# 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
- 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

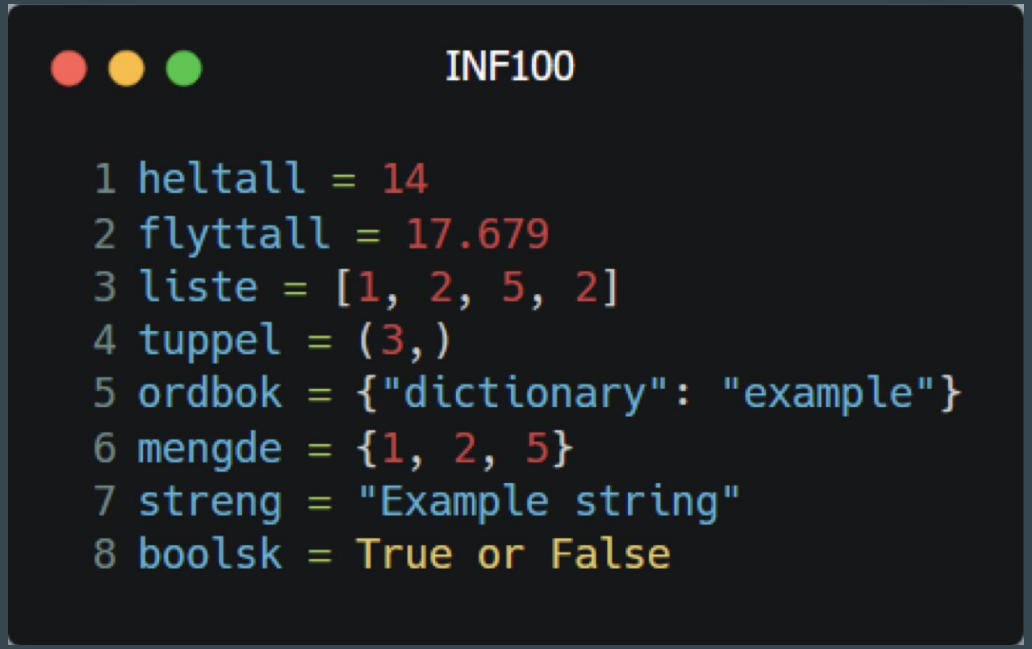
# Variabler

- 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

```
a = 420
b = 'True'
c = ['hei', 10, [False, True, False]]
d = 6.9
e = 'Gleder meg til sommerferie<333'



# Hvilken type får de følgende uttrykkene?

(1) a
(2) b
(3) c[2]
(4) e * d
(5) b * a
(6) c[2][0] == a
```

# Booleans & if statements

•••

med Sander

# Boolske verdier / Boolske uttrykk

Matematiske verdier:

1, 23, 100, -3.14, ... , ∞

Boolske verdier:

True, False

Matematisk uttrykk = Enkeltverdier kombinert med matematiske operasjoner:

(20 + 100) / 12

Boolske uttrykk = Enkeltverdier (boolske eller matematiske) kombinert med boolske operasjoner:

5 <= 10 AND a / b == 3

Uttrykk kan evalueres til en ny matematisk verdi

(20 + 100) / 12 = 10

Et uttrykk MÅ kunne evalueres til en ny boolsk verdi for å være et gyldig boolsk uttrykk

5 <= 10 AND 1 > 2 OR a == a

= True AND (False OR True)

= True AND True = True

Matematiske operasjoner:

+, -, *, /, //, %

Matematiske operasjoner som produserer en sannhetsverdi (True/False)

== , >, >=, <, <=, !=

Boolske operasjoner:

AND:

| | T | F |
|---|---|---|
| T | T | F |
| F | F | F |

OR:

| | T | F |
|---|---|---|
| T | T | T |
| F | T | F |

NOT:

| ¬T | ¬F |
|---|---|
| F | T |

## 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()
```

# Oppgave

https://codingbat.com/prob/p143951

https://codingbat.com/python/Logic-2

# 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:
        print(x**2)
```

# Finn feilen

```
def remove_zeros():
    l = [0,3,5,6,2,1,0,1,0,4,5]
    for i in range(len(l)):
        if l[i] == 0:
            l.pop(i)
```

# Funksjoner

med Sander

Matematisk funksjon:

$f(x) = 2x$

Generalisert:

funksjonsnavn(input) = output

Opprette en funksjon:

$f(x) = 2x$

Bruke/kalle på funksjonen:

$2 + f(2) = 2 + \mathbf{4} = 6$

Python funksjon:

```
def f (x) :
    return 2*x
```

Generalisert:

```
def <funksjonsnavn> ( <parameter1> , <parameter2>):
    KODEBLOKK
```

Opprette en funksjon:

```
def f(x):
    return 2*x
```

Bruke/kalle på funksjonen:

```
a = 2 + f(2)
```

# Parameter vs Argument

## (To sider av samme sak...)

Parameter!

```
def plus_one_times_two( number )
    return  ( number + 1 ) * 2
```

Argument!

(Parameteret “number” får tilegnet verdien 5, som er argumentet)

```
a = 5

a_increased = plus_one_times_two(a)

print(a_increased) -> 12
```

Et funksjonskall i koden erstattes med funksjonen sin returverdi.

val1 = 5

val2 = 8

Det vi skriver:

```
min_value = min(val1, val2)
```

Det som skjer:

```
min_value = min(5, 8) -> evalueres til 8
    min_value = 8
```

```
1  print(print("Jeg digger kræsjkurs!"))
```

```
Jeg digger kræsjkurs!
None
```

# OBSOBS

En funksjon kan kun returnere én gang på en kjøretid

Så snart datamaskinen leser en return-statement stoppes funksjonskallet

Eksempel:

```
#Count all elements in l (2d list)
def count(l):
    sum = 0
    for row in l:
        for number in row:
            sum += number
        return sum

l = [[1,2] , [3,4]]

print(count(l))
```

```
#Count all elements in l (2d list)
def count(l):
    sum = 0
    for row in l:
        for number in row:
            sum += number
    return sum

l = [[1,2] , [3,4]]

print(count(l))
```

# 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)
```

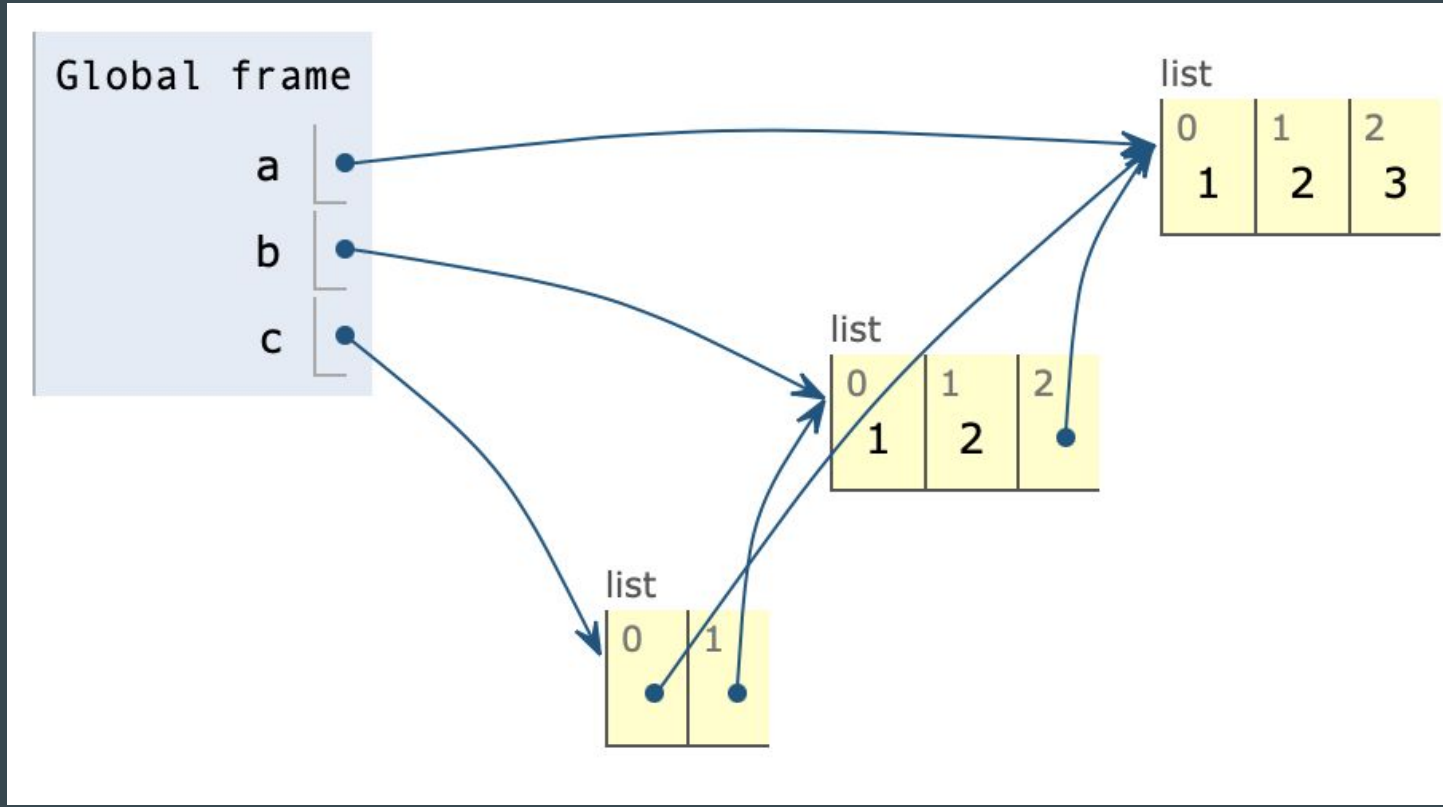
# Tupler

- 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
```

# 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
c
list
0
1
2
1
2
3
list
0
1
2
1
2
list
0
1

# Set

...

med Sander

# Lister:

l = [ “a” , 5 , True ]

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

# Set:

# Set:

S = { 4, 14, 2, 9, 10, 11, 13 }

## Do:

S = set()

Set.add(x)

Set.remove(x)

if x in S -> True/False

for item in S

len(S)

## Do not:

S.append(x)

S.pop(x)

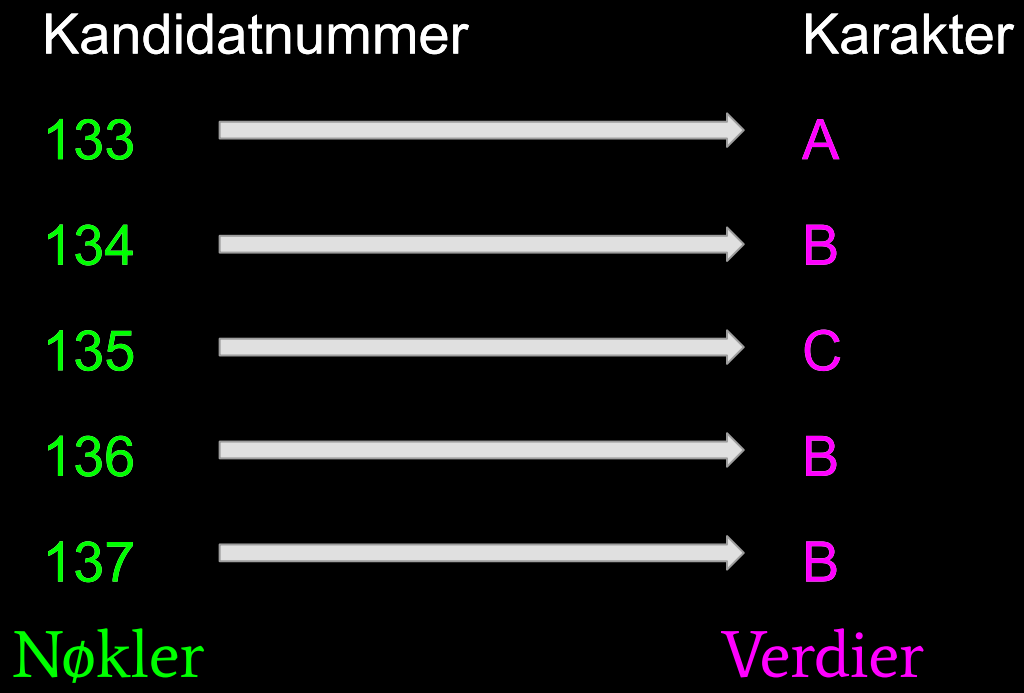
S[3]

S.count(x)

# Dictionaries

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

# Dictionaries



# Dictionaries

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

# Dictionaries

```
d = {
        113 : “A”
        114 : “B”
        115 : “C”
        116 : “B”
        117 : “B”
}
```

d.keys( ) -> [113, 114, 115, 116, 117]

d.values( ) -> [“A”, “B”, “C”, “B”, “B”]

# Filbehandling

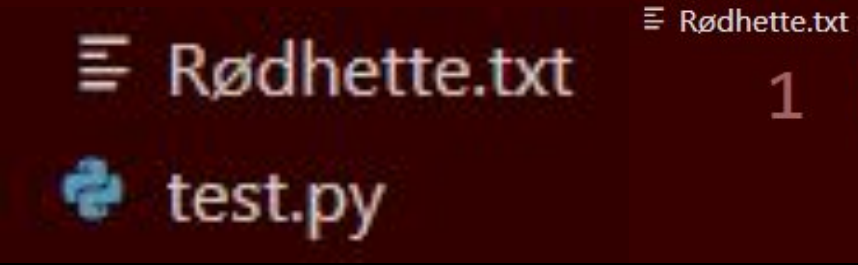
Hvorfor?

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

Hvordan?

- Tekstfiler
- CSV

```python
with open("Rødhette.txt", "r", encoding="utf8") as f:
    content = f.read()

with open("Rødhette.txt", "w", encoding="utf8") as f:
    content = f.write("Det var en gang en ...")
```




Rødhette.txt

1 Det var en gang en ...

```
1  Fødselsår,Fødselsmåned,Fødselsdato,Navn
2  1994,02,25,Tord
3  2007,12,03,Nina
4  2014,09,05,Lilletord
```

```
with open("Fødselsdatoer.txt", "r", encoding="utf8") as f:
    content = f.read()
    list_of_lines = content.split("\n")
    first_entry = list_of_lines[1].split(",")
    print(first_entry)
```

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

```python
with open("Fødselsdatoer.txt", "r", encoding="utf8") as f:
    content = f.read()
    list_of_lines = content.split("\n")
    first_entry = list_of_lines[1].split(",")
    print(first_entry)
    navn = first_entry[3]
    dato = first_entry[2]
    måned = first_entry[1]
    år = first_entry[0]
    print(f"{navn} hadde bursdag den {dato} i {måned} {år}")
```

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