti.read_text()
print(innhold)
```

```
Fødselsår,Fødselsmåned,Fødselsdato,Navn
1994,02,25,Tord
2007,12,03,Nina
2014,09,05,Lilletord
```

```
csv_fil = Path("eksempel.csv")
innhold = csv_fil.read_text()
liste_av_linjer = innhold.split('\n')
første_linje = liste_av_linjer[1].split(",")
print(første_linje)
```

```
['1994', '02', '25', 'Tord']
```

```python
csv_fil = Path("eksempel.csv")
innhold = csv_fil.read_text()
liste_av_linjer = innhold.split('\n')
første_linje = liste_av_linjer[1].split(",")
print(første_linje)
navn = første_linje[3]
dato = første_linje[2]
måned = første_linje[1]
år = første_linje[0]
print(f"{navn} hadde bursdag den {dato}. i {måned} {år}")
```

```
['1994', '02', '25', 'Tord']
Tord hadde bursdag den 25 i 02 1994
```

Nyttårsløpet - Typisk final boss på eksamen!

Livprogging!