

# PRESEDENS

INF100

HØST 2025

Torstein Strømme

# HJELP, DET ER EKSAMEN!

Bruk kursnotatene, ikke KI	8
Gjør tidligere eksamensoppgaver	7
Forklare løsninger for hverandre	6
Gjør tidligere laber på nytt uten hjelp	4
Begynn tidlig med repetisjon	3
Øv på å skrive pseudokode/skrive med «grove steg»	3
Fokuser på grunnleggende konsepter	2
CodingBat og lignende	2
Lag egne notater til å ta med på eksamen	1
Delta på krasj-kurs	1
Delta på gruppetimer	1
Skriv ned underveis det som er vanskelig	1
Øv på manuell kodesporing	1

# VANLIGE FEIL PÅ EKSAMEN

- Å ikke prøve
- Manglende identifikasjon av delproblemer
  - For mange ting på samme linje
  - For mange ting i samme funksjon (manglende bruk av hjelpefunksjoner)
- Dårlige variabelnavn
  - `for i in a:` vs `for i in range(len(a)):`

# VANLIGE FEIL

- Precedens
  - `x and y in z`
  - `x == 3 or 4`
- Funksjoner
  - `print` vs `return`
  - `return` i destruktive funksjoner
- Lister
  - indeks vs. element
  - løkker over indeks vs elementer
- Løkker
  - For tidlig `return`
  - Hva skal gjøres én gang, hva skal gjentas flere ganger?

# EKSAMENSSTRATEGI

- Les gjennom alle oppgavene
  - Ikke kast bort tid på et problem du ikke har en klar plan for før du har lest nøye gjennom alle oppgavene.
- Første gang du leser problemet: identifiser delproblemer
  - Skriv ned kommentarer/idéer. Hva skal gjentas flere ganger? Hjelpesfunksjoner gjør hva?
  - Selv om du ikke klarer løse hele problemet, kjenner du kanskje igjen en del av det du klarer å løse.
- Prioriter oppgaver du kan løse raskt og oppgaver med mye poeng.

# EKSAMENSSTRATEGI

- Bruke selvbeskrivende variabelnavn
  - Lettere for deg selv å forstå hva du driver med
  - Lettere for sensor å forstå hva du driver med
- Spis godt før du starter
- Sov godt før du starter
- Gå en lang tur dagen før (men ikke så sent på kvelden at du ikke får sove)

“99% of people fail to solve this problem”

$$6/2(1 + 2)$$

RECAP

UTTRYKK

`(x - y) * 2 + foo(x) > 42`

```
res = (x - y) * 2 + foo(x) > 42
```

```
if (x - y) * 2 + foo(x) > 42 :  
    ...
```

```
while (x - y) * 2 + foo(x) > 42 :  
    ...
```

`((x - y) * 2 + foo(x) > 42) and foo(y) < 95`

$$(x - y) * 2 + \text{foo}(x) > 42$$

også et uttrykk

også et uttrykk

$$(x - y) * 2 + \text{foo}(x) > 42$$

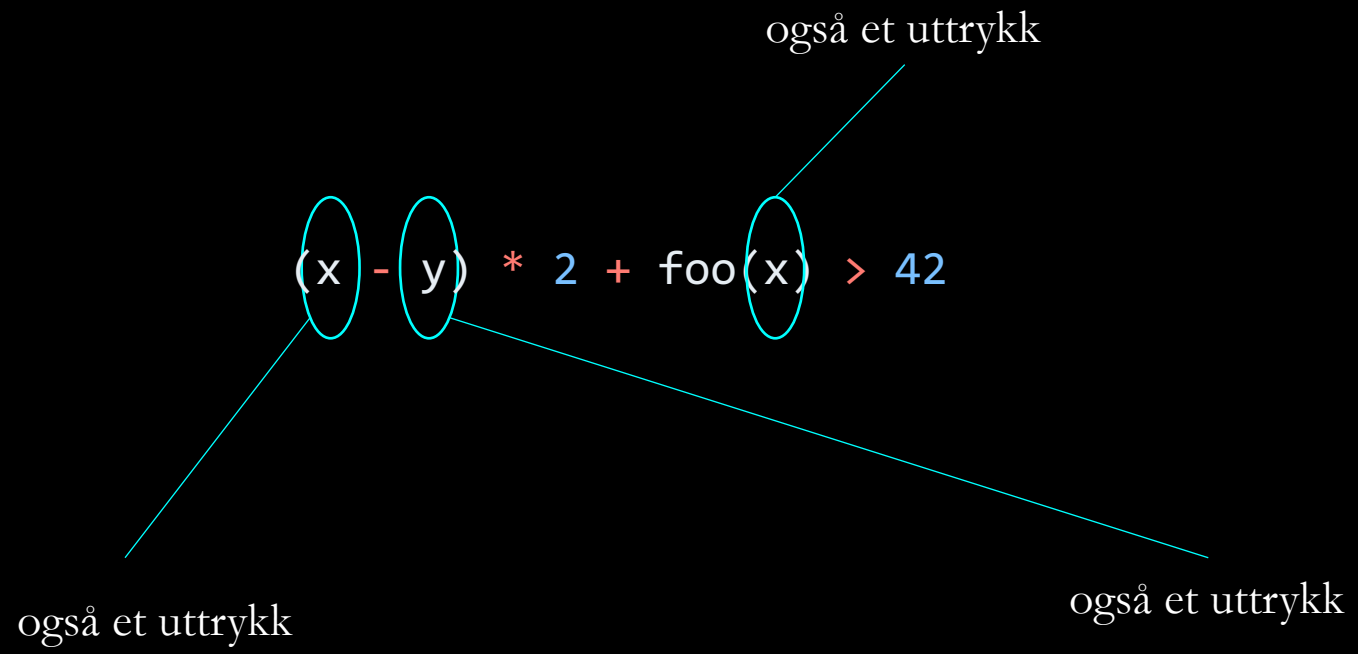
også et uttrykk

også et uttrykk

$$(x - y) * 2 + \text{foo}(x) > 42$$

også et uttrykk

også et uttrykk



# HVA ER ET UTTRYKK?

- Verdier er uttrykk
- Variabler er uttrykk
- Funksjonskall er uttrykk

```
42      True      'hello'  
  
i      number_of_students  
  
max(x, y)    a.sort()
```

- Hvis A og B er uttrykk og  $\Omega$  er en operator så er  $A \Omega B$  også et uttrykk

```
42 + foo(x)      (42 + foo(x)) * 95
```

- Hvis A er et uttrykk, så er (A) også et uttrykk

```
(42 + foo(x))
```

```
+ - * / // %  
< > == != in  
and or not* ...
```

\*det er egne regler for «unary»-operatorer som ikke beskrives her

# EVALUERING AV UTTRYKK

- Hvis et uttrykk er en verdi, er uttrykket ferdig evaluert.
- Hvis uttrykket er en variabel, evalueres uttrykket til den verdien variabelen referer til.
- Hvis uttrykket er et funksjonskall, evalueres uttrykket til returverdien
- Hvis uttrykket er i en parentes, evaluer uttrykket inne i parentesene
- Ellers:
  - Velg operatoren utenfor en parentes med lavest precedens lengst til høyre<sup>★</sup>, og klipp uttrykket i to:
    - Evaluer venstresiden av uttrykket
    - Evaluer høyresiden av uttrykket
    - Kombiner verdiene med operatoren

★ unntak for \*\*-operatøren (eksponentiering), der velges den lengst til venstre.

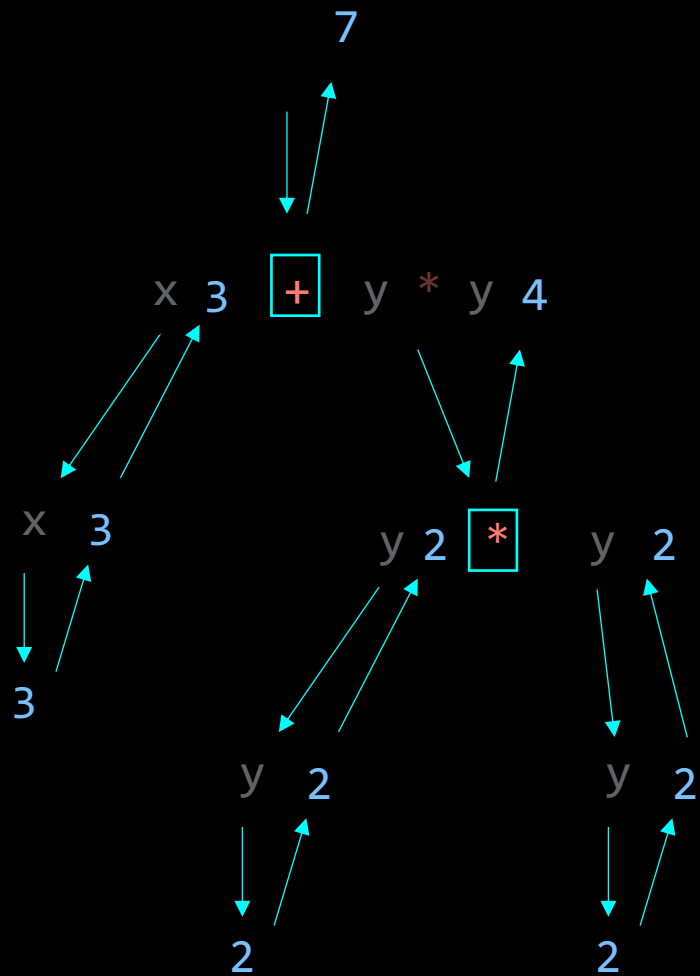
EKSEMPEL

∨ VARIABLES

∨ Locals

x = 3

y = 2



EKSEMPEL

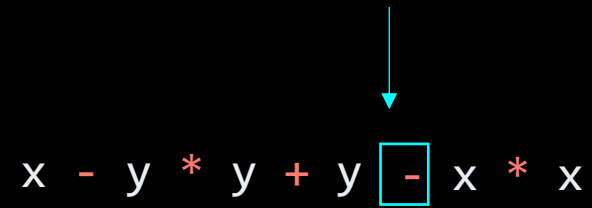
∨ VARIABLES

∨ Locals

x = 3

y = 2

x - y \* y + y - x \* x



∨ VARIABLES

∨ Locals

x = 3

y = 2

x - y \* y + y - x \* x



∨ VARIABLES

∨ Locals

x = 3

y = 2

x - y \* y + y - x \* x



x - y \* y + y

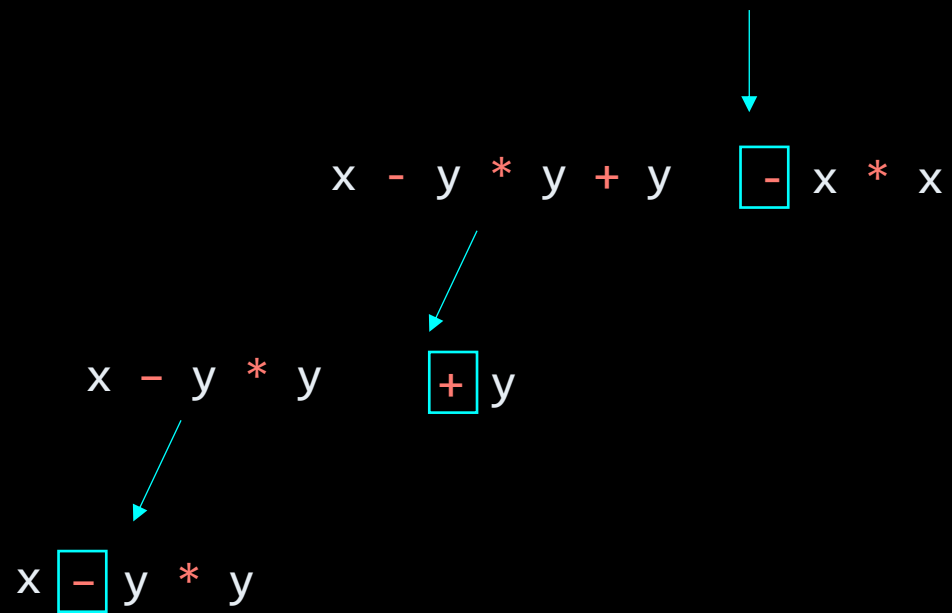


∨ VARIABLES

∨ Locals

x = 3

y = 2

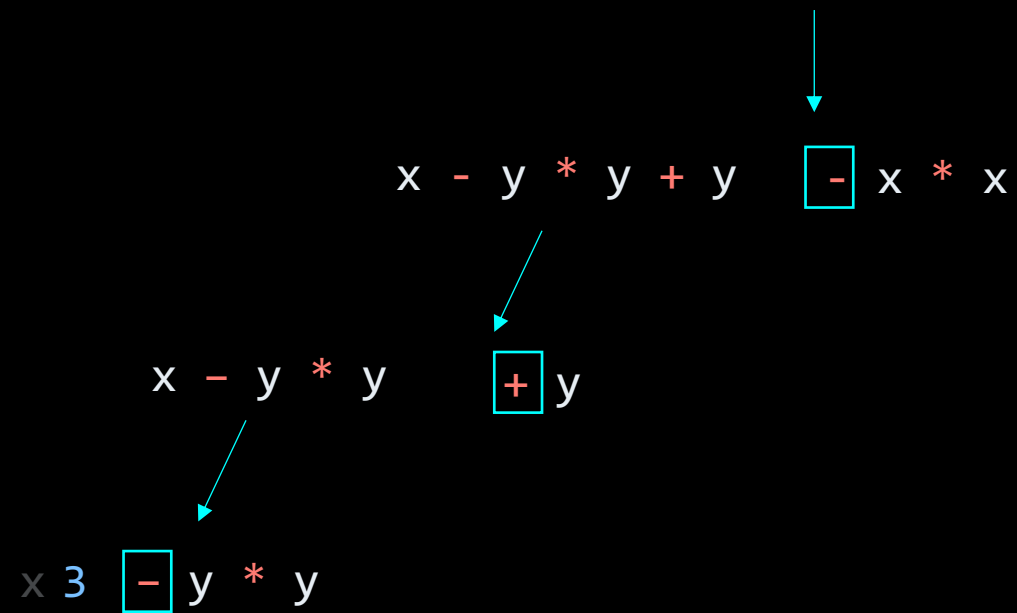


∨ VARIABLES

∨ Locals

x = 3

y = 2

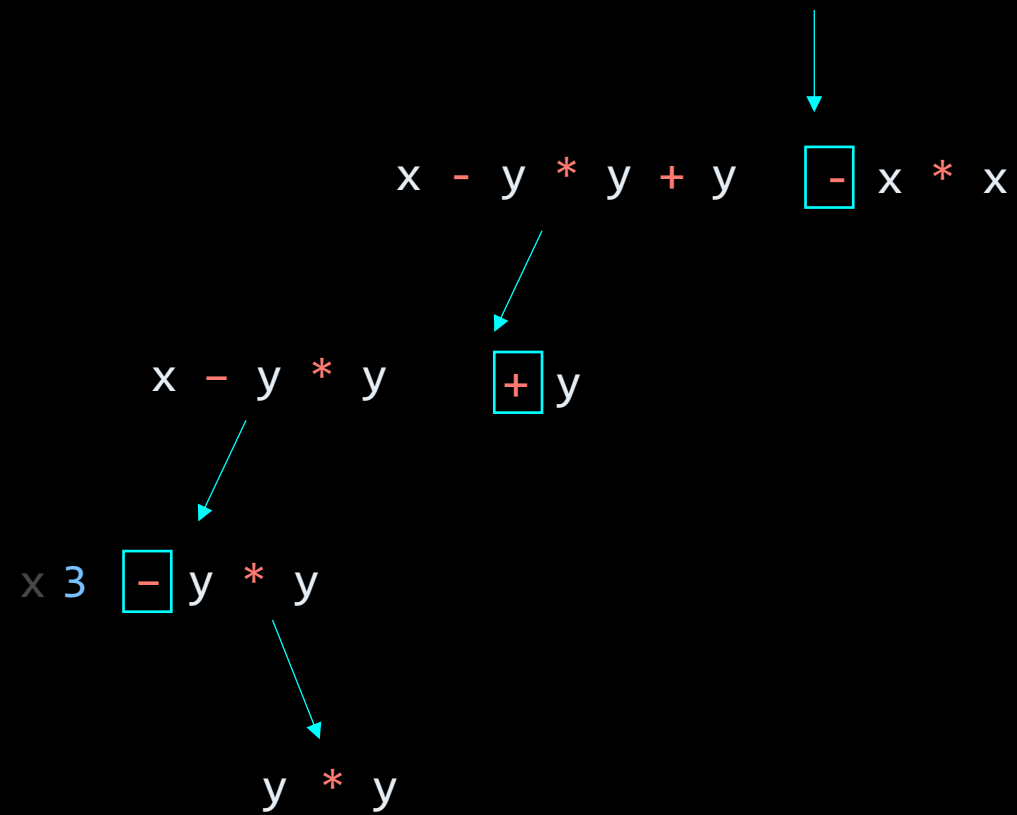


∨ VARIABLES

∨ Locals

x = 3

y = 2

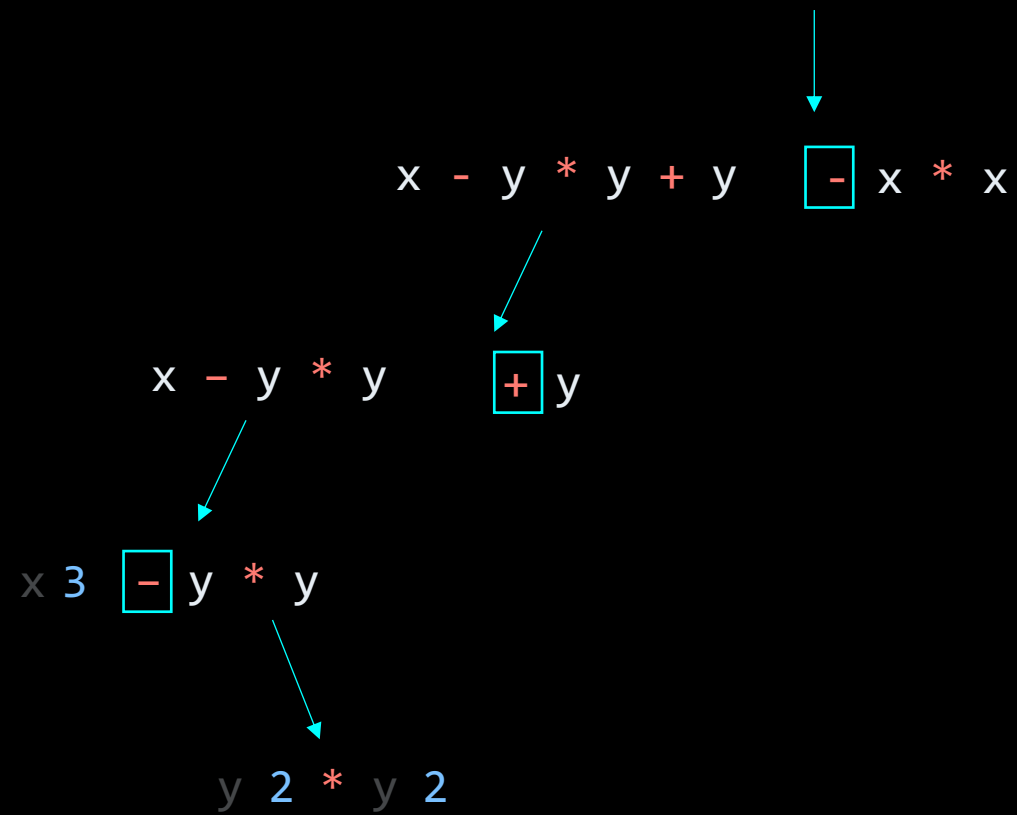


∨ VARIABLES

∨ Locals

x = 3

y = 2

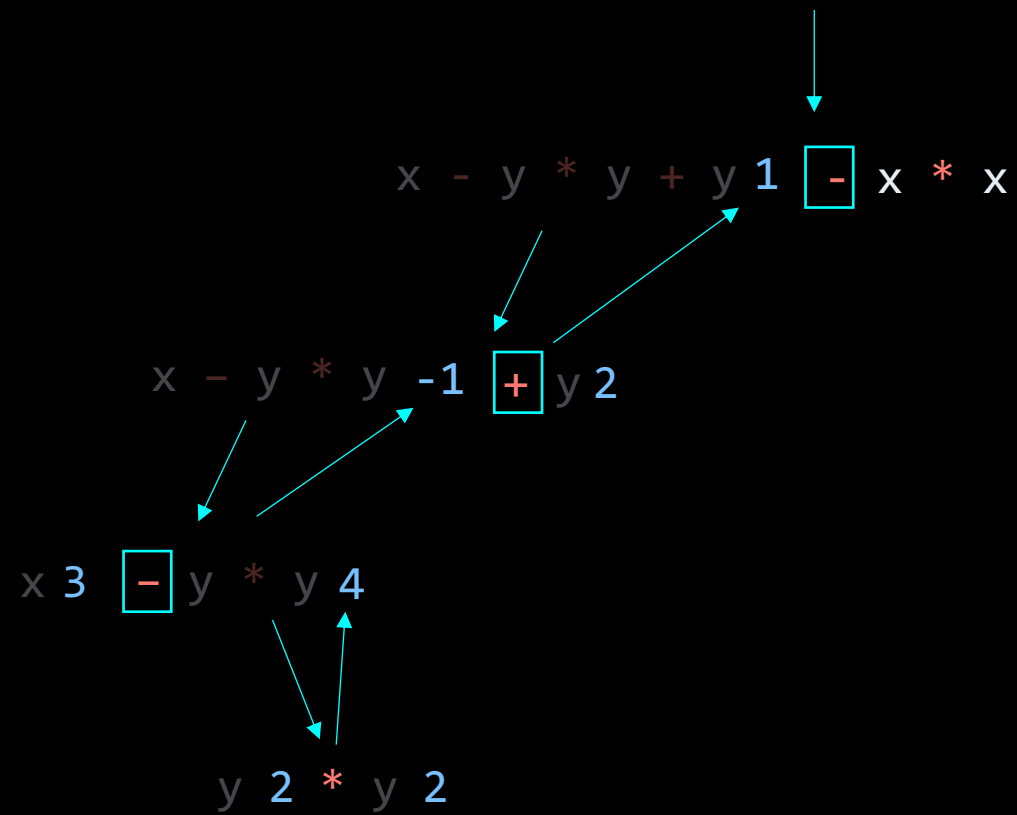


✓ VARIABLES

✓ Locals

x = 3

y = 2

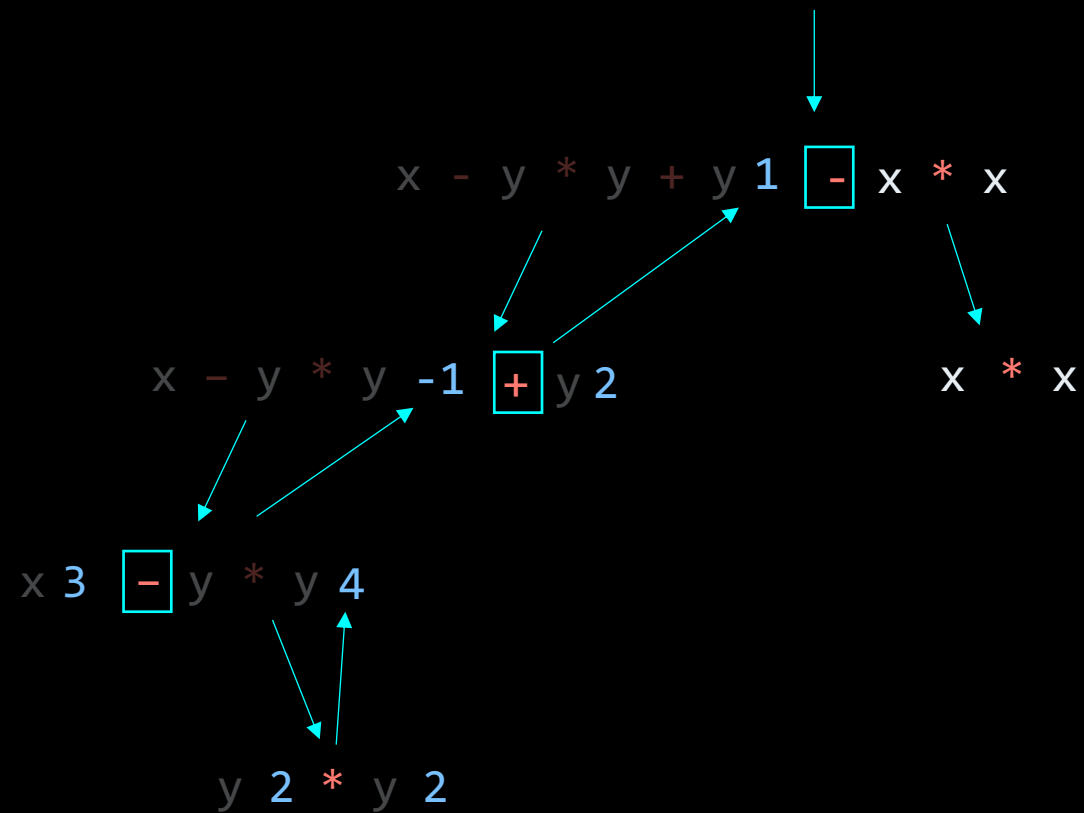


✓ VARIABLES

✓ Locals

x = 3

y = 2

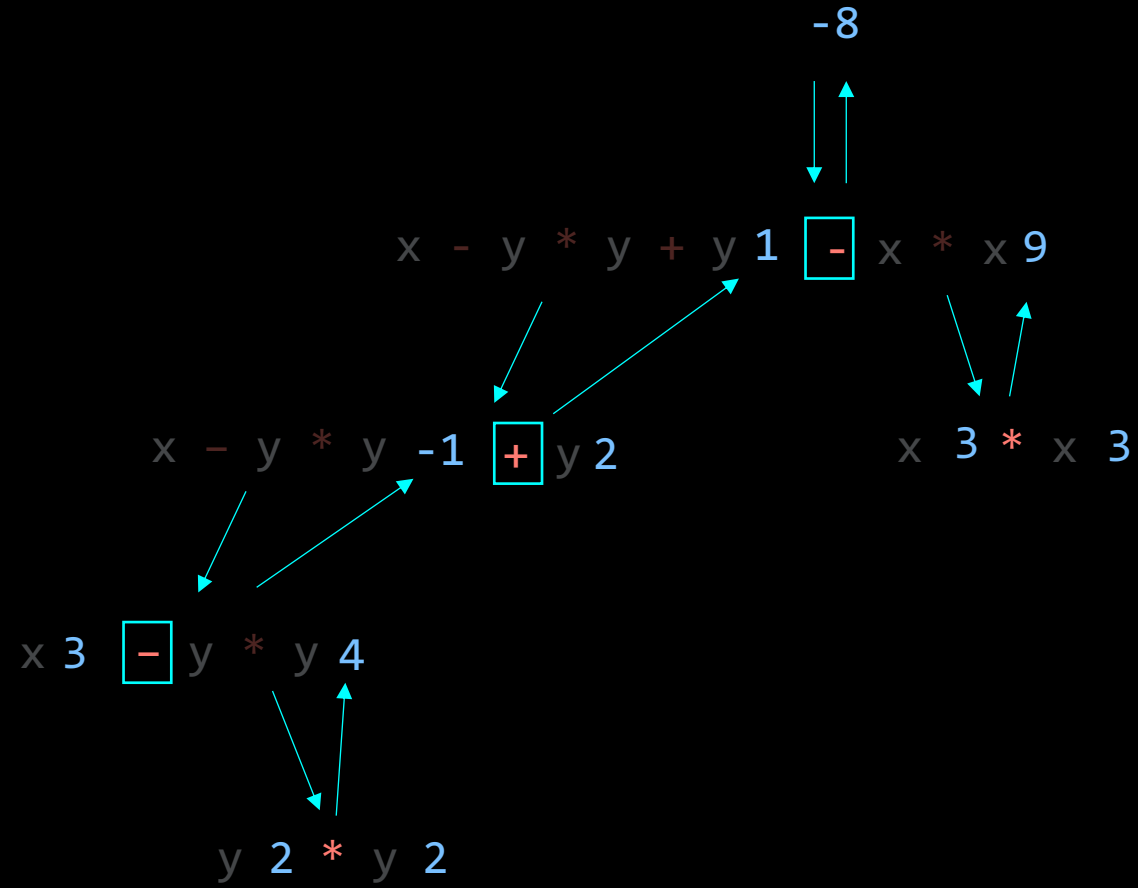


✓ VARIABLES

✓ Locals

x = 3

y = 2



# EVALUERING AV UTTRYKK

- Tommelfingerregel:
  - Operasjoner med høy presedens utføres først
  - Operasjoner med lik presedens utføres fra venstre mot høyre (unntak: \*\*)
  - Evaluering av uttrykk på «samme nivå» evalueres fra venstre mot høyre
  
- Takeaway:
  - Benytt parenteser!

# LOGISKE OPERATORER: SANNHETSTABELLER

x	y	x or y
True	True	True
True	False	True
False	True	True
False	False	False

# LOGISKE OPERATORER: SANNHETSTABELLER

x	y	x or y	x and y
True	True	True	True
True	False	True	False
False	True	True	False
False	False	False	False

# KORTSLUTNINGSEVALUERING

## ✓ VARIABLES

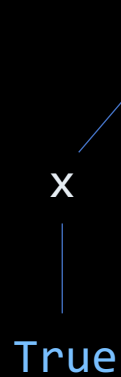
### ✓ Locals

x = True

y = False

z = True

x or ((x or y) != ((x and y) or (y == z)))



# KORTSLUTNINGSEVALUERING

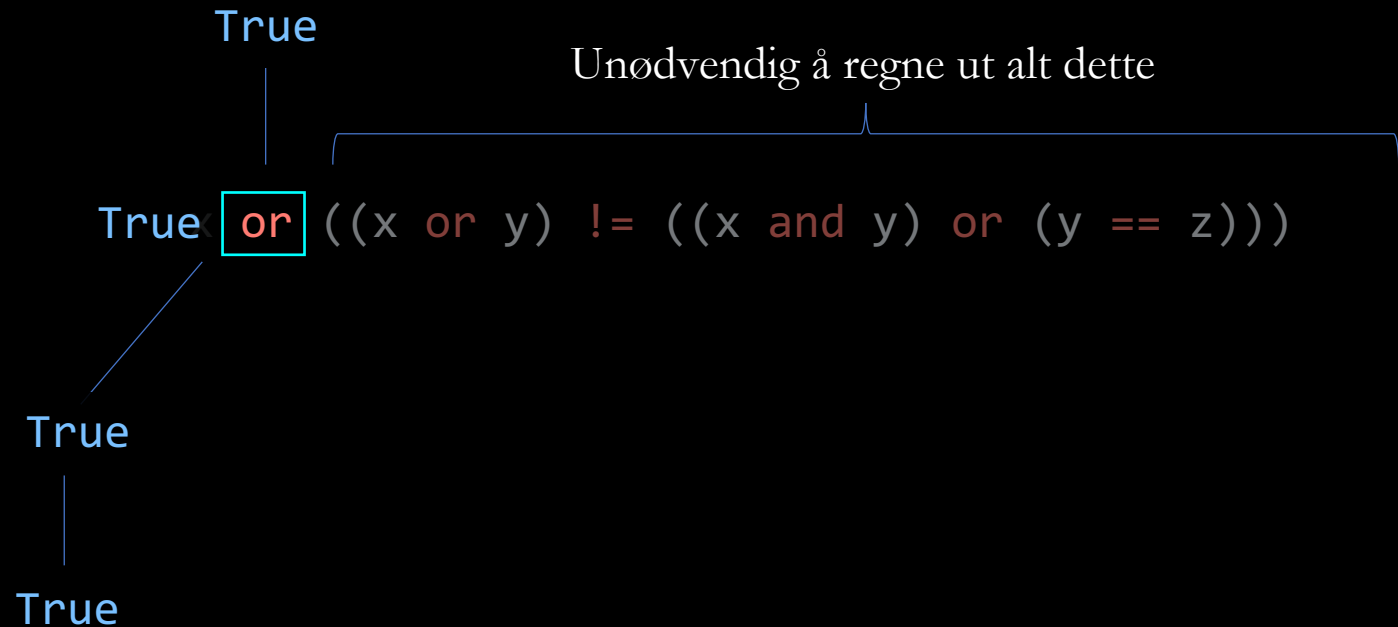
## ✓ VARIABLES

### ✓ Locals

x = True

y = False

z = True



# KORTSLUTNINGSEVALUERING

```
def starts_with_x(s):  
    return len(s) > 0 and s[0] == 'x'
```

```
s = ''  
if starts_with_x(s): # Krasjer ikke!  
    print('yay!')
```

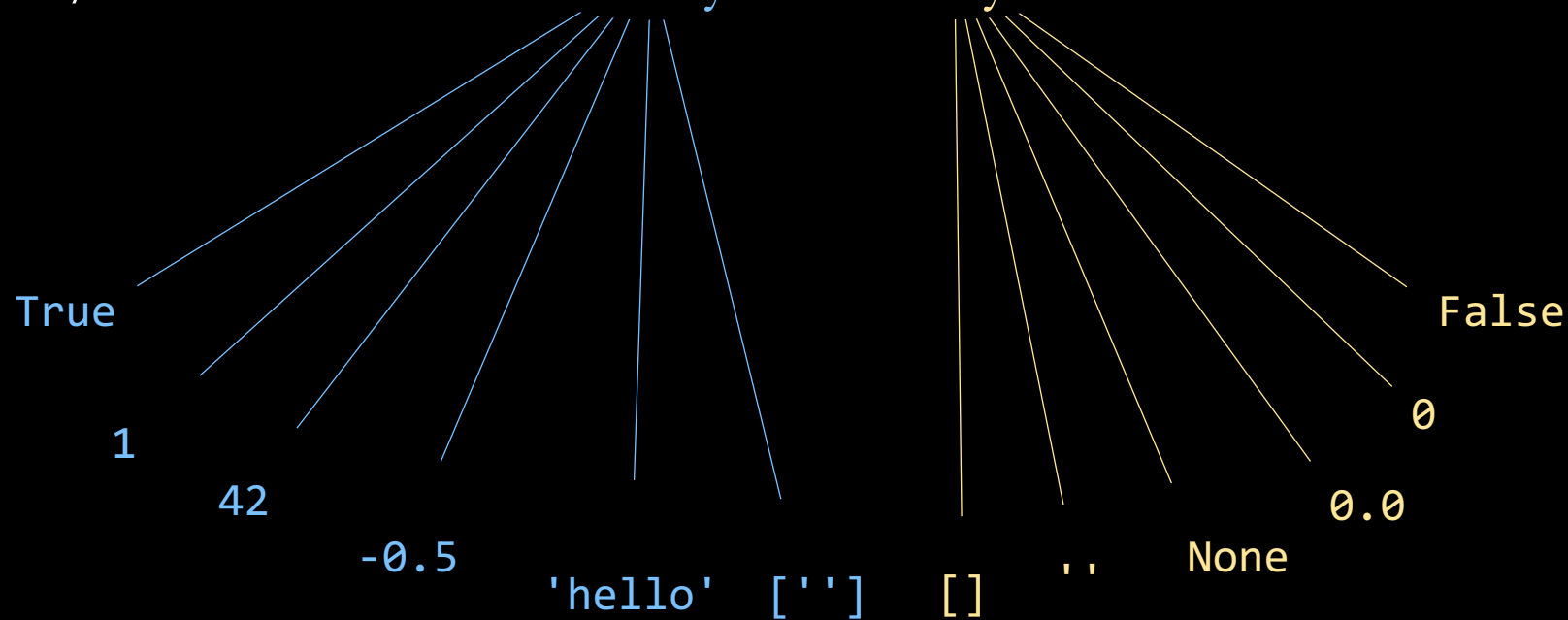
# KORTSLUTNINGSEVALUERING

- Venstre side av **or** –uttrykk er **True** → høyre side evalueres ikke
- Venstre side av **and** –uttrykk er **False** → høyre side evalueres ikke

# TRUTHINESS

«sannhetsaktighet»

Alle objekter/verdier er enten **truthy** eller **falsy**



# TRUTHINESS: OR

```
foo = 5  
bar = True  
x = (foo or bar)
```

venstresiden er truthy; derfor evalueres uttrykket til venstre side 5

Hva `lft or rgt` egentlig evaluerer til:

Hvis `lft` er truthy, evaluer til `lft`;  
ellers evaluer til `rgt`

# TRUTHINESS: AND

```
foo = True  
bar = 5  
x = (foo and bar)
```

venstresiden er truthy; derfor evalueres uttrykket til høyre side 5

Hva `lft and rgt` *egentlig* evaluerer til:

Hvis `lft` er truthy, evaluer til `rgt`;  
ellers evaluer til `lft`

# SANNHETSTABELLEN STEMME

x	y	x or y	x and y
Truthy <sub>x</sub>	Truthy <sub>y</sub>	Truthy <sub>x</sub>	Truthy <sub>y</sub>
Truthy <sub>x</sub>	Falsy <sub>y</sub>	Truthy <sub>x</sub>	Falsy <sub>y</sub>
Falsy <sub>x</sub>	Truthy <sub>y</sub>	Truthy <sub>y</sub>	Falsy <sub>x</sub>
Falsy <sub>x</sub>	Falsy <sub>y</sub>	Falsy <sub>y</sub>	Falsy <sub>x</sub>

# SANNHETSTABELLEN STEMMER

x	y	x or y	x and y
42	'hei'	42	'hei'
42	''	42	''
0	'hei'	'hei'	0
0	''	''	0